Zutha Ideas

# Tue 6-Jan-09

* Website designed to collect useful suggestions, products (free or commercial), reviews etc.
* Users submit items with suggested categorization
* Each item has an attached forum, change log (changes to item or description by poster) and score.
* Users can vote for items (score them)
* Items with higher scores are promoted in website
* User that submit items that gain a high score gain rank and prowess in the site. Intention is to create competition which drives people to find really useful “stuff” to add to the site under their name.
* Section for feedback about site itself – suggested improvements or solutions to known challenges I have published.
* Entire site under hierarchical organisation: Home 🡺 <major categories like “feedback”, “site info”, “advanced search”, “Items”> 🡺 etc...
* Users can be voted on for their forum entry relevance and posted item quality. Users with a higher rating are given a higher weighted vote. Hence there is an incentive to contribute to the website because it yields public influence and potentially commercial power.
* Items that advertise a commercial product will be labelled as “unpaid commercial” will not be visibly ranked and will be placed at the bottom of all lists. Users may still vote on the item but votes will not be shown publicly. The poster of the item will see the rating of the item based on these hidden votes however. This way a person that posts such an item can lobby the company in question to sponsor the item for advertising benefit, using its ranking to prove its merit if they wish. Paying a monthly fee for the item will allow it to enter the standard ranked pool. It is likely that commercial items will be overlooked, but popular ones will not be and they are the ones that matter.
* Users can host commercial items directly through the site for free, but they pay a small percentage of actual sales as commission.
* I will add my own purchasable items to the site in exactly the same way (if I think they are good enough and worth money).
* Site design should be simple, fast to load, and compatible with mobile devices.
* An item in a list under a particular category should display a succinct title; brief description; public score (derived from no. Votes and average user score with higher ranked voters given a higher weighting); an admin score, given by me (if present); the Poster’s ID and their rating.
* Users may only vote on any item, User, or forum posting once, but they may change or remove their vote later. A system needs to be derived to prevent bots accumulating accounts to leverage particular items. E.g. up to 5 accounts per IP address and 1 per email address. More would be required if this issue became serious however. Possibly some kind of unique publically available ID number, like driver’s license number. Another step would be to delete inactive accounts and their scores, or maybe just reduce the weighting of inactive Users significantly, but have their scores show up in an Item’s full score list still so that advanced users can assess the relevance of inactive users’ scores themselves. Obviously an effective anti-bot check would be needed on account creation. I sense though that the most effective measure would be to reduce the weighting of Users that only vote and don’t contribute. It may be simply too hard to prevent mass production and use of accounts by people or bots. If the value of votes was on user contribution and their peer rating (quality), instead of quantity, then there is no way to abuse this. Except that the value of an inactive User has to be worth something, so with enough mass produced votes a significant influence could be attained. Maybe, for a particular Item, the impact of low rated votes could level off after there are a large enough number of them.
* A provision could be created for advanced Users to create custom ranking systems with a built in editor. At a basic level they could choose whether to place more emphasis on my vote than the public vote. At a more advance level they could alter the degree to which rating affects the value of votes and even alter the parameters for calculating a User’s rating. Premade Custom Ranking Systems could be made available under their own category, subject to the same voting systems as everything else.
* Absolutely everything should be allowed as an Item. Such things as feedback about the site, reviews of anything in the world, homemade programs, literature and other creative items, media, political analysis... all would be put in their relevant category and have their own forum and score attached.
* Forum entries should be voted on and ranked as well. Users can organise entries by rank as well as time posted, and filter by User or other parameters. All entries should be self-contained so any entry that references another should link that entry or a part of it, to their reply so that it appears in a foldable section of their entry.
* An effort should be made to be open about the ranking structures and the financial influence that is involved. The selling point of the site should be that rankings are based entirely on peer review and all feedback entirely 2-sided (or more). Commercial products do not start equal and get raised above free products by the payment of a premium. Instead, commercial products are given a disadvantage by default and paying money (and a specific, limited amount), only lifts the status of the item to the level of everything else. Thus Users can be confident that everything they find is being given an accurate peer review.
* Leader boards: highest ranked users in various categories, users with most premium items
* Prizes for Leader board winners over a set period (subject to website income)
* User ranking affects the weighting of votes on other Users. Overall effect of weighting is that those who are voted to know what they are talking about get more say in declaring who else knows what they are talking about.
* Certain, (maybe all) categories should have their own rating sub-category so that users have an overall rating and hierarchical sub-ratings for different categories
* Items can appear in multiple categories. The poster’s score on an item affects his rating for all the categories the item is relevant to
* Every item (/category) has links to related (maybe only directly linked) categories
* Posters can define search keywords for their item
* Search results can be organised by Category results, item keyword results, item title/description results, item attachment results (including linked web pages), item forum results, Users. Item results can also be organised into their categories
* Limited page space for item. Title, description only. Content stored within attachments (including linked web pages). Forum will be directly under the item info and description. Option to “freeze panes” on the item info and description when scrolling down the forum.
* On search with no results: suggest user search internet and post his discoveries here as briefly as he likes. He can expand on his post later if he chooses.
* Main interface: logo and search box at very top. Just under – category directory (like vista, with arrows after every category listing sub-categories).
* Make lots of site features customizable. Profiles can be saved. A profile can be assigned to mobile device bowsing. Must have an account to store settings. Without an account, only certain settings (like whether Item info and description is freeze panes/how much of forum post content is shown in the list before that post is focused) can be configured and settings only last that session.
* Data Structure: related tables
  + Categories: ID (autocount); directory (subcategory location in hierarchy); Name; description; score
  + Items: ID (autocount); name; categories (list of cat. Ids); desc.; poster; attachments (including web links); User score; admin score
  + Users: ID (autocount); email (login/username); password; Name; rating
  + Votes: type (category/Item/User/forum post); ID (within type); User ID (of voter); score (out of 100); data voted
  + Forum posts: ID (autonumber); User ID (of poster); body text; score; linked posts
* Encourage users to post in forums when they have purchased a product because of learning about it through this website. Be open about the fact that such testimonials will help us sell more subscriptions to companies and help us develop the website. Also, the more commercial items that are paid for, the more effective and relevant the site’s information.
* Make subscriptions for commercial items increase with item score. Anytime a commercial item stops being paid for, its score is hidden and it is placed at the bottom of lists again (in random order with other commercial items). Forum posts remain as normal however.

# Jul-09

* Main properties I want my item rating process to have:
  + Those who contribute ideas/content that lots of people like gain more influence to decide which ideas are good.
  + Those who rate items highly which subsequently become very popular gain more influence.
  + There is an incentive to engage in the rating process.
  + There can be a substantial financial return from owning a very popular item. This makes it realistic for information to be made freely available which is expensive to make. E.g. Movies. (This goal is obviously ambitious and may not be realistic to reliably produce this environment, but I want to search for a way to do this because it would be incredibly valuable to society to find one).
* Items are valued based on a currency system. Users can earn currency by contributing value to the site and use this currency to invest in items that they like. Investing in an item boosts its rank, so this is like voting on it.
* 10% of each investment goes to the item owner. 90% is divided up among previous investors in that item in proportion to the amount they invested. This system creates an incentive for users to invest in items that they believe will become popular.
* All currency is created by admin. No currency is ever automatically created or it would be possible for bots to produce potentially unlimited amounts of currency.
* All new users start with zero currency.
* When site first starts off, I will inject currency into the system by giving it to owners of items I think are valuable contributions. I will continue doing this whenever I see items that I like.
* In order to create an incentive for users to invest in items that already have a large investment in them, bonuses will be granted to all investors when an item reaches certain thresholds of investment size.
* The most effective strategy for accumulating currency (and thereby influence), should be to invest in the items that are the best. Investing in an item early that later becomes popular is profitable because every subsequent investment gives money to initial investor and the first few will give most of investment to initial investor. Investing in already very popular items is profitable because every threshold reached creates profit for every investor. Users can invest multiple times in an item with no limits on quantity to help boost it over the threshold. However, the bonus will only be given with approval from admin. This it to check for attempts at cheating. Admin will have total discretion to decide whether the item is legitimately worth the amount invested in it.
* Users can also gain currency by donating real money to an item. The donation does not generate any currency, but it creates an investment in the item which can yield profits from subsequent investors. The purchased investment (with a donation) does not give any currency to item owner or previous investors. This is so that people cannot create currency for themselves with lots of real money; they must still earn the currency by making good investment choices.

# Thu 23-Jul-09

* To stop spam: make limit of 10 new items per IP address per hour, 30 max per day.
* Make tree viewer for admin that has root at selected item (because there can be multiple was to go up level from an item). It is possible for infinite loops to occur, but these are potentially allowable in a tree viewer as long as not all data loaded in one go, but sublevels are only loaded as they are opened.
* To prevent items being separated from connection with the root item, make initial upstream link (on item creation) permanent.
* Users gain levels (0 with no funds, 1 with >0). Level 2 and above must be granted by admin (or user with privileges) after user reaches a threshold worth (measured by quantity of invested currency). Gaining higher levels will require higher scrutiny by admin of specific contributions made and history of user’s actions on the site. This is because higher User level grants greater privileges for assisting in the management of the site.
* Users with higher level have greater privileges such as:
  + Adding/changing item directories
  + Making items into nodes
  + Granting new item levels (with associated bonus currency for all investors)
  + Granting new levels to lower level users than themselves
  + Granting requests for item changes (I may decide to make items unchangeable after creation to prevent users getting high ranks for items and then changing them to spam.
* Items gain levels when they reach certain worth threshold (defined by amount invested in them). Level gain must be granted by admin or privileged user. When an item gains a level, every investor gains bonus currency.
* Record every event that occurs on website in a table such that events can be easily reversed. E.g. all actions performed by a particular user in a given timeframe can be reversed instantly. Store prior state and state after action. Action is only reversed if state has not been subsequently changed again.
* Items can be given labels like: Node (like a folder), Question (asking other users to give an answer in exchange for currency), Request for admin attention, Comment, Software (hosted directly from the item)
* Find a way to make the impact of links to other pages on that page’s GoogeRank be related to the item’s rank on my website.

# Mon 3-Aug-09

* Put a permanent banner at the top of the site that gives hints/suggestions/requests that are relevant to the active item/category and to other known information about the user.
* Allow users to sort by date (both directions), Author worth, Item worth, relevance (which takes user specific information into account), and a weighted combination of these 3 valuation methods.
* Allow all users (and even non users) to rate items on level of advancement/amount of expertise required to understand the content. Show a simple average rating of all votes because there isn’t really an incentive to misuse this feature and it doesn’t count for much.

# Mon 10-Aug-09

* Items and Users gain levels at specific increments of accumulated currency: total currency spent by User/total currency invested into Item. The progress bar that shows on every item changes colour with each level.

|  |  |  |
| --- | --- | --- |
| **Level** | **Currency required to reach level** | **Progress bar colour** |
| 1 | >0 | Blue |
| 2 | 200 | Green |
| 3 | 500 | Yellow/gold |
| 4 | 1500 | Orange/bronze |
| 5 | 10,000 | Red |
| 6 | 100,000 | Purple |
| 7 | 1 million | Black |
| 8 | 1 billion | Badge replaces progress bar |

# Tue 11-Aug-09

* Progress bar shows proportion of total points to level. I.e. if user has $300 then they have 60% of $500 needed to reach level 3. In rare cases a user may be given permission to reach a greater level than indicated by their points , so it will sometimes be possible to be at less than 40% of level 3, but not usually. If a user reaches threshold point, bar stays full until admin (or a higher level user) grants them permission to level. All the above applies to items as well. In fact it is items for which the process of giving permission to level is more important, because whenever an item levels, currency is injected into the system.
* Store as many options as possible in Settings table under a “default” username, but only allow a small amount to be set individually by users. That way, if people ask for particular options, they can be granted. The purpose of this is to avoid flooding users with options, and to make it look like I’m very efficient at granting people feature requests.
* Currency required to level should go up by factor 10 each time:

|  |  |  |
| --- | --- | --- |
| **Level** | **Currency required to reach level** | **Progress bar colour** |
| 1 | >0 | Blue |
| 2 | 100 | Green |
| 3 | 1000 | Yellow/gold |
| 4 | 10,000 | Orange/bronze |
| 5 | 100,000 | Red |
| 6 | 1 million | Purple |
| 7 | 10 million | Black |
| 8 | 1 billion | Badge replaces progress bar |

* Use % units for font size.
* Move top navigation links to just right of logo. Use top right corner for:
  + Logged in: funds, level, account link
  + Not logged in: log in form, register link
* Anti-spam precautions:
  + Limit no. Registrations per IP per day
  + Limit no posts per IP per hour
  + When a user is reported as spamming, it is verified manually if the allegation is true and then the IP address is banned from sending data to the site.
  + If an IP address is banned for a proxy situation where it causes many people to be banned when one person at the IP was spamming – then those affected must manually contact site admin and convince an admin that the spamming will stop if the IP is allowed again.
  + Avoid using a CAPTCHA because there are moving towards becoming obsolete and are a pain in the arse.

# Fri 14-Aug-09

* I have decided to use an xml database instead of a relational one like MySQL. This will allow me more flexibility with structure and make it possible to put flexible self-describing data in items so that searches can look for data in particular contexts. E.g. mark-up a list as a list of good things with standard tags so that search requests can target phrases in that particular context: a list of good things.
* Use the standard multiple page system for each item with next page buttons and numbers to show all the pages. Try to avoid letting any item acquire too many direct sub items though. By default, send data in XML format and let the client transform it into html with xslt. Allow an option for sending data in html though, because it may be better for some devices. If users want huge number of items per page (make that an option as well), then html will load better because browsers can load it on the fly, whereas xml has to be fully downloaded before it can be displayed.
* I’ve changed my mind (again) about what form I send data in. I think it would make most sense to stick to what is most normal and send straight html. I tried making tags as short as possible in the xml and managed to get a page of 22 items from 29KB to 25KB. I also discovered that for html of the same data, deleting the formatting spacing to indent lines changes to size from 52KB to 34KB.
* Find a means to compress data while it is sent to client and decompress on client (I think browsers have a functionality to do this, but it is apparently never used).

# Wed 2-Sep-09

* URLs all must start with an item code. Every place in the website is characterised by an item and can be referenced by the URL: <domain>/<item code> plus some optional extra info.
* A URL should always look like <domain>/<item code>/<item name>... if the <item name> is missing or wrong, the user should be redirected to the correct name for that item code.
* Every link to an internal item should be automatically updated to reference the correct <item name>. This allows the user to change the item name if they wish and all links will be updated. Since the <item name> part of the url doesn’t actually matter, but is always redirected to the correct url for that <item code> no further work is required to redirect old links from external sites.
* Support for other languages than English should be implemented so that any item can be written in any language, but should be categorized accordingly.
* Every item contains an xml document. Data like “lists of good things”, “lists of bad things”, “drop down menus”, “subheadings” etc. Should all be marked up accordingly.

# Sat 19-Sep-09

* Provide tools to create various forms of marked up data, such as lists of good stuff, ratings of things (which appear as coloured bars in the Item detailed view) etc.
* No video, images, software or other data intensive media should be hosted. The focus of the site should be the information about information, rather than the information itself. Conspicuous suggestions should be placed in the site for sites that are designed for hosting media and users should encouraged to host their media on these sites and link to them from my site.
* There should be functionality for users to add custom mark-up options. The purpose of the mark-up is so that different types of data can be recognised by search engines. Advanced Search should allow users to search within data that appears within certain tags, like “list of good stuff” or “opinion” or “suggestion” or “rating” or “fact” or “reference” etc.
* Every kind of thing that is mentioned within the text should be allowed to able to be marked up, but it will be important to promote guidelines about how to use this functionality so it is used in a consistent way throughout the site. Things like “person’s name”, “book name”, “software name” should fall under this slightly different mark-up category of describing the language itself for the sake of computers.
* A system needs to be devised to allow users to extend the possible mark-up tags, while still maintaining the principle of tags being information about information (about information) and ensuring that tags do not repeat the role of the item hierarchy.
* On second thought, it may be ok to repeat the role of the hierarchy with tags.

# Tue 13-Oct-09

* Connect the body of each Item to a Wave (initially hosted by Google Wave, but I’ll eventually make my own Wave Server).
* I could ultimately integrate the Wave interface into my website, but the first step will be to allow access to Items a user owns through a Google Wave extension. When a user requests access to an Item through the Wave extension, they would have to enter the username and password of the owner of the item to get access to it. They could then share the wave with any conventional Wave users they wish. In this way, groups could collaborate on an Item under a collective username.
* The extension should provide a field under the Title of the Wave for editing the Item Summary. This field should be limited to very simple formatting if any at all.
* The Waves should eventually be hosted by my own Wave server, but only once I’m confident I can provide adequate resources to do as good or better job than Google.
* The website itself should provide basic Wave editing functionality, but not the sharing with other users part. The website editor should be adequate enough for those without Wave accounts to edit the basic text and formatting and insert pictures etc.
* Anyone who has a Wave account with any Wave Server can access their items and edit them more richly and collaboratively. I don’t think I’ll make a separate Wave interface even if I make my own Server. My Wave Server will be specifically for hosting Waves that represent Items in my website. User will have to choose a 3rd party Wave Server like Google Wave to use Wave’s rich features. The sharing will then go on between conventional Wave users and have no association with users of my website.
* I could make every Item an embedded Wave or use a Wave extension to edit the source code of each Item body dynamically. Making each Item an embedded Wave would require that every page view access the Wave Server. It may be risky to do it this way, because any failure or latency of the Wave Server would significantly slow the site or render items inaccessible. It would probably be better to build my own renderer of rich Wave content and have a Wave extension dynamically edit the source code.
* If I made my own Wave renderer in the site, I wouldn’t need to make my own Wave Server. That could make things easier. That way any 3rd party Wave Server could host a Wave and have it link to an Item on the website, but the Item itself is hosted by my website. This would be similar to embedding a Wave Server with basic functionality into my website, except that it wouldn’t do any of the communication part of the Wave Server, just the Wave editing and rendering.

# Thu 5-Nov-09

* Set up referral lines to commercial online products and give users virtual currency proportional to the amount my site earns when they buy a product through my site. As with the process of donating to get virtual currency, the currency is like an investment in the relevant item that only earns you new spendable virtual currency if others invest in that item after you or the item levels.
* If big companies try to effectively buy virtual currency directly from rich users on the site by paying them real money to invest in their products, this should be ok. It will provide yet another incentive for people to make virtual currency on the site because they can sell it directly for real money (unofficially). I don’t believe big companies will be able to influence the ratings too unfairly because a rich user will probably have gotten that way by investing in good products, and the most productive way for them to maintain their virtual wealth while making real money out of it would be to accept real payments only for commercial products they actually think are good. In this way, really good commercial products are likely to be slightly overvalued compared to their free competition, but bad commercial products are unlikely to get ahead of good free products (which definitely happens a lot in most advertising mediums I currently know about).

# Sat 7-Nov-09

I worked out that if you invest making total Item value up to , then when the item value is , you will have earned approximately . So every time the value of the item increases by a factor of , you will earn an amount equal to your investment. I then factored in the effect of taking 10% out of investments to go to the item owner (leaving 90% going to previous investors) and represented this as a value p=0.9. My revised formula was simply:

# Sat 21-Nov-09

**Possible Names for my website**:

* Athenova.com
  + Athena is the Greek god of wisdom
  + Nova is an astronomical explosion

**Purposes of my website**:

* A place to go to learn what you don’t know you don’t know
* Promote free information and create an incentive for people to provide information based resources for free.
* Provide an advertising medium for not-for-profit organisations. E.g. allow charitable projects to find interested people who are willing to donate
* Create an incentive for people to donate to free projects

# Fri 27-Nov-09

**Purposes of my website**:

* Bring a larger user-base together so that people can ask questions that will reach a larger and therefore more knowledgeable user-base. This effect will be further magnified by having a greater incentive to answer questions. People should be encouraged to invest in questions that they also want answered, as well as good answers to those questions.

# Sat 28-Nov-09

* In searching through items of a particular category, say tutorials for learning the basics of Ubuntu, to find out which item(s) is/are the best, people will learn a lot about that topic. The incentive to do this research about items in a category comes from the benefit of picking the best item to invest in, which is greater profit in virtual currency and therefore more influence in the website and potentially greater income in real money terms (from winnings or selling influence to businesses for real money).
* Items should have a “linked to” field which causes any items added to it to be added to the linked item as well. This allows automatic resolution of multiple representations of multi-tiered categories. For instance, if an item is categorized under Australia=>Politics, it should also be categorized under Politics=>Australia.

# Sun 29-Nov-09

**Purposes of my website**:

* New paradigm for commercialising information based resources which replaces the increasingly incompatible old model of direct trading discrete items for money.

**XML DB design**

* Items
  + ID
  + Title
  + Author
  + Worth
  + Date\_created
  + Date\_modified
  + Summary
  + Parents
  + Children
  + Document
  + Flags (information about Items that applies to them no matter where they exist in the hierarchy. Flags may also directly influence the functionality or behaviour of an item.)
    - **Link\_request** (link two items together)
    - **Change\_request** (request that a higher level user alters a particular item. Description of change requested is put in summary or document)
    - **Locked** (direct subitems cannot be created. At least one subitem must be unlocked which is dedicated to comments.)
    - **Spam\_alert** (request that an item be marked as spam, usually the parent of this item)
    - **Spam** (this item is spam)
    - **Suggestion** (suggest a way to improve the site)
    - **Question** (this item is a question for which an answer is requested, with the implication that a good answer will probably be rewarded at least by the question asker with an investment)
    - **Answer** (this is an answer to a particular question. The question will be one of this item’s Parents. New subitems of Questions will default to this flag.)
    - **Admin\_alert** (this item was created by Admin and contains important information that concerns the parent item.)
* User
  + Name
  + Avatar
  + Worth
  + Level
  + Funds
  + Items (list of owned items)
  + Settings (hierarchy of settings containing only altered settings)
* History
  + Event @type=create|edit
    - Timestamp
    - Address
    - Old (doesn’t exist for Create)
    - New
    - User
* Investments
  + Item
  + User
  + Amount
  + Timestamp
* Settings @id
  + Name
  + category
  + description
  + Options (could be list of options or type definition like Int or float)
  + Defaultvalue
* Hierarchy (constructed from Item child list recursively. Items can appear more than once)
  + <#id1>
    - <#id3>
    - <#id6>
  + <#id2>
    - <#id4>
    - <#id5>
      * <#id6>

# Thu 03-Dec-09

I realised that my current method of distributing currency has a major flaw in it. A person who controls multiple user accounts can invest in his own item when no one else has yet invested in it, and make all that currency back, which allows him to invest all of that again in the item. This can be done indefinitely, allowing a user to boost the value of an item infinitely.

I have thought of a means of partially solving this problem. A tax needs to be taken out of every investment of say 20%. This means that even if a user owns an item and no-one else has yet invested in it he can recuperate a maximum of 80% of his investment (instead of 100% without the tax).

So in the perfect scenario of making 80% back on your investment every time until you run out of currency, you can add 4 times your initial investment to the value of the item. If you repeat this process just 5 times instead of infinite, you increase the value of the item by 2.7 times your initial investment.

I will implement a delay (maybe in the order of an hour) on receiving currency earned from another’s investment in an item you either own or have an investment in. This will make the process of repeatedly investing what you earn back on your own investment much more tedious and reduce the extent to which users can multiply the influence of their currency. It will also give other users time to notice what’s going on and invest in the item themselves if they choose. Once this happens, the return from investment of an individual user will diminish from 80% and this too will reduce their ability to multiply the influence of their currency.

This solution is not ideal because it reduces the return on an investment by legitimate means by 20% also. If the value of an item increases by a factor of e (2.718) after you invest x currency in the item, then you will have only made back 0.7x instead of 0.9x.

Also, I would prefer that the system implements a paradigm that inherently does its purpose without need for workarounds to avoid specific vulnerabilities to “rigging the system”.

# Sat 05-Dec-09

* How to track all currency but still allow arbitrary division of currency?
* Detecting a user trying to multiply his currency to increase the value of his item would not be hard. It would be ineffective to try to mix up the money trail by transferring money to lots of other users before investing in the target item again, because of the tax on all investments (only means of money transfer). Therefore any attempt to inflate the value of one's item would show up as repeated investments in that item by a small no. of users. This situation should be automatically flagged to alert high level users not to level it unless it really deserves it - in which case I don't mind that the owner makes extra profit.
* It would be possible for a high level user to level such an item to help his friend or his own alternate user. This should be discouraged. Hopefully a culture can be created where people will write critical comments under users they discover doing this. This will worsen their reputation and make it harder for them to level. Similarly, users who grant level to disreputable users should have critical comments written about them which make it harder for them to level.
* I want to avoid direct punishment because it could be abused and even if only done by me, much of the behaviour that I would want to punish cannot be proved and/or is open to interpretation.
* Don’t call things spam. Any advert should simply be categorized appropriately. If advertisers try to abuse the site by flooding, they should be encouraged to use the site as intended because categorizing appropriately is likely to reach the target audience better. Users that continue to "flood" should have all their items marked to indicate as such. The company being advertised should be criticised and marked as a "flooder"- don't bother much with the user.
* No items should ever be deleted. Flooded items should all be put under the first item that said what they say, removed from other categories, and locked.
* Allow checkbox selecting of items (inc. search results) and mass actions like adding/removing/setting parents/flags. Mass Edit functionality should be selected as a page of an item (to allow selecting of its children). Search results should appear as the children of a temporary item with similar Mass Edit functionality.
* Prevent users investing in their own items. This is to stop them using the multiplier effect to boost their own worth (measured by qty invested). To use this effect it would be necessary to split the generated worth across at least 2 users i.e. by investing in each other's items. This would lead to an acceptably small boost to worth of less than 2 times actual currency used (i.e. initial investment). Since the use of earned currency in this way has an opportunity cost of investing it legitimately, which could produce excess cash exceeding that invested, it is hardly worth the trouble to invest in yourself. Unless you can get a higher level user to level your item. These methods should reduce the amount of profit made to acceptable levels if the item is granted a level up. As long as there is enough of a blow to reputation from doing this, it would not really be worth it.
* Possible names: Angor

# Tue 08-Dec-09

* Add functionality to my Search Engine to search within the results of a Google Query. This will require a Google API that allows me to send a Google Search string and receive the results as an ordered list of URLs. I can then parse the urls (all of which will be within my site) for the item id, and use that list as an additional filter for the search. The page rank returned by Google can be weighted and combined with other weighted sort mechanisms like item value, to produce the final result order.

# Wed 09-Dec-09

* Some flags (like locked) require a "level" to be specified such that only a user that level or higher can override the effects of the flag. Highest level flag is admin (maybe call this level the name of the site, which I want to work as the name of a god-like being). Level can be specified as being the same as the level of the item, so that the flag level increases with the item level. Syntax: add an optional “level” attribute to <flag> element.
  + Edit: “admin” should be the name of the highest permissions level, and it should be reserved for flags that will never be edited. The user with the same name as the site should be given the level “admin”, which provides it with infinite funds and all privileges. I want to create another user which I will use as often as possible, which starts as a normal level 0 user. I will use this user to add all my contributions to the site. The admin user must be used to create Help and Documentation type items though.
* Add functionality to set server status to maintenance. This will allow any existing operations to finish and block any new requests from anyone except admin.
* Make verify\_hierarchy() function which checks that all parents are matched to children and the hierarchy is consistent with the parents of each item. Parents are considered correct and the hierarchy and children elements are checked against parents.
* History: each user request that causes changes is a transaction. A transaction contains events like create\_item, add\_parent, modify\_document. Modify events store old and new value; can be inverted. add\_parent has inverse remove\_parent. create item inverse is several events: add\_parent(trash), add\_flag(locked).
* Design principles: no data is ever deleted; all actions are reversible.
* Store whole items at each revision in history container. Maybe users too. Consider storing most user info in linked item.
* User’s item automatically draws specific info from user data. I can't think what info that would be. User item has subitems: Profile (editable by owner only), user History, Comments.
  + Edit: user item is owned by user and allows the display of a profile. Provide a template profile on user creation. Item is locked/barren (can't have more children) at Level admin, with subitems Comments, User History, User’s Items, User’s Investments.
* “User’s Items” item should simply be a parent of all the items made by that user. Whenever an item is created, the author’s item should be added as a parent. An optional attribute for the <parent> element needs to be available to make it permanent. Syntax: @locked = $level, where $level is the minimum level required to override the lock. In the case of the author’s item being the parent of every item that user creates, @locked = “admin”. This effectively means that this parent is permanent, possibly even for “deleted” items. Since deleted simply means sent to trash, this is fine from a design perspective. I think items will mostly be sent to trash if they are an abuse of some rule, if they are flood items, or if they just contain gobbledygook or something. If a user has created any such item, it should be visible to all, so the item should remain in that user’s “User’s Items” item.
* “User History” item should also contain subitems, one for each action the user has taken in the site. Users can than add comments under actions to approve or disapprove of them, and actions can be invested in as a way of approving of them.
  + It would be useful if an action that was generally disapproved of showed up obviously so that people glancing over a user’s history can easily tell if they have done anything inappropriate or suspicious. Since negative voting does not fit the investment based voting mechanism of the site, this is not an option. However, items that are marked as criticisms and receive a lot of investment should show up in the parent item as a negative gauge of opinion.
  + Two additional gauges should be added to items, which are sort of one degree of separation from direct investment in the item. One gauge measures the total investment in comments marked +ve and the other gauge measures total investment in comments marked –ve. If an item is marked “barren” (meaning it can’t have more children), then comments are taken from the “comments” subitem. Otherwise, any direct subitems marked as +ve or –ve comments contribute to these two extra gauges.
  + This system maybe dangerous since an item’s parents can be changed, so an item marked as a –ve (or +ve) comment that has a large worth could have additional parents added to it. These parents would then have their 2nd degree gauge’s affected by the new subitem, even though the comment might not be about them. To avoid this problem, a comment should have the target of the comment stored in its metadata.
  + I’m starting to dislike this whole idea because it’s getting messy. To allow efficient calculation of the 2nd degree gauges of value in each item, they would have to store a list of +ve & -ve comments. The actual value of this 2nd degree gauge would have to be calculated on every item request and this would significantly slow load time, especially if the gauge is to appear in all listed subitems as would be desired.
  + I still want an effective but simple way to highlight user actions that should be taken into account when a higher level user is deciding whether to grant them a level or not.
  + Possible resolution: every comment under a User action should also automatically appear under the User’s “Comments” item. +ve and –ve comments should have some visual indication of their bias. It is now easy to get a crude but quick overview of a user’s reputation by seeing the distribution of highly ranked +ve and –ve comments that will appear at the top of the comment list.
* “User’s Investments” item should contain subitems; one for each investment the User has made. In the same was as User History, this allows comments to be made about the investments, praising or criticising them.
* When a user creates an item under an official “comments” item (which will always be a subitem of a “barren” item, it will automatically be given the metadata: “comment”, with the parent of the “comments” item being the target. The user should be asked whether the comment is +ve or –ve or neither when they first create the item.
* Creating a user only requires username & password. Any private user info like payment details is stored in Settings. Hence there is no need for a User management page. Settings should be an item under Home with each setting an ancestor item of the Settings item. The unique thing about settings items is that they will contain forms that pull data from and send data to the user's private settings.
* New design principle: every page in the site is in the item hierarchy. I.e. all urls start with an item id; even resources like images.
* For Document field, allow all html except forms & javascript. Also ban tags: <html>, <header>, <body>. Styles can be inline styles or embedded (or whatever you call it when styles are defined in the html document instead of in separate css file).
* All site functionality should be available with javascript turned off. This would require defining different implementations of some functionality if javascript is off.
* if javascript is off, document editor should simply be a text box that allows entering html as text. The html will be verified when it is sent to server. This method is not very accessible to people not familiar with html, but since it will be a minority of people who have no javascript this is acceptable. Suggestions can be given for generating html with another application.
  + Editing other fields like metadata may also require some innovation without javascript. Metadata nodes should be chosen from a dropdown list and the type of data given to a particular metadata node can be different depending on the metadata node chosen, which causes a dilemma for static non-javascript forms.
  + One possibility for metadata is to make it a textarea field and expect the user to work out the correct syntax (which must of course go through an advanced validation system). This could be realistic. Tutorials for editing metadata directly as xml text could be linked to.
  + I think the plain block of text approach is the only way to go for non-javascript access. As long as proper validation is used, this shouldn’t create a security issue. Advanced users may want to use this method of data entry even if they have javascript.
* For those with javascript, provide a simple html based rich text editor similar to Google Docs. Provide option to show (and allow editing of) the raw html being produced by the editor.
* Make an item under Home called Featured Items, which is just a list of my favourite items.
* Add a link to every item called “Top 100”, which simply links to an empty search with parameters in the url to show the first 100 items on one page, overriding the user’s “items per page” setting. This means that for items with less than 100 ancestors, less than 100 items will be displayed, and for items with more than 100 ancestors, a next page button will be visible.
* Parameters like default sort order, items per page etc. can be defined in url. If not specified, the default (given by the relevant setting) is used.
* Allow custom sort specification to be defined in url by specifying weightings for various valuation factors like Google rank, worth, author worth, user specific relevance (an advanced and difficult to implement feature that I probably won’t add early on).
* It might be realistic to abandon the Settings, History and Investments Containers altogether and simply use items to represent Settings, Events, and Investments with metadata specifying data that needs to be read programmatically.
* Item <flags> element options:
  + Locked (@level,@owner\_level): prevents editing of <title>, <summary>,<document>,<metadata> fields. By default @level = $item\_level + 2, @owner\_level = 0.
  + Barren(@level,@owner\_level): prevents adding children to this item.
  + @level=$level, means a user must be at least level $level to override effects of flag. This doesn’t apply to item owner
  + @owner\_level=$level, means the item owner must be at least level $level to override the effects of the flag. The value of $level will usually be one either 0, “admin”, or the same as @level.
* Bullying of specific users is very possible because of the ability of high level users to edit lower level items. One disincentive to bullying is that it will appear as bad reputation which will make it harder to level. The problem with this is that a user can get to a relatively high level by acting honestly, and suddenly turn into a bully. It may be difficult for that user to gain any more levels, but he is already a high enough level to continue bullying lower level users and even high level users because he can edit their low level items.
  + It will therefore be necessary for direct punishment to be applied to users, such as losing levels.
  + A user must be two levels higher than another user to grant them a level. They must be 3 (or maybe even 4) levels higher to take away a level. This should be done very sparingly.
  + The privileges that come with having a high level should be kept somewhat separate from a user’s worth. If a user is very good at playing the site and contributes a lot of value (hopefully these two traits match up pretty well), then they should not be prevented from continuing to contribute their ideas, even if they turn out to be a bully and their privileges must be reduced.
  + Privileges may have to be reduced in the worst case of a user accidently creating destructive edits. This situation is likely to be quite subjective and reducing a user’s level should be a last resort to give them the message that they need to listen to suggestions to improve the way they make edits. Obviously the existence of suggestions to change their editing behaviour is a prerequisite for going to the extreme length of reducing their level.
  + The disadvantages of having a reduced level as a user (lower than worth should allow), are:
    - Less power to edit other items (and in rare cases their own items).
    - When users sort items by author worth, their items will appear lower because Level is a more powerful weighting factor than worth.
  + It may be necessary in the worst case scenario to actually ban a user from access to the site. I can’t think of a good reason to do this and I want to keep it that way, because I believe a positive approach will help reduce wrongdoing by contributing to a positively oriented culture in the site.
  + Offences such as denial of service attacks, flooding etc. need to be dealt with by blocking IP addresses because any number of level 0 users can be created. I might disable the level 0 accounts as well in these cases, but this would not be that important.
  + Vandalism offences and bullying can be dealt with by reducing level and don’t require banning account.
  + If a user is punished by having their level reduced, it may make sense to change the permissions on changing their level so that a high level user is required to undo the punishment and start granting levels again.
* If Settings, Investments, and Events are all stored in Items, with the details specified by metadata, then the only Containers required are Users, Items and Hierarchy.
* Settings will store the default values in <metadata>, description in the <summary> element, and a form for specifying custom user settings in the <document> element. This system also allows comments to be added under each setting which elaborate on or critique the functionality of that setting.
* The History container may still be necessary to store copies of each revision of an item.
* Another alternative for storing revisions is to copy an item to a new item and then edit the original. This seems more consistent with the design pattern of other functionality.
  + It may be misleading that the ID of the historical revisions would be different from what they had been when they were the current item.
  + When an item is edited, the old item is copied to a locked historical item with a new ID, then the original item is changed.
  + How can all the information stored in an item be captured in a new item which is located in a different place and has different permissions and flags? I.e. if the <parents> element of the revision is to be captured in the item that stores this revision, then it can’t be stored in the new item’s <parents> element, because that must be changed so that the historical revision items only appear under the History subitem of the original item.
  + Another problem: this would entail accessing an Item’s History by finding a subitem named “History”, but this would be messy at best, and impractical for an item with many subitems. The reason this system works with User Investments and such is because the User’s item is barren and has a small number of direct subitems.
  + I think it will be necessary to have a History container that simply stores entire items (with the original ID), as revisions. Since revisions do not need to be able to allow subitems to be created, this system is fine.
* Another question is where Investment items should go.
  + I like the idea of making Settings and Actions into standard items, because it allows comments to be added as subitems to them, and Investments in them.
  + It may not be practical to make Investments into items because it would create a recursive issue in which investments have investments as subitems. There are various other issues with it, and I don’t think the concept really makes much sense. It is not really necessary (or sensible) to be able to Invest in an Investment.
  + The question remains: how to provide a list of Investments made by a user, as well as a list of Investments in a particular item, in a simple and cohesive manner.
  + Investments in an item can appear under the Item subpage “Investments”. That was the system I had already planned.
* The History subpage should show a list of items almost exactly the same as normal items except without “add subitem” option. Each revision item should have an additional header stating info about the revision. Info includes revision number, editor, editor level at the time of edit, timestamp of edit (also indicated by the date\_modified value of each revision), fields changed.
  + The first revision should state “**created** by <user> at time <timestamp>”
  + All subsequent revision should state “**modified** by <user> at time <timestamp>”
  + Last revision is the current item.
  + Each revision can be opened to show the full document.
  + The address of a specific revision is <domain>/<item id>/History/<revision number>
  + Item data like <parents>, <flags> are kept in each revision, but have no effect. Historical revisions only appear in this context: under the History subpage of the Item.
* URL format:
  + All URLs start with an Item ID
  + Site resources are found at <domain>/0/resources/
  + User resources like avatar images are found at <domain>/<user’s item id>/resources/
  + Item resources are found at <domain>/<item id>/resources/
  + Item subpages are found at <domain>/<item id>/<subpage>
  + Subitems of subpages, like historical revisions are found at e.g. <domain>/<item id>/History/<revision number>
* Extension of the <flags> concept: replace <flags> element with <permissions>, @type, <metadata>
  + <permissions> element defines users’ ability to edit each element of an item
    - Format: <children @level=”$level” @level\_owner=”$level” @level\_edit=”$level” />
      * @level defines permissions for all users except item owner
      * @level\_owner defines permissions for the owner of the item
      * @level\_edit defines level required to edit these permissions. If this value is not “admin” it will probably be defined with reference to the variable $item\_level. Editing this attribute requires admin permissions.
      * $level = “$item + 2” means only users at least 2 levels higher than the item level can edit this field. This variable-in-an-attribute functionality may require something like an “eval()” function to implement. Security needs to be taken into account here.
      * $level = “admin” means only an admin level user can edit this field
      * $level = “3” means only level 3 users or higher can edit this field
    - Other permission targets (children is used in the format example):
      * Parents (individual parents may also require specific permission to edit. This is specified by @level, @level\_owner (& maybe @level\_edit) attributes of the specific <parent> element.
      * Title, Summary, Document (might be put under a single permissions target).
      * Metadata (further permissions may apply to children elements of <metadata> but these will be specified as attributes of those children in the same was as individual <parent> elements described above.)
      * Level (ability to grant this item a levelup.)
      * level\_premature (ability to grant a level to an item that does not meet the target worth for that level.)
  + @type attribute of the <item> element gives a categorization from a list of mutually exclusive options. This categorization is deeply tied up with the functionality of the item.
    - **Comments**: the “comments” subitem that goes under an item with <children> permission level greater than 0.
    - **user\_item**: the item that represents a particular user. I may give this item unique features to allow access to the User’s Investment and Action history.
    - **Action**: an item which represents a user action
    - **Setting**: an item that provides an interface to edit user specific settings. Settings items allow html forms in the <Document> element.
    - **Normal**: all other items
  + <metadata> element contains dynamic data that can be added to by users. Metadata is used for example in <Action> items to specify such data as the Xpath address of the edit, the old and the new value. This data can be used to reverse the actions. There is a specific set of metadata tags available to Normal items (and a potentially different list available to other item types). Users can request additions to this list. This is similar to the system that will be used for html markup where only specific @class attributes are allowed, but the list can be increased by user requests. Examples:
    - <link\_request /> request that a link be created between two items. Note: I am uncertain whether I will implement the “link” concept because it seems messy. The primary situation it caters for is categories that would be named differently depending on the parent category. I would rather items be named such that they can be placed under all the appropriate parents without requiring different names depending on the parent category.
    - <edit\_request> request that a higher level user alters a particular item. Description of change requested is put in summary or document. This metadata tag could also be extended to allow a fuller description of the change requested, in the metadata.
    - <suggestion> suggest a way to improve the site or an item.
    - <question> this item is a question for which an answer is requested, with the implication that a good answer will probably be rewarded by the question asker, with an investment in the good answer.
    - <answer @question=”$id”> this is an answer to a particular question. The question will be one of this item’s Parents. New subitems of Questions will default to this flag.)
    - <admin\_alert /> this item was created by Admin and contains important information that concerns the parent item.
    - <acknowledgements> can contain multiple <acknowledgement> children which contain references to websites, people, books etc. from which some of the information in this item was drawn.
* Add [(?)](file:///D:\My%20Dropbox\Projects\Zutha\nowhere) links to various places in the site, which link to a relevant help item. Such links should open in a new tab because they will often appear on pages containing forms and the form data could be lost of the user opens a link in the normal way. Examples:
  + When creating a new subitem, two parents will be added by default and editing these parents will be disabled. The two parents will be the item from which the “create subitem” button was clicked, and the “User’s Items” subitem of the User’s Item. Add [(?)](file:///G:\Projects\InfoMarket\nowhere) link next to the disabled parent fields which links to an explanation for why the field is disabled.
* Put a list of links to useful searches somewhere on the site. Examples:
  + Find all comments about any of your items that were created within the last x days.
  + Find all unread comments about any of your items
* Make functionality to add useful searches for easy access within the site. This can be done with bookmarks, but it could be useful to have access to useful searches from anywhere and this can be done by storing the searches on the server.
  + This list should initially (at user create time) contain a default set of useful searches which can be edited by the user.
  + Help about this function should link to an item designated to Useful Searches, with community contributions etc.
* Add a read/unread flag to items. Since this is specific to each user, it may make most sense to store this information as a list of “read” items under the User document.
* Necessary Containers: Items, Users, History, Hierarchy, Statistics
* Investments should be as Action Items which can be listed under User’s Actions. The Investments subpage of an Item should generate a summary of Investments from the relevant Investment Action items.
  + Information stored in an Investment Action Item:
    - a human friendly summary of the information like “barneyii invested $1000 in the item 4REW – ‘Photovoltaic Technologies’ at time 2010-06-11 9:26:14 PM”
    - the worth of the item at the time of investment
    - the level of the item at the time of investment
  + The new design principle is that information about an Item should be organised under subpages of that item, whereas information about a User should be stored as items which can be organised under a finite set of subitems of the User’s Item.
  + The Investments subpage of an Item needs to draw add\_level Actions specific to that item as well, because levelups should be shown in the Item’s Investment history table.
  + Investments subpage representation should show a cut down version of Investment items in an unalterable timeline order, with levelup events included in the timeline. Each Investment or levelup event should be accompanied by a link to the actual Event/Action Item.
* Types of Action items:
  + Create user
  + Create item
  + Edit item (this is the container/transaction item. It contains subitems defining the specific fields edited)
    - Edit title
    - Edit summary
    - Edit document
    - Add parent
    - Remove parent
    - Edit parent permissions (change the editing permissions on a single parent)
    - Add metadata (add a metadata child)
    - Remove metadata (remove a metadata child)
    - Edit metadata (edit the value of a metadata child)
    - Edit metadata permissions (change the editing permissions on a single metadata child)
    - Edit permission (change editing permissions for a specific field – or field group in the case of title, summary, document)
  + Add\_level
  + Take\_level
  + Investment
* All Action Items should be owned by the User that did the Action, but all Action Items are locked for editing by anyone except admin (which means they are never edited)
* The names of Action items can be an extended, human friendly description of the action because the name does not need to be considered when the item is processed by the computer (e.g. for reversing an action).
* When editing <metadata> or <parents> in no-javascript mode, subitems that you do not have permission to edit will be shown outside the textarea field. Trying to put these locked subitems in the textarea field will fail validation.
  + To remove a <parent> or <metadata> child in no-javascript mode, the xml element just needs to be remove from the raw text. This might involved removing the text “<parent>TR5Q</parent>” from a sequence of such <parent> elements in xml form.
* Granting (or taking away) a level should be done on the User’s Item.
* I’ve changed my mind again about the content of the User’s Item.
  + It should be locked to edits by anyone except admin, but be owned by the User.
  + It should contain simple information (not personal information) about the User.
  + It should include buttons to grant or take a level if those options are relevant and available to the viewing user.
  + It should contain info regarding the suitability of this User to be levelled. This is probably not going to be used for most Users. It will only be put on the User’s Item page of badly misbehaving Users as an alert to higher level User who are considering granting them a level.
  + It should contain links to all its subitems (these links are superfluous of course – they are mainly just to give this item some more content)
  + A Profile subitem should be available to editing only by the User it applies to.

# Thu 10-Dec-09

* create\_item event has subitems specifying modification of each field (with old value being blank)
* For every Action that can be done in the site, first create the action Items, then “run” the action items.
  + This design principle may allow admin to create and “run” advanced Actions with xml in a more flexible way than can be created with the site interface.
  + Actions should also be designed to be able to be run multiple times, though this should never happen in the normal operation of the site.
  + Create\_item actions need to be generic, so not store a particular item ID, but every time they are run they’d have to store the ID of the item the action created in order to maintain reversibility.
  + I’m starting to dislike this idea. The original purpose of actions was to record a detailed history of all actions taken in the site, and to potentially allow reversal of actions in the case of an extensive vandalism attack.
  + I think I will stick to the principle that Action/Event items are only created when the Action is requested, and they are immediately performed. If an action needs to be reversed, a reverse\_action() function should create an opposite Action item, and then run that.
* Event items obviously do not generate event items on creation (or an infinite loop would occur)
* Investment Event has Payment subevents, specifying every individual payment to a User (or the treasury in the case of a tax), with payment source being the investor.
* Granting a level Event also has Payment subevents, with Treasury being Payment source. "Action doer"/Event owner should be the User that granted the LevelUp.
* In statistics, record flow of currency in and out of treasury to keep track of total currency in the economy.
* Example Scenarios to explore the flexibility required in the Permissions system
  + A new item is created by a level 4 user. He wants to make it a category parent item, which means it has a finite set of sub-categories under which items can be added, but new items should not be carelessly added directly under this category parent item. This requires setting permissions on children to prevent low level users from adding children. The item creator should be able to set permission to a maximum of level 4 (his own level). Let’s say the owner sets @level=”4”. This means other users must be at least level 4 to add children to this item. A level 6 user might decide the item in question is causing too much dispute and that the decision making about its children needs to go to a higher authority. He sets @level=”6”. It could make sense here that the item owner also, is banned from editing children because he is less than level 6. Maybe it is not necessary to give an item owner inherently more privileges over editing his own item than other users.
    - By default, <children> permission should be missing, which is the same as both attributes being set to level 0. This means anyone can make a “category parent” item, but they can only prevent users of a lower level than themselves from editing children.
  + An item should require a user 2 levels above the item level to grant it a level. A user should not be able to level his own item. A level cannot be taken away from an item. I cannot think of a situation where a single item should need different permissions than all the others in terms of levelling. This could allow it to be a globally set permission. <level> would then not be a subitem of <permissions> and level wouldn’t be managed with the permissions system. The level\_premature permissions target does not apply to items because an item should never be able to be levelled prematurely. A user can be levelled prematurely, however.
  + Levelling users requires more flexibility with permissions. A user must be at least two levels above another user to grant him a level. Levelling a user prematurely should require a higher level (maybe 3 or 4 levels above). Taking a level away should also require a higher level (perhaps 3 or 4 levels above also). If a level is taken away from a user, it should be more difficult for that user to level than another user at that level.
  + By default, low level users should not be able to edit other people’s items because low level user cannot be policed - you can easily create large numbers of them, especially level 0 users. All fields except <children> should by default require say, level 3, to edit. This means that users below level 3 cannot edit their own items after creation, by default. This could be a useful principle, because it prevents unaccountable (low level) users from creating a good item that gets popular, and then changing the content into an advertisement for something completely different.
* Instead of having a @level\_edit attribute, it might work to simply require a user to be at least level $level in order to edit an attribute @level=$level.
  + An owner of an item cannot edit the @level\_owner attribute.
  + For a particular field, an owner must be at least the maximum of @level and @level\_owner to edit that field or the @level attribute.
  + Another user must be at least level @level to edit either @level or the field it applies to.
  + Another user must be at least the maximum of @level and @level\_owner to edit the @level\_owner attribute.
  + Possible problem: If a user is a higher level than the owner (even if only level 1) than he can prevent only the owner from editing his own item.
  + I think the only reason I invented the @level\_owner was for the <level> field, because a user should not be able to level his own item too easily.
  + Maybe I can simplify this process by making it impossible for a user to ever level his own item. For all other fields, his permissions are specified by @level.
  + I think the owner of an item should have more say over editing it than other users, in general, even if the owner is a low level.
* <permissions> element could be replaced by @edit\_level attribute inside the relevant element. This is more coherently extendible to putting an @edit\_level attribute inside specific parents or metadata children.
  + <title @edit\_level=”3”>the title</title>
  + <parents @edit\_level=”3”>  
     <parent edit\_level=”admin”>RR4T</parent> # RR4T might be the owner’s “User’s Items” item for example.  
     <parent>YQ4</parent>  
    </parents>
  + If a subitem of <parents> or <metadata> doesn’t have a @edit\_level attribute, then it inherits it from the parent’s (<parents> or <metadata>) @edit\_level attribute.
  + The default for @edit\_level is 3 for all relevent elements except <children>, whose default is 0.
  + Elements with @edit\_level attribute:
    - Title
    - Summary
    - Document
    - Parents
    - Children
    - Metadata
* The <level> element under a User document could contain attributes @levelup\_gap=”2”; @premature\_levelup\_gap=”4”; @leveldown\_gap=”4”, where 2, 4 and 4 are the defaults respectively. To grant a user a level, you must be @level\_gap levels higher. To grant a user a level prematurely, you must be @premature\_levelup\_gap levels higher. To take away a level from a user, you must be @leveldown\_gap levels higher. These values could be changed in rare cases. E.g. @levelup\_gap could be increased after a user is demoted for misbehaving, so that they can’t easily be promoted again.
  + Each of these attributes described the level gap required to change that attribute as well.
  + The maximum value a user can set one of these attributes to, is the level gap between themselves and the user the attribute applies to.
* There should not be a relative association between item level and user level. I have abandoned the idea of putting expressions like “$item\_level + 2” in the @level attribute of item editing permissions.
  + The concept of item level and user level is distinct. An item level represents the perceived value of that item. It should not **necessarily** represent the sensitivity to editing abuse or the rights required to edit it, but it might in some cases.
  + Rights required to level users should be relative to the user level. I have decided to use a relative (i.e. levelup\_gap) system for permissions on changing user level.
* The @level = “$item\_level + 2” concept may not be required. It doesn’t really make sense to limit editing based on the worth of an item because that doesn’t necessarily define how sensitive it is to editing abuse.
* When a user tries to add a subitem to an item with limited permissions on <children>, a dialogue should appear asking if they want to add their new item under the “comments” subitem, and explaining briefly why this necessary. If the user requesting a new item has high enough permissions, an option should be given to add a new direct subitem, but with a caution that this should only be done if the user understands why adding children is locked and what sort of new direct subitems would be acceptable.
* It is important that levelling is seen as a privilege related to responsible editing contributions and not an intrinsic right to those with the prerequisite worth. The communal based editing system relies on the users with the high levels and greater editing privileges having a responsible approach to editing and an understanding of the etiquettes relevant to each aspect of editing. This doesn’t mean that disputes between high level users are a big problem, or that different etiquette can’t be developed with the input of thoughtful users.
* Levels 1 and 2 should be granted quite easily. To reach level 3 and beyond, a user should be able to show that they have contributed constructively to the site by doing such things as better categorizing items, adding appropriate parents to items, renaming items slightly so that they can be categorized appropriately in my relevant places.
* Allow resources from any item to be linked to any other item. However, prevent resources in the site from being accessed by external domains.
* When a user picks their avatar image, the image must be below a certain size (in kb). If it is not, the user is prompted to let a new compressed image be automatically created.
* Item subpages:
  + Subitems
    - Shows the item Heading, then summary, followed by subitems. This view makes subitems appear at the top of the page event for items with a lot of content in the <document> element.
  + Document
    - Shows the item Heading, then Summary, then Document, followed by subitems. This is the default page that appears when an item is first opened. “<domain>/<item id>” is the same as “<domain/<item id>/Document”. The purpose of the Document page’s apparently superfluous existence is that I might make it possible for a user to change the default page.
  + History
    - Shows a list of revisions with the current item being at the top.
    - Subitems have headers with a header title like “Revision 3” which contain who, what and when details about the edit.
    - A normally formatted item appears below the header (containing Title, Summary, level & worth info).
    - Another possibility is to have the list of revisions look completely different from the standard list of Items system. It could even look more like a table.
    - Actually I don’t like the idea of making revisions look at all reminiscent of normal items because they aren’t. Revisions should have a completely different format.
    - Each revision should have a simple layout with the following information
      * Revision # as a title and a link to the detailed specific-revision page
      * Editor (user who made the edit)
      * Fields edited. Each edited field listed should link to the relevant Action Item.
      * A link to the Action Item that represents the whole revision.
    - A revision can be opened as a subitem of the History page. This will show all the information in the item in a unique view that includes all the normal fields plus <metadata>, <parents>
    - The item and author level and worth should be saved in each revision, as should the level of the editor at the time of the edit.
    - Revisions can be selected for the “compare revisions” feature. This feature will probably involve advanced techniques and may not be implemented immediately. The purpose of this feature is to find the specific changes made to the Summary and Document fields between Revisions. I should research how this will be done so that I can design the site architecture to make its implementation easier, if possible.
  + Investments
    - Show the Item Title and ID at the top
    - Main content is a table contain a time ordered list of Investments made in the item.
    - Some rows could be levelups (these should be made very obvious).
    - Each row should contain investor, timestamp, amount
  + Metadata
    - Metadata could be arbitrarily complex xml. It could be difficult to display this in a reader-friendly way.
    - Each direct subitem of <metadata> needs to be clearly separated. The content each could simply be displayed as raw xml.
    - The most complex metadata will be found in Action Items. It is not necessary for this to be easily understandable by everyone. Indeed, metadata in general is only likely to be used by advanced users, so it doesn’t matter if it displayed as raw xml.
    - <metadata> subitems should be expandable/collapsible (with javascript)
    - In the case of storing the full html of a Document in, say, a modify\_document Action item, it is definitely going to be necessary to display that as raw xml (only valid xhtml will be allowed in Document of course).
    - The value of a metadata tag should be stored in a box containing a selectable string of text. This would mean a long string (as in the case of a large Document), may not all show in this box. It may have to scrolled, or copied and viewed in notepad or something.
    - Since all metadata comes from a fixed set of options, whose structure could be described in detail, it could be realistic to lay out the expected elements as expandable/collapsible sections (like the way xml appears in a browser). When a value is expected based on the model, and more xml appears, it should display as raw xml text.
  + Parents
    - Show a simple list of parents, Ids followed by Item Name as a link
  + Resources
    - Show a table of Resources attached to this item
    - a row might contain a filename (as a link to the resource), and some metadata like filesize, data\_added etc.
  + CreateSubItem
    - The form for creating items. Whatever item this is a subpage of will automatically be placed in the <parents> field of the new item (and probably locked there).
  + Edit
    - The form for editing this item
  + Search
    - Shows the standard search page with a long search bar at the top and search result items underneath.
  + AdvancedSearch
    - Shows the advanced search page with lots of combo boxes and text fields to define all the different search functionality
    - After pressing search, the normal Search page opens with the Advanced Search parsed into a single line of text.
    - This requires the development of an advanced syntax (more advanced than google’s) for creating complex searches with a single line of text.
    - The AdvancedSearch page does what Google’s advanced search page does: gives you a more intuitive interface for performing advanced searches. All the advanced search functionality is available in the normal Google Search bar though.
    - Form data from AdvancedSearch page is sent by Post. Server processes the data and turns it into a single string search. User is redirected to the Search subpage, with this auto-generated search string as a get variable. This string will appear in the search box. This is probably similar to the way Google implements Advanced Search.
* MassEdit function
  + I had considered allowing mass edit from the subitems page, but I think that will be too messy. Also, it would be better if this functionality only appeared in more advanced contexts.
  + MassEdit option is only available for search results page.
  + All search results pages will show checkboxes next to every result to allow selecting of that item.
  + Mass Edit actions available are adding/removing/setting, parents or metadata. At least 1 item has to have been checked for MassEdit to be available.
* All pages should be created initially in xml form. XSLT will be used either on the server or on the client to transform the xml into full html. Some information that is the same for all items may only reside in the XSLT file, such as forms, headings etc.
* Architecture should be designed so that additional Look and Feel sets can be defined by a collection of XSLT files and a CSS file.
* The recording of events should be complete enough that theoretically the whole history of the site could be played back.
  + It might even be useful to set up a read-only side server
* Investments sourced by Donations or Product Purchase instead of virtual currency create an Investment Action Item, but without any Payment Action Item subitems.
* There needs to be a notifications system to tell users they have levelled or gained more cash since they last logged in. This could be combined with a panel to show random tips to help people discover the features of the site.
  + Tips should be stored as Items. Each time a user sends a request to the server a new random tip should be pulled from the set of Tip Items.
* The architecture of the site should allow easy implementation of an API for 3rd parties to create their own front-ends to the site. E.g.
  + An iPhone/Android app that is more suited to that interface
  + Voice or even Brain Machine Interface controlled interfaces
  + Another website that draws xml data from my server and sends xml to it. This could implement significant interface alterations that go beyond the visual changes allowable by different sets of XSLT and CSS files.
* Make a function to extract all Action items from the database, rebuild the hierarchy (from parents only
* Problem: when a User is first created, there is no User Item to put the Action Items under, but an Action Item should be created first, and then run, to create the user.
* Clarification of the Action Item concept
  + Create\_user.
    - Create\_item (the user item)
      * ...
    - Create\_item (“User’s Items” item)
      * Add parent (the user item)
      * ...
    - Modify\_item (add the User Item as a parent to this create\_user Action Item.
      * Note: this is beginning to get rather messy and somewhat incoherent
  + Create\_item
  + Modify\_item
  + Investment
    - Withdrawal
    - Payment
  + User\_promotion
  + User\_demotion
  + Item\_promotion
* I think it would be a much cleaner system if Action Items didn’t have subitems.
  + One of the main purposes for this idea was to record every payment given to prior investors, under an Investment. This is so that Users can get statistics about where their income is coming from.
  + It doesn’t make sense to comment on (or add any other type of subitem to) subitems of Action items. It is the user’s action as a whole that merits comment, not the technical break up of what goes on to do that action.
  + All the technical details about how to do a particular action can be stored in the metadata of an Action item, including the many steps required in the create\_user action.
  + Each individual payment of an Investment could be stored in the metadata of the Investment Action Item. This isn’t conducive to collecting financial statistics for a particular user, though.
* When a user edits an Item’s Document or Summary field, he should be asked to summarise what he did. This info will be added to the modify\_item Action Item.
* I have decided I need to reincarnate the Investments Container as an expanded concept which I might call Transactions
  + Transactions container stores all cash flow events.
  + Transaction parts (for an Investment type transaction):
    - Owner Income
      * Cash received as the owner of an item when another user invests in that item
      * 10% of investment
      * 1 of these per transaction
    - Investor Income
      * Cash received when a user invests in an item you already have an investment in.
      * Many of these per transaction
    - Investment
      * Cash spent by the investor in the investment
      * 1 of these per transaction
    - Tax
      * Cash sent to Treasury when you make an investment
      * 20% of Investment
      * 1 of these per transaction
  + Transaction parts (for a levelup type transaction):
    - Investor income
      * Cash received when an item levels that you have an investment in
      * Many of these per transaction
    - Injection/Payment (I can’t think of a good name)
      * Payment (usually by the Treasury) equal to the current worth of the item, divided up proportionally among existing investors in the item
  + At any one time, the total cash in the system + balance in Treasury (which will be –ve) = 0
  + The debtor and creditor of every transaction is recorded
  + Timestamp obviously recorded also
  + All cash transfer occurs through investments. There is therefore a particular item associated with every transaction.
    - Edit: cash transfer also occurs on item levelup.
* The main reason I wanted this transactions system is to make it a more natural process to pull financial history data about a particular item.
  + I wanted to have a flat list of all cash transfers from which those concerning a particular user can be pulled and turned into a table.
  + I’ve realised that the different parts of a transaction really can’t be coherently separated into descrete “cash transfers”; they are inherently part of that transaction.
  + The format that makes most sense for storing data about a transaction is almost exactly what I would have put in the metadata of an Investment Action Item.
  + I’ve also realised that with proper indexing, pulling all the “Investor Income” parts of a transaction that apply to a particular user, would be easy with XQuery.
  + I think I am still thinking about query efficiency in terms of the relational database model.
  + If I’m going to arrange the “investor income” parts under a Transaction parent, it might as well use the existing Item model and maintain consistency.
* A User’s Budget Item will contain a table of transactions (showing the part of the transaction relevant to the user).
  + This data will be pulled from the metadata of Investment and LevelUp Action Items.
  + Whenever a user’s Budget Item is requested, it should be automatically updated to contain the latest transaction data.
  + It might not actually be much more efficient to just append the latest transaction data instead of pulling all the data every time.
  + If it is not too slow to pull all the data every time, this method might allow more flexibility.
  + The Budget Item could contain controls to view different periods of time, and maybe do calculations of some sort (I’m not sure what kind of calculations would be helpful at this point).
* Indexing metadata
  + Indexes should be created for specific metadata children (or even specific ancestors), not for the whole metadata element.
  + The index should be customized to suit the kind of data that metadata child contains
    - For a metadata child like <question />, only a presence checking Index is required
    - Some metadata might contain text designed for humans (i.e. containing sentences & such). These metadata children might require a “substring” index.
    - Metadata that is designed to be machine friendly and stores single, discrete values in a metadata element, will only require an “equality” index.
* One good thing about allowing users to edit other users’ items, is that people who want to help make other people’s data more machine readable (including by search engines), can add semantic metadata to their documents.
  + A user might add an item containing a list of good products for a particular purpose. They don’t bother (or don’t know how) to add metadata inside the Document html, to define that data as a “positive list”. Another user (of a high enough level), could look at the html of this item and add in the appropriate metadata to describe the list as a “positive list”. Once semantically described in this way, the text inside this list should be searched when searches are made for keywords that need to be found within a “positive list”.
* A “comments” type item cannot have <children> editing permission raised above 0.
* Because of the 20% tax on Investments, only about 1.6 X initial Investment is made back after the item’s worth increases by a factor of 10.
  + This is from my equation where .
  + If , then
  + Also, if p=0.8, an increase of factor 3.5 is required to make your money back, instead of 4.2 for p=0.7
  + I don’t want to reduce the 10% earned by item owner because, for one thing, I like the way it creates the neat rule that a User’s level must be >= 1 level below their highest level item.
  + I could reduce the tax to 10%.
  + A tax of 10% allows, in theory, by cyclic investing, a multiplier effect of 9X.
  + Getting the full 9X multiplier is impractical because it requires investing infinite times
  + With only 10 investments
  + Since you can’t invest in your own item, you’d have to divide this multiplier effect among multiple users. This diminishes its usefulness.
  + A possible solution to this whole dilemma: components of an investment that would go back to yourself (because you have invested in that item before) instead go to the treasury.
  + This would mean that with just two users, p=0.1 after the first two investments, so the multiplier is at infinitum.
  + Using many users, so that one investor isn’t re-investing in the one item....
  + I’ve glimpsed the skeleton of a very complex process that might allow you to still get a noticeable overall multiplier effect using many users. It involves sending an initial investment through a complex web of investments among many users; avoiding letting any one user invest in the same item twice (or at least while they still own the dominant share of investments in it).
  + If anyone is determined enough to use a method this complex, they probably deserve to get the small multiplier effect it might glean.
  + The total multiplier could never be more than 9 (for p=0.9), and it would be divided up among multiple users. i.e. if divided among 3 users, each user’s item could gain worth no more than 3X initial investment.
  + You could use an “out and back” mechanism, where the main user invests in an item whose owner invests back in the target item of the main user. Then the main user invests in another item owned by a different user and so on. I am assuming all these users are controlled by the one “being”. Maximum multiplier effect that can be gained for the target item is
  + The tax is still necessary to limit this very complex method
  + The tax also adds a cost to “giving” cash to other users by investing in a new item of theirs.
  + I still want 90% of the first Investment (all but the tax), to go to the item owner.
  + On second thought, this design is what allows the complex method of using the multiplier effect to work, and it may not be necessary. It might be more sensible for 90% of first investments to go to the treasury. After all, in most cases, users shouldn’t care what actually happens to the currency they invest. They still get the investment worth that much. The only question would be the impact on the virtual economy.
  + Investments from the treasury should give 100% to item owner as a first Investment and operate normally otherwise, except without the 10% tax. This is just to remove superfluous payments back to the treasury.
  + I have decided that for first investments in an item, 10% goes to item owner, 90% goes to treasury.
* The net worth of all items will grow disproportionately to the amount of cash injected into the economy. This is because every unit of currency is spent multiple times and each time it is spent it contributes to the worth of an item.
* Since user worth is total amount spent and all of an investment contributes to an item’s worth, net user worth and net item worth should be equal across the whole economy.

# Fri 11-Dec-09

* At request time, save a timestamp that will be used for all items created from that request. For some requests like create\_user, this will lead to multiple items with exactly the same date\_created. This is fine. Timestamps should indicate the moment of action and not worry about technically when it is carried out. This system will also allow date\_modified to be saved in an action. If the database is then recreated from the action items, all items will have the same timestamps as they should.
* Search function should have ability to add multiple items for which the results must be descendants (the item from which the search is performed is in this list by default and can’t be removed).
* Namespaces should probably be used with the metadata of Action items. It will be very important for data in Action Item metadata to be unambiguously and uniquely defined. There will be Indexes on specific subelements of <metadata> which must not have a chance of including data entered by users in <metadata>, <summary> or <document>.
* It could be realistic to allow permissions setting within specific <div> or <span> elements inside the <Summary> or <Document>
* The permissions system should follow the simple rule that an element’s edit permissions are defined by the maximum @edit\_level of any of its ancestors.
  + I can’t actually think of a situation when you’d want to define a whole element us uneditable, and then specify a particular subelement only as editable by low levels. However I think it would make more sense to allow this flexibility.
  + Therefore permissions should be set by the most specific @edit\_level attribute, rather than the maximum one.
  + This could allow an item’s <document> to be mostly locked, but allow low level users to edit sub sections of it.
  + This feature will probably be advanced to implement smoothly. Ideally, the rich text editor should cater for which elements it is allowed to edit.
  + Users without Javascript will just have to know that they can’t change an element that has a @edit\_level attribute higher than their level.
* <author> element should be replaced with <owner>. This is because some items (like the “User Item”) will be owned by a user that didn’t create it. Mostly this will be in the case of auto generated Items like Event Items.
  + I have been thinking about the User associated with an Item as the Item’s owner for a while, and I just realised that there are some cases where the owner isn’t the author.
  + Items that were actually made by a user can be identified by being subitems of their “User’s Items” item.
* Encourage users to put microformat semantic metadata in <document>s
* I’ve been feeling like I need to think of every conceivable feature I might want in my site and design the architecture to support those features, before I get too far into actually making it. I’ve realised this is - and more importantly, will continue to, hold me back from getting it up and running.
  + I need to instead consider how to make the architecture extensible to new, as yet unforseen features.
  + I’m not really sure how to do this. I think, though, that it’s more important that I actually create the site. I don’t have to account for everything before I even start.
* I’ve realised the architecture of my site is already really conducive to including semantic descriptions in RDF format.
  + Every single page on the site is identified by an ID, and can be referenced as a resource.
  + I can create Items to define custom predicates (an RDF concept), and reference them with URIs.

# Sat 12-Dec-09

* Parents page should show each parent as a full item, inc. summary, worth info etc.
  + Items that describe the relationship between its parent and its children are predicates. a "metaitem" (<metadata> sub-element) should allow this predicate to defined by a URI. Custom Predicates can be identified and described by items in the site.
  + e.g. an item describing a parent company like Nestle, might have a subitem called "subsidiary companies of Nestle" which would have subitems like "Peters". "subsidiary companies of Nestle" should be defined by metadata as a particular predicate, "is subsidiary company of".

# Sun 13-Dec-09

* Store revisions within each item.
  + Maybe create a new field called <revisions>. This would allow metadata to be stored in each revision in a cleaner way.
  + When item is created or modified, store whole item (except <revisions> element) in <revision rnum="#">.
  + This will mean duplicating the whole item at creation time and with every edit, but it will be easier to store data about changes made at the time of changes.
  + Maybe it would be better to make revision 0 when the first edit is made. This would reduce a lot of redundant information
  + Revision 0 and revision 1 would be created on first edit. After that, each edit creates a revision storing the item after the edit.
  + Omitting revision 0 until the first edit would be more consistent for items that will never be edited, like most Event Items, and View Items etc.
* Make a Complaints item with subitems for each level including admin.
  + To complain to level 5 you have to be at least level 3 etc.
  + I’ll set permissions for admin section to be the highest level that has over about a dozen users. This permission would have to be updated as users level.

# Mon 14-Dec-09

* A User’s password hash cannot be stored in a publicly available item. However, it would be useful to store it in an event so that the site can be rebuilt from events with user access included.
  + Make a flag that specifies an item as invisible.
  + This could be used to create invisible events that store such things as a user’s private settings changes (including password changes).
  + Items could also be made invisible for legal reasons, such as copyright infringement. This is more in line with my design principle than having to override the event recording system when erasing data.
* The permission system allows the same interface to be used for almost all site maintenance, by any level user including admin.
* The MassEdit interface will possibly remove the need for reversibility info to be stored in event items.
  + The value of being able to reverse events would be to reverse all actions done by a destructive user.
  + Actually, now that I think about it, reversibility is still required. MassEdit can be used to deal with Flooding, but vandalism requires that the specific actions and the details of edits be reversed.
* When creating items, and in fact in any event items, the details of the implementation should not be described in the event data. There should be enough information to do the action, and reverse the action, but no more.
  + This could potentially allow the whole architecture of the site to be redesigned, and the event data from the old implementation could be used to “rebuild” all the data in the new architecture. The new architecture could rebuild each event in the new implementation.
* Put a “reverse actions” form on a user’s Item
* When reversing a set of actions, check that the “new” value of a field matches the current value. If so, revert to the “old value”. If the current value is different from the “new” value of the action, don’t reverse that field.
* Even the login and “create user” pages should be items
* Every setting change should be a single “invisible” event.
* Every browser request that does an action should be associated with a single event with the timestamp corresponding to the time the browser request is first received.
* Updated levelling table

|  |  |  |
| --- | --- | --- |
| **Level** | **Currency required to reach level** | **Progress bar colour** |
| 0 | 0 | White on grey |
| 1 | 100 | Blue |
| 2 | 1000 | Green |
| 3 | 10,000 | Yellow/gold |
| 4 | 100,000 | Orange/bronze |
| 5 | 1 million | Red |
| 6 | 10 million | Purple |
| 7 | 100 million | Black |
| 8 | 1 billion | Badge replaces progress bar |

* When a user is created, several items must be created automatically, including the “User Item” and its subitems “User’s Actions”, “User’s Items”, “User’s Investments”, “User’s Finances”.
  + These item creations should be stored in a single event – the user creation event, because they were caused by a single browser request.
  + I am unsure whether this is the best way to do it, actually
  + In the Revisions section of the “User’s Investments” item, say, the “revision 0” description should read something like “Barneyii creates item T5YY”
  + Actually, it might be ok for this description to be “user Barneyii is created”. It’s not ideal, but “Barneyii creates item T5YY” is not really true either, since it’s an automatically generated item.
  + It would make more sense if the description for revisions was more like “created on 2009-12-14 at 6:54:13 PM”
  + This could be solved if a description was separately created and stored for each revision, instead of being pulled from the Event Item.
  + I think it would be cleaner to separate the automatic creation of default User items into separate Event Items than the “create user” event.
  + If this were done then there has to be a way of defining what order the events occur in. The User must refer to the “User Item”, which must therefore already exist. Therefore the first event that must occur when creating a new user it to create the “User Item”
  + If the timestamp of the events were different the order could easily be established.
  + The timestamp should be saved in an event when the event is created.
  + The timestamp of items created by a “create item” event should be the same as the “date\_created” of the event. This is for consistency if the database were rebuilt from the event items.
  + Similarly, date\_modified of item modified is set to the “date\_created” of the “edit” Event Item.
* I’ll think I’ll revert to the idea of making the main “User Item” be his private Profile item, which only he can edit. I’ll make a subitem for controlling levelling of the user and some other tools.
* It might be confusing that the User Item (or a user’s Profile item), still has a measure of the item’s worth and level as well as the owner’s worth and level.
  + Since people will associate the user’s Profile item with the User (as they should), they might expect that the Profile item’s worth and level should be the user’s worth and level.
  + I can’t think of a neat design change to fix this, so I’ll just have to explain the distinction somewhere.
* Store the user’s level, worth and funds as metadata in his “User Item” so that searches can be done on users.
* Since there is some official information like a User’s level, worth and current funds, to put in their User Item, maybe I’ll reverse my decision again and make the Profile item the subitem
* The main User Item can contain:
  + Controls for levelling the user
  + Control for reversing User’s action over a period of time in the case of vandalism
  + level, worth & funds info
  + official notices that concern this user’s eligibility to be levelled
* Since I want to store level, worth & funds info in the User Item anyway, and it would be silly to update these values in two places every time they change, I could store these values in the User Item only.
  + The Users container could be replaced by a “Settings” container that only stores a user’s private settings.
  + Even better, another item called “Settings” could be added as a subitem to the User Item. This item could be made invisible to keep private information like password and financial details safe.
  + The event that creates the User’s “Settings” item should be invisible as well
* Create user event sequence
  + Create “User Item”
    - Add parent “Users”
  + Create “User’s Actions”
    - Add parent “User Item”
  + Modify above two Events
    - Add parent “User’s Actions”
  + Create Settings
    - Add parent “User Item”
    - Set password
  + Create “User’s Items”
    - Add parent “User Item”
  + Create “User’s Investments”
    - Add parent “User Item”
  + Create “User’s Profile”
    - Add parent “User Item”
  + Create “User’s Finances”
    - Add parent “User Item”
* The first two events: Create “User Item”; Create “User’s Actions” do not automatically get put under “User’s Actions”.
  + All other events automatically have “User’s Actions” and “Events” added a parents
* To reduce the minimum height of an item
  + Put Owner level in the top left corner of their Avatar image, like in an RPG
  + Possibly show the owner’s progress bar underneath Avatar image. More likely I won’t show anything but the User’s level on each individual item. You’ll have to go to the User’s Item to find more info about them.
  + Show Item progress bar at the bottom of item, with almost the same width as the item i.e. progress bar is a percentage of item width
  + Show Item Level in a corner of the item (or maybe at the end of the progress bar or something).
  + Show Item worth on top of the progress bar in the centre. i.e.
    - ====================== 7045 =========-----------------------------
    - Where ======= is the completed section, and ------- represents remaining worth required to level

# Tue 15-Dec-09

* To make a field only editable by owner: plevel="private". This feature would apply to Profile item.
* Make User's Settings item only viewable by owner. add metaitem: <private />
* I’ll probably store “view” events as Items as well, and create statistics from them.
  + View Events store data about every page request, such as viewer’s IP address, OS, Browser, page viewed etc.
  + View Events should be invisible because there would be privacy issues with making that sort of data public
  + View Events obviously don’t require extra events to create them. (Only Actions specifically requested by a user require an Event Item).
  + View Events won’t need to be voted on or have children or resources or revisions etc.
* Architecture should be designed to not require every child element of <item> to be explicitly present. This will allow easier extensibility.
* Required elements should be added when first required, i.e. the <children> element is added when the first <child> is added

# Thu 17-Dec-09

* Problem: currency can be "given" to other players by investing a large amount in an item they are the sole investor in. currently you can make an unlimited return on a tiny investment if the next investor invests huge.
* Possible solution: you can only earn up to the value of your investment from any one investment. If an investment is worth more than current item worth, the difference goes to treasury.
* A user's own prior investments should be used in the division of their later investments, but any currency that would have gone back to them instead goes to treasury.
* It is possible to donate to your own item. Currently this will increase item worth proportionally to money donated. Since you get the money back, anyone can indefinitely increase the value of an item with them as donation target. I don't even want a rich person to be able to massively increase an item's worth with donations.
* A donation must not contribute to item worth. Instead let it add to a field I’ll call <invested> which measures investments by normal means, by donation, or by purchasing a product through the website. <invested> field will be used to divide up each investment. A donation still earns currency from subsequent investments, but does not itself add worth to an item or generate new currency.

# Mon 21-Dec-09

* Make a distinction between Users and People.
  + A User should only contain information about that User’s activities within the site.
  + The Profile concept I had originally thought of attaching to a User should be moved to an Item which identifies a person
  + A User item can be linked to a Person Item.
  + A Person Item should be able to be linked with external resources that identify people, like social networking sites.
  + It should require password confirmation to link a person to a social networking account like a Facebook or Google account.
  + This would confirm that the user linking the Person item to the social networking account is the owner of the social networking account, but it doesn’t confirm that the Person Item they are editing should be linked to that social networking account
  + Perhaps a User account should be optionally associated with a person. This would allow the password confirmation from a social networking account to identify the User as the person represented by the social networking account.
  + I want it to be possible for any User to create a Person item representing any person.
  + What’s most important is that the correct Social Networking account is linked to a Person item. I could allow any user to attach a particular social networking account to a Person Item and rely on a peer review process to choose these links correctly.
  + I could allow a User to link themselves to a Person item by giving Password confirmation that they own a social networking account linked to that Person Item.
  + This concept is reliant on the community choosing the correct social networking accounts to link to Person items. I can’t really rely on this, so it might be dangerous to allow a User to link themselves to a Person item through a social networking account that might not correctly identify that person.
  + I just realised it is the social networking account that is the identified entity. Whether that account identifies a particular person is a different matter, but I should be worrying about linking Users to social networking accounts. This is easily done with password authentication.
  + A Person item can then be dynamically edited to link to multiple social networking accounts. If at a particular time a Person item links to the same SN (social networking) account as a User, then that User is at that time identified with that Person. However, this more flexible architecture allows peer review to resolved incorrect associations easily. This is just a matter of correcting the SN accounts linked to a Person item.
* The purpose of identifying People with items is for semantic structure.
  + A company can have a list of People subitems which represent employees.
  + People can have subitems representing things they have done in the real world
  + People’s real world deeds can be criticised

# Mon 27-Dec-09

* register Zutha.com (.org .net)
* redirect to ZuthaNova.com
* other domains that could be registered:
  + zuthapedia (encyclopaedia)
  + zuthanary (dictionary)
  + zuthabooks
  + zuthapeople
  + zuthaproducts
  + zuthadebates
  + zuthacause
  + zuthaprojects
  + zuthagroups
  + zuthaware (software)
  + zuthaideas
  + zuthanswers
* These could each be portals that treat their topic item as the top item.

# Tue 28-Dec-09

* I need a way to let entities have different worth depending on context.
* e.g. an item called "most lovable fictional character" should contain subitems representing characters that can be invested in to determine the winner of the list.
* However an item called "weirdest characters" might contain some of the same characters, but they should have independent Worth in each list.
* Each character could be represented by a different item in each list. This would be somewhat messy, though and would undermine the concept of items uniquely identifying an entity.
* I could make a kind of item called, say, "contender" which acts as the recipients of votes in specific competitions. A Contender item should allow only the one parent so that each competition is isolated. Contender items should link to the entity item they represent - possibly such that the summary and document is pulled from the entity item.

# Wed 30-Dec-09

* Every xml element can contain a @visibility attribute that can either be “visible” or “hidden”. This allows me to, for example, store IP addresses in event items but keep this information hidden from the public.

# Fri 1-Jan-10

* Add native support for argument mapping into the hierarchy structure
  + This could be done by creating an argument mapping specific item type
  + The different kinds of elements found in an argument map could be identified by the item title. The content of the item would be the same as the content of an element of an argument map.
  + One challenge to overcome is how to support co-premises.
  + It would need to be possible to show the argument map in a more comprehensive view than the default linear list of children.
  + Implement a mathematical system for measuring the outcome of the argument map (as discussed with Damian)
* One day, it should be possible to upgrade the interface for interaction between people and the database to support such technologies as brain machine interface. All that’s required now is to make the data as semantic as possible.

# Sat 2-Jan-10

* Store all CSS and XSLT data for any possible page in the site in a single file each. This will mean only a single XML file will need to be sent with each request (if the setting is chosen to use XSLT on client side rather than server side).
* Consider only showing the Avatar image for the Main Item, and not for all the subitems. This will reduce the number of server requests as well as page load time and bandwidth. It will also allow items to be potentially more compact.

# Sun 3-Jan-10

* If it is possible to assign accessibility/visibility settings to every subelement of an item (such as individual meta-items):
  + All User info can be stored in the User Item. Password hash and any other private details can be hidden from the public but still be stored within the item
  + Statistics info, like visit events can store confidential info like IP address, but hide it from the public
* Item Creation process:
  + User Request to create an item, with relevant data is sent to a “create event” method.
  + “create event” method creates an item (using “create item” method) that contains all information sent by the user request.
  + In this case the Event item is of type “create item”. It contains initial data for the item that will be created from it. It also contains a separate “add parent” action for each parent.
  + The Event item is sent to a “execute Event” method, which executes each Action inside the Event item in order
  + Each action has its type verified (to check it is a valid action) and is then sent as an object to the relevant method
    - E.g. if the action is a “create item” action, the data contained in the action will be sent as xml to the “create item” method. The “create item” method will know how to deal with the data
    - This moves the parsing of the specific action data to the relevant method so that each action is completely independent and it is easier to make new actions.
* Use namespaces for metadata children so that indexes on specific metaitems can be specific to the elements that appear inside metadata and not get confused with the same names element elsewhere in an item.
* At the end of doing an Event, additional “common” tasks should take places, such as creating revision 0 (or whatever revision is relevant to the edit) and linking this Event Item’s id to that revision.
* Levelling table:

|  |  |  |
| --- | --- | --- |
| **Level** | **Currency range** | **Progress bar colour** |
| 0 | 0->10 | Grey on white |
| 1 | ->100 | Blue |
| 2 | ->1000 | Green |
| 3 | ->10,000 | Yellow/gold |
| 4 | ->100,000 | Orange/bronze |
| 5 | ->1 million | bright Red |
| 6 | ->10 million | deep Red |
| 7 | ->100 million | Purple |
| 8 | ->1 billion | Black |
| 9 | 1 billion + | Badge replaces progress bar |

* handling cloud computing architecture:
  + Event items can be created with no fear of clashes.
  + It doesn't matter if the actual execution of events lags a bit behind their creation. The lag could even increase or decrease depending on current server load.
  + Events are sent to a central server and executed in order of event creation. The changes are then distributed to all supporting servers. This could be done simply by sending out the event items and having them executed on each supporting server. It might, however, be important to use more advanced means to synchronize servers to make sure the database remains identical on each one.
  + If a clash occurs then the latter event is rejected. The event should probably then be deleted. The user that requested the event should be notified of the event failure.
  + Since this would have to happen with the server response to their request, the delay between event creation and execution could not be too long.
  + This system could be implemented with the same structure that would allow an API. Xml events are sent around and executed.
* A percentage of revenue from Affiliate programs should go to the owner of the item that earned the money. This creates an incentive for item owners to get the affiliate program set up with their item. It also creates a palpable financial incentive to post product items on the site.
  + A percentage of donations should also go to item owners for the same reason.
  + The affiliate and donation payment details need to be confirmed by a high level user before they become usable.
  + Anyone can initially add the financial details to an item which allow donations or product sales to be made from it. If the user that initially sets these details has a low level, the details will not be active, but merely suggested details. A higher level user (something like level 6) will need to confirm the details.
  + If a high level user is the initially creator of the financial details, then they can be available right away.
  + There should be serious consequences for a high level user adding incorrect payment details to an item. This abuse could raise serious legal issues so the punishment will probably have to be as bad as reducing the user’s level significantly. It must be impossible to abuse this privilege more than once.
  + These financial details can always be modified later.
  + The level required to change them can easily be raised by a very high level user or admin once the details are confirmed beyond doubt to be correct.
  + I’ve realised it is going to be necessary for admin to create an affiliate account with each item’s associated company.
  + It may be that all the owner of an item can do to hasten the activation of an affiliate system for his item is to make an “affiliate program request” item with a link to the relevant affiliate account creation page. Admin will then have to deal with this.
  + This would be a safer solution – and probably the only way to do it.

# Mon 4-Jan-10

* All information should be sent in and out of the database (or model) as xml
  + Xml documents called “requests” should be sent to the model to process.
  + The model will return an xml document
  + The returned xml should contain enough information for the appropriate XSLT file to transform it into the response html page.
  + This system will allow easy integration of an API system which simply requires the “requests” to be sent to the server as appropriately formatted xml documents
  + Each “request” document should contain the editor’s username and password-hash
  + When server receives an html request it immediately transforms the Post and/or Get data into a “request” xml document and sends this to the model
  + Every “request” will be transformed into an Event item and then executed.
  + After an action request like create\_item, a default view\_page request will be executed that returns a page to the user.
* Failed events should also be stored
  + Event items should have an element called <status> that is either “executed” or “failed” (or maybe even “pending”)
* Requests should simply contain the data from the forms in xml format
* For “modify\_item” request, the model will process the data to check which fields were changed and make individual Actions within the Event item to represent each changed field
* I need a versatile means of editing permissions on xml nodes that doesn’t require user to edit the raw xml.
  + Since almost every node in an item can be given an edit permission level (plevel), plevel needs to be stored as an attribute inside each node
  + This makes it difficult to have a separate form for editing permissions because the permissions cannot be represented as a list – only as a hierarchy
  + Maybe I could create a dynamic hierarchy interface for editing permissions.
  + There are fields whose permission can be edited that will not appear on the Item edit page, so permission editing may need to be separate after all.
  + The permissions tree could simply be a clone of the structure of the item but with plevel attributes inside the relevant elements

<permissions>

<item>

<title plevel=”3” />

<level levelup\_gap=”2” premature\_levelup\_gap=”4” leveldown\_gap=”4” />

<document plevel=”3”>

<span plevel=5>

</document>

</item>

</permissions>

* + The permissions section would need a namespace
* Making it possible to set permissions for any node causes quite a few challenges
  + The main reason I need this functionality is to set permissions on specific metadata.
  + The primary challenge with the above system of a bare, replicate tree containing permissions info is that nodes inside document may not be unique, so it is difficult to target them.
  + If permissions only need to be set on sub nodes of metadata, then the issue of non-unique nodes in <document> is not an issue
  + The issue still exists for setting permissions on individual <parent>s however.
* The javascript version of the create\_item page should have a dynamic tree of all the nodes that can have permissions added.
  + A text box next to each node will allow setting the permission on that node.
  + The data for this tree will be pulled directly from the plevel attributes integrated into the item’s xml structure. No separate node for storing permissions will be used.
  + For nodes like <parent> that are not unique, additional information may need to be added to this tree, such as the id of the parent.
* Item Level permissions should not be editable on item creation. Possibly permissions will not be editable at all on item creation.
* Any top level elements like <title>, <metadata>, <parents> that are present in a “request” should contain the full content of what is to replace the current data.
  + Only top level elements can be missing from a request.
  + If a <parent> is missing from <parents> it is assumed that its removal is requested
  + If @plevel is missing from an element, it is assumed that its removal is requested.
    - This means the permission will be taken from the closest ancestor with @plevel set.
    - If it is a top level element like <children> then removing @plevel will either set it to default or be disallowed – I haven’t decided which yet.
* Actions should be independent of their context (i.e the event they reside in)
  + Actions should be able to be sent as they are to an action executor
  + Each action therefore requires its own <target>
  + This will allow a single event to contain actions targeting multiple different items
  + This is required in the case of mass event reversals or mass edits

# Tue 5-Jan-10

* If @plevel data is stored in attributes of the action metadata of event items, then this indicates the edit level of this part of the event item. This is not wanted because it means low level users could edit the metadata inside an event.
  + This could be solved by ignoring any permissions on metadata elements below the top level “metaitems”

# Sat 9-Jan-10

* Make a “search” item type
  + Search items could be used to, for example, find all items beneath the “books” item of type “book”.
  + This would allow users who aren’t confident with generating their own searches to easily get access to common searches.
  + A search that finds all book items under “books” would show the highest worth books at the top
  + This means users adding items only have to worry about categorizing their item specifically (such as under an Author item), and not generally (such as under “books” item).
* Use a separate ID sequence for Settings and Events
  + The base 32 character set should omit “s”,”e”,”o” and “i”
  + Settings items should all start with lower case “s”
  + Event items should all start with lower case “e”
  + Charset: 0123456789ABCDFGHJKLMNPQRTUVWXYZ
* Check if it is feasible to divide up the items.dbxml container into multiple containers and still be able to have full flexibility in searching
  + This could be done by making a separate index.dbxml database that simply records the container that stores each item
  + It might also be possible to do a join of all containers using XQuery
  + A search under a particular item would pull the list of Ids from the descendants of that item, then make a set of all the items corresponding to those Ids from the join of all items in all containers.
* Item top level types (defined by <item> element’s @type attribute):
  + Normal
  + Setting
  + Event
  + User
* Only admin can create Settings
  + Creating a “Setting” type item should have its own Request and Event type and be available only to admin
* I seem to need about 3 tiers of item types:
  + Types that are site structures and should not be directly creatable by users
    - Event, Setting, User, Normal
  + Within Normal type there are item types that affect item functionality, and are creatable by users
    - Argument\_map\_node, Search, Competition
  + Also within Normal type are types that are for semantic and searching purposes only
    - Book, Product, Software, Hand Tool, Person, Game, Country
    - This tier will definitely have types that will overlap (e.g. “product” and “Program”)
    - Therefore this tier needs to be more in the format of labels, where multiple labels can be given to a particular item
* Category items can contain template data which is automatically added to the metadata of their subitems
  + an item at: {Books->Books by Author->Books by Robert Jordon} could contain template data that cause a subitem to automatically be defined as a resource of type “book”
  + I want the public to be able to define their own item types and make templates for their own item types
  + The item type “book” might be associated with a template that automatically puts a table at the top of the <summary> element of a new “book” item. The table might contain fields like Author, publication date, publisher, revenue etc.
  + This template system could be made flexible enough to define specific options for fields.
    - The Argument\_map\_node item, could have its possible values for <title> limited to {Support, Opposition, Against, Co-Premise, Reason, Evidence ...}
  + In order to do advanced things like show an argument map whole, or export it to Rationale, code would need to be written.
    - I need my program architecture to be flexible enough to let me add in objects to my PHP object tree when advanced functionality is required for an item type.
    - If an item type does not have its own dedicated PHP object, then it needs to be dealt with by a default handler.
* I can join the last 2 tiers into one more flexible tier.
  + Any advanced functionality in “Normal” items will be provided by dedicated PHP subclasses of “Normal” item
  + For an item to benefit from the advanced functionality of types like “Search” and “Argument\_map\_node”, they must be linked to a resource item that defines that item type.
* For consistency I’d like to be able to define the top tier of items in the same way (Event, Setting, User, Normal)
  + This would require a resource item that defines the type “Event” to exist before the first Event was created, but an Event should be made for the CreateItem event that creates the resource
  + It is really only the Event type that is causing trouble. User and Setting types seem to fit far better with the rest of the normal types
* Perhaps I should separate the concept of Events from Items
  + The confusion of creating items to represent the creation of an item has been getting to me already.
  + Site Events are fundamentally different from an item which represents a real world piece of information.
  + Settings and Users also contain information specific to ZuthaNova.com but they contain static information and they are conducive to thinking about as Items
  + Events don’t fit the concept of an Item (vague as that concept is), as well as everything else in my site.
  + Events still need to be treated like items ad appear in the hierarchy so they can be commented on and appear and a “User’s History” item etc.
* I’ll make the top tier of types contain two “classes”: Event and Object
  + <item class=”object”> OR <item class=”event”>
  + Object will encompass Settings, User, Resource, Search, Argument\_map\_node, Book, TV, Product, Person etc...
  + The only kind of item that is “created” by a User, is an Object
  + Events occur as part of the site structure – they are not “created”
  + This distinction means “class” is never selected by the user
  + Admin can now create an object of type “settings” without the messiness associated with selecting “type” from a list that also contains the “Event” type
* I’ve changed my mind about having separate ID sequences for Settings and Events
  + There is no big problem if the ID numbers get long as time passes because every browser request causes an Event with its own ID
  + The bigger issue is the database not handling the amount of items that will eventually be generated from storing every PageView Event
  + This could be solved by splitting the database into multiple Containers
* Actually it still bothers me that the majority of IDs will go to ViewItem Events.
  + I would prefer that ViewItem Events, and probably Events in general have a separate sequence of IDs
  + However I like the consistency of a single ID system
  + If I put a lower case “e” before every Event ID, but still omitted “S”,”O”,”I”, and “L” from the base 32 set, then it would be like a Base 31 id system for Objects
  + I think it makes more sense to make “e” an extra letter, making a strictly Base 33 system, but Object IDs would effectively use a Base 32 system.
  + I’ll re-include “L” into the character set since it doesn’t look like any other letter as long as it is capital
* I’ll put Events in a separate container to begin with
  + This will allow me to test out the multiple Items container database structure and make sure it is sound
  + It will be Events that will accumulate fastest and will most need to be stored in multiple containers eventually (when they are in the order of billions)
  + I may not ever have to make an additional items container because there will be several orders of magnitude less Objects generated than Events
  + Since it will mainly be Objects that need to be searched, keeping all Objects in one container will make the process easier
* It should be possible for advanced users to create their own Request xml document and be able to do advanced tasks like:
  + Queuing many actions in one Request.
    - This could be useful for complex editing tasks which require a series of actions to be done before the state of items will be stable again
  + Joining the search results of multiple search queries
* A system should be set up to allow item owners to “sell” products for Zuth (virtual currency)
  + I could do this with an interface like PayPal.
  + The sale page of a product could have an option to pay with Zuth
  + The user would log in their ZuthaNova.com credentials and confirm the payment
  + The confirmation page would have a link to the ZuthaNova item that you are investing in, in order to receive the product. The transaction would then be performed from the 3rd party website through an API.
  + There would be a limited set of products that would be conducive to this system. Obviously products that cost money per unit could not work this way, since Zuth is not real money.
  + This system is purely for the intellectual property kind of product
  + The benefit of allowing people to buy your product with Zuth is that it will increase the renown of your product (this relies on ZuthaNova already being a big thing of course).
  + I thought it might be an issue that I, as admin, can buy any of these products available for Zuth credit, effectively for free. I’ve realised that the whole point of this system would be people can get these products for free. What they are giving the company in return for the product, is public recognition. This applies just as well to me, as admin, buying a product in this way.

# Mon 11-Jan-10

* Eventually there should be an incentive for companies to suggest that their customers buy through ZuthaNova.com so that the item that represents the purchased product increases in value.
* Items that represent commercial products should only start requiring a subscription to show their worth after they reach about level 3.
  + The required subscription should increases steadily with the level of the item
  + The pricing would have to be flexible so I can tune the cost to the market

# Tue 12-Jan-10

* Use [Yahoo BOSS search API](http://developer.yahoo.com/search/boss/) instead of Google because Yahoo’s is much less limited.
  + Use Yahoo’s ranking from a search term and filter out all results that exist under the current item
* Other possible search APIs:
  + <http://www.filestube.com/apidoc.html>
  + <http://www.bing.com/developers>

# Fri 15-Jan-10

* The majority of items should be based on templates
  + An example template would be ‘book’
  + Certain items would enforce subitems of only certain types, where each ‘type’ is defined by a template
  + Example: the item ‘books by Robert Jordon’ would enforce only subitems of type ‘book’
  + Only xml data should be stored for item content
  + By using my own format for storing data, I can be sure I make it exactly as flexible as it should be.
  + Allowing any HTML, or even HTML with certain tags unavailable, would create a security vulnerability
  + I want as many items as possible to conform to templates in order to maintain as much consistency as possible.
  + The more consistency there is, the easier it will be to maintain an orderly system because users will be able to use the existing structure and formatting of items as an example
  + I want to allow the creation of custom items which don’t conform to a predefined template, but I don’t want it to be too easy for anyone to use such. Specific templates should provide enough flexibility to cover most users’ purposes
  + New templates can be request, or even created by users themselves.
  + It would be ideal if the template data was stored in items representing that template, so that users can create their own item ‘types’

# Tue 19-Jan-10

* The database and crucial components of the operation of the site should be contained behind a SOAP API
  + I will make my own client to access this API, which will provide the ZuthaNova.com web interface
  + Later on I can add additional Clients, each with their own caches, to speed up local delivery of content
  + By natively using the API myself, I ensure that 3rd party users have access to all possible functionality
  + I will also save development time by not needing to invest additional time developing a separate API
* When a user logs into ZuthaNova.com, the Client accesses the user’s settings through SOAP and stores these in the $\_SESSION variable
* I’ve decided I won’t allow my API to be used to host an alternate website frontend for the Zutha database
  + Allowing 3rd parties to host their own users and relay their requests on to the central server would not benefit me
  + I want to record all user requests including their browser and OS data etc. I would not be able to trust 3rd parties to relay this info on to me correctly
  + 3rd parties might also use caching (as I will with my clients), but not send the request to the central server to be recorded
* The purpose of the SOAP API:
  + For scaling purposes, I can host the content from multiple Clients distributed around the world, each with their own caching system
  + Mobile device clients can be built which connect directly to the central server via the SOAP API
  + Robots can access the site programmatically
* Every 3rd party call to the SOAP API must be by an authenticated user
  + For 3rd party API calls, I will store the Browser/OS details of the SOAP sender
  + For calls by my own Clients, the Browser/OS details will be sent with the SOAP message
  + My own clients will have to be securely identified

# Wed 20-Jan-10

* Make a search function dedicated to finding equations of a particular form
  + Use an algorithm to find equations that are identical except for the choice of variables
* The naming of items that represent equations or other concepts that need special characters to express will be difficult because special characters cannot be easily put into the URL
  + One options is to abandon the concept of putting the item name in the URL altogether. Special characters can then be allowed in Item titles
  + The main reason I want item titles in the URL is for Search Engine Optimization
  + People probably don’t look at URLs very often to find out what it is about
  + Most item names will be appropriate for the URL and it would be a shame to abandon this useful SEO principle just because there might be items that need to be named with special characters
  + Another option is to escape special characters: %2F for “/” for example
  + I’m increasingly feeling that putting item names in URLs will become messy to deal with. There are many kinds of names that will need special characters including questions, equations, foreign language names, accents on characters etc.
  + I’m already ignoring the item name part of the URL. It is only going to be a nuisance by requiring me to validate it and redirect requests with incorrect item names.
  + Many prominent sites (like YouTube) already use ID based URLs, and I don’t notice any negative consequences from that method.
  + I’ve noticed that most sites do seem to name their pages with the full name of the article. This is probably because this approach is recommended for SEO.
  + Having read some articles about URLs and SEO I am convinced it is a good idea to put the item name in the url
  + I could create a separate field called url\_name that is automatically generated from the item’s title, but is editable by high level users.
  + “why does e^(PI\*i)=-1 ?” could be turned into “why-does-e^(PI\*i)=-1” if adding most special characters is ok or “why-does-e(PIi)-1” OR “why-does-e\_(PI\_i)\_-1” if special characters are disallowed
  + Some item names (like names completely in Chinese characters) will not be expressible in URL-available characters.
  + I could allow the URL\_name to be blank. Such URLs would then look like <http://zuthanova.com/4H8K//SubItems?s=100&f=200#4TFFP> (where ‘s’ and ‘f’ are the ‘start’ and ‘finish’ index for which subitems to show and #4TFFP is a bookmark which will move the page to that particular subitem in the list)
  + I’ve realised that UTF-8 encoding is allowable in the URL so I can put Chinese characters in the url\_name
  + The main problem is dealing with ‘?’ and ‘/’ characters
  + Since I am using regular expressions to deal with the URL I can easily invent my own format for dealing with special characters.
  + The only time I really need special characters to be stored in the URL is for sending search queries by GET. Since GET data is automatically encoded to be URL friendly, this is no issue.
  + It would not be a problem to simply remove certain special characters
  + It seems common now for non ASCII characters like Chinese/Japanese/Hindi characters to appear in URLs. However it appears to be frowned upon to put symbols like ^,+ and \* in urls, and such symbols are not even registered by search engines anyway.
  + I’ve decided I will allow Unicode word characters (regex \w) and the ‘-‘ character in my url names
  + I found a webpage <http://www.urch.com/forums/gmat-math/130650-100-x-y-100-100-x-100-x-y-xy-100-0-xy-100-x-y.html> that has used what I think is a sensible strategy to dynamically create a url name for the title “**(100 + x) - (y/100)(100+x) > 100 x - y - xy/100 > 0 xy/100< x – y”**
* Items should have other categorization fields than <parents>
  + Such fields as <language>, <country>, <type> should be optionally available
  + They should point to items that identify individual languages, countries and types
  + These fields should go in <metadata>
  + It will be important for users to be able to set their settings or specify in a search to only show items in their own language or relevant to their own country or of a particular type (like book)
* When a SOAP request is returned, the correct URL to generate that returned document should be generated and compared with the actual URL. If the actual URL is wrong (eg the ID contains S instead of 5 OR the item name is wrong OR there is additional unrecognised fluff at the end ) the browser should be 301 redirected.
* I need an object type called “type” to define each type
* Items have a field that defines which item types it can have as children.
  + <allowed\_child\_types plevel=”5”>
* I need to upgrade the “comments item” concept
  + Items that allow only limited types as children still need to be able to be commented on
  + Such items that limit possible child types could still have large numbers of children and so a Comments item could easily get lost amongst them all
  + I could allow comments to go directly under their target item and have a filter option to show only subitems of a particular type
* The main controls for an item should include a filter for showing only subitems of a particular type
  + The parent item will have a field that specifies the default filter (i.e. “ALL” or something specific like “Book”)
  + Comments will go directly under the target item
  + Comments can be created by anyone under any item regardless of its <children> or subitem type restrictions
  + Comments should therefore not be visible by default unless they are the only item type present among the subitems
* Items should have different instances of <title> and <document> for each language they have been translated into
  + It would not work to make a whole separate site for each language since I would not be able to help maintain it
  + I want all users who gain a high level to be able to help the maintenance of the whole site, not just the items that are in a language they understand
  + All item types except “comment” should require at least an English version
  + Non-english speakers must create an English version of an item before they can create a version in their own language. This will be a hindrance to non-English speakers, but the problems that would arise with separate sets of items in other languages would be worse.
  + Comments can be created only in a non-English language
  + The url\_name should also appear in the chosen language (defined by ‘lang’ GET variable)
  + The main focus of the multilingual support will be to translate English items into other languages, not to create content exclusively in a non-English language
  + I feel a bit guilty about being so prejudiced, but the site relies on me maintaining the content and choosing people who are good at helping maintain the content, and I can’t do that for items in another language or people working with items in another language.
* I’m not sure what should happen to the alternate language versions when an item’s <document> or <title> is changed
  + I could simply let every change to any language version be its own revision
  + I could create some kind of notification feature to let users of the alternate languages know that one of the language versions has been changed
  + I still feel this issue is going to be messy and I’m not sure how to deal with it
* Perhaps I could narrow down the kinds of items that must have an English version
  + Items that represent real world concepts like specific books or programs or places etc. must have a single instance, with possibly multiple translations
  + Items which are reviews of a product or articles written by a ZuthaNova user do not really need to have translations. For such items, it would probably be better if translations were separate items, as it would prevent confusion arising from versions changing in only one language and not the others.
  + It is mainly items which allow only specific fields (such as Books, Software, Companies etc.) that should be unique and have internal translations.
  + Items with specific fields to be filled will be less susceptible to diverging content in different languages (a problem likely to arise with articles or freely editable content if such existed in multiple language versions).
* I think I’ll need to allow free html editing after all, for items like articles, which authors will want to be able to design freely and probably visually.
  + It would be difficult to create a javascript editor that shows the rich text document in real time, but stores custom XML tags instead of HTML. In fact, it would not necessarily make any sense to do this anyway.
  + I should still find a way to significantly limit the scope of the html that can be used.
  + The javascript editor can easily limit the kinds of html content available, but I need to make sure these limitations are also ratified by the API

# Thu 21-Jan-10

* If an item is created which duplicates another then it should be merged into the first created. The duplicate should be replaced with a notice linking to the correct item and have additional investments blocked.
* When creating an item of a type that should contain no duplicates (like Website/Book/Software/Movie) if the name given is already used, the requestor should be asked to check that his contribution is new.
* Item revisions should be accessible via a GET parameter.
* Events relevant to an item could be made subitems of that item. Event items would be filtered out of the SubItems page. The History page could show only subitems of class 'Event'
* Comments could have a dedicated page. I'm not really sure how to implement the type filter system - perhaps it's not needed.
* An item with only Comment type child objects should default to Comments page on open.
* What subitems should be shown on the Document page of an item with only Comments?
* Perhaps the Document page doesn't need to show subitems. I just need to appropriately define the default page to open. An article should open on Document, the Books category should open on Subitems, something like a recipe might open on Comments.
* Limit the number of results available to normal users to 1000 (like Google, Yahoo and Bing does)
  + Allow admin (and perhaps some high level users) to access more results so they can do large scale MassAction tasks
* Request Types:
  + ViewItem
  + CreateItem
  + ModifyItem
  + Invest
  + CreateUser
  + ChangeSetting

# Sat 23-Jan-10

* The Client (receiver of Post and Get requests), should sensor the data as little as possible
  + I want incorrect requests to be sent to the database and recorded
  + I will then be able to examine the data to find out what people are doing wrong, or if and how people are trying to hack my site.
  + It will be easier to maintain the code if I only have to change the validation of data at one end (the DB/Server end)
  + Get and Post fields still need to be escaped at the Client end.
  + If I try to send all data that is processed by the Client (URL, Get and Post), to the Server, it will become too messy and would require an alternate SOAP message format than the standard API
  + The main reason I want to record incorrect requests, is to find people who are trying to hack, primarily so that I can ask them to keep hacking in exchange for a reward if they find any vulnerabilities.
  + The only meddling that is worth noticing is effective alterations of POST or GET variables that actually make it through the simple Client filter, yet have incorrect syntax when they reach the Server end.
  + Another reason to let most of the validation happen at the Server end is that it will make the code easier to update.
  + The Client should only check syntax, not content of data

# Wed 27-Jan-10

* The design of the API should be independent of the implementation of the website
  + Instead of ViewItem requests, all information should be requested from the database using a flexible query mechanism
  + Multiple API queries may be made by the Client to return the information for a single page
  + The Client will put together the information it acquires from the database into a single xml document which will then be transformed into html for viewing either at the server or the browser end.
* The Client will send ViewItem events to the API with every request it receives from a user using the website.
  + Only my own Clients will be allowed to create ViewItem events
  + API calls from 3rd parties will be recorded as “API Queries” or something
* The whole concept of recording every View event is feeling increasingly messy to me
  + I feel like capturing all information is worthwhile because I might find a use for it later and what I don’t capture I can’t use
  + I want to record browser statistics, but that’s done by others anyway
  + I want to be able to look at what individual users are doing, but that is invasion of privacy and I can’t really think of a good excuse for having that ability
  + I want to be able to setup algorithms to find items that might interest individual users based on the kinds of things they have viewed in the past.
  + If it was known I collect all browsing activities on my site, that might cause a lack of trust
  + Adding data to the database for every View event would increase the resources required by the site substantially and slow response times for simple page views substantially.
  + I wouldn’t need to have the separate ID scheme for Events if I didn’t create events for every View event
  + If I didn’t try to record statistics about user-agents, it would remove the messiness of needing a slightly different API implementation for my own Clients compared with 3rd party clients.
* I need to record identification info about any events that modify the database
  + Stats data like IP address and User-Agent should be recorded in every Event item
  + I will need this data to, for example, reverse all actions done by an IP address
* I need a way to identify IP address involved in a denial of service attack.
  + I can easily limit the number of items an IP address can create in a given period of time
  + In order to detect an unusual number of view requests in a short time, as in a denial of service attack, I would need to record statistics about requests
  + If I limited the user accounts that could be created by each IP address, then I could use statistics about users to ban users that are abusing the site
  + At the moment, though, people can view pages without being logged into a user account. I would need to force people to log in to a user account to just view items if I wanted to rely on statistics about users alone to detect abuse (like Denial of Service attacks).
  + Denial of Service attacks can be blocked at the Client end. It should be a simple process to detect too many requests from the one soure in a short time.
  + I is better to have requests blocked at the Client than have to be sent to the API before it is detected that they should be blocked.
  + 3rd party clients of the API will be easy to block from abuse also, because there are a limited number of them, each with an identifying id code.
  + Web Servers can log events much more efficiently than I could by creating an Event item for every View event.
* Denial of Service attacks can be Distributed, so blocking IP addresses is unlikely to be that helpful anyway.
  + I will need advanced methods to protect against such attacks
  + Recording View events on the API end is not really going to help for this
* ZuthaNova.com should be more of an architecture to support an information paradigm than a specific service
  + The API should provide broad functionality so that 3rd parties can innovate to create their own useful features
  + Features like guessing items that a user might like from their viewing history can be left to 3rd party innovators
  + It should be a choice by users to allow their viewing history to be recorded and used for gleaning the kinds of items they might like
  + The implementation of the API should be as generic as possible. It should provide access to the functionality of the underlying database in the simplest, most intuitive way I can think of.
  + The website front-end that I will create should not influence the design of the API. It should use the generic API and form the data into the web interface that I have envisioned.

# Sat 30-Jan-10

* What happens when the first investment in an item is huge (like 1000 Zuth)?
  + When an item reaches each threshold (10, 100 and then 1000 Zuth) it is supposed to be levelled and every investor has their investment returned
  + If the item is levelled first to level 1 (from 0), should this single investor receive 1000 Zuth, then another 1000 for level 2 and another 1000 for level 3?
  + Levelling up an item to level 1 should only generate 10 Zuth as that is the threshold required to reach that level
  + If a single user has invested 1000 in the item and it is levelled to level 1, they should receive 10 Zuth back for that
* If the worth of an item reaches 1000 from many different investors, but the item has not yet been levelled to level 3, there should still be an incentive for users to invest in it before it has been levelled so they can gain from the levelup bounty
  + If a levelup event simply generates a fixed amount of additional Zuth (1000 for 3, 10000 for level 4 etc) which is divided proportionally among investors at the time of levelup, then there is an incentive to invest before the levelup event because you will gain some of the fixed bounty – but also cause existing investors’ bounty to reduce.
  + If an item’s worth gets a long way ahead of its level, then the amount earned by each user when the levelup finally comes will be less than the amount they have invested.
* If an item is levelled early, there should still be a fixed bounty produced.
  + This will mean users will earn back more than their investment
  + I can’t think of a reason why an item would be levelled early, but it would create a weird situation. There would be a diminished incentive to invest in an item that has been levelled early.
* Every item must conform to a specified Template
  + The template will determine:
    - which fields are available
    - which fields are shown in the “summary” view – the cut down version of the document shown for each subitem in a list of subitems
    - the XSL transformation to perform on the xml data to produce html
  + I want it to be possible for people to make alternate XSL and CSS data so that they can chose an alternate look and feel for the whole site.
  + The <template> field should not determine specifically what the item is. A given template is just a particular set of fields which may suit a wide or small variety of actual objects
  + The “book” template seems pretty specific, but a book has a specific set of data that is relevant to that object
  + The “Multi Part Document” template is much more broad. It defines a way of representing a general, large document using a multi tiered hierarchy of “Document” objects.
  + The “Document” template is used by the “Multi Part Document” template. It allows a flexible document to be created from multiple fields including paragraphs, embedded videos, computer code, headings, ordered lists etc.
* Templates:
  + Book
  + Product
  + Technology
  + Image
  + Document
  + Comment
  + Event
  + News
  + Business
  + Country
  + City
  + Culture
  + Group
  + Machine
  + Thing (e.g. drink bottle, rubbish bin)
  + Clothing Item
  + Natural phenomenon
  + Place
  + Person
  + Character
  + Fictional Universe
  + Sound
  + Art
  + Software
  + Debate
  + Record (such as the fastest 100m sprint speed or the largest amount of alcohol consumed – things found in the Guinness book of records)
  + Zutha Competition (e.g. most lovable fantasy character)
  + Zutha Contender (a contender for a Zutha Competition. Links to the original item. The Contender item allows separate investments specific to the Competition)
  + Category (e.g. Science Fiction Books, Companies, Books, Tutorials)
  + Argument Map
  + User (a site user)
  + Profile (the profile Item of a site User)
  + Setting (a setting item for the site)
  + Project
  + Financial Aid Appeal
* The Template for an item is defined by pointing to the Template item which defines that template.
  + This makes the definition of the template unambiguous
  + Templates themselves can have a hierarchy
  + An item defined by a template does not need to know anything about the template hierarchy in order to take its functionality from that template

# Sun 31-Jan-10

* All pictures should be stored in their own Image item.
  + To directly access image file use url format <http://zutha.com/5HG3/apple/files/apple.jpg>
  + The Image item should store metadata about the image such as its dimensions, mime type, the source of the image
  + Large images should automatically have a thumbnail image generated, stored at <http://zutha.com/5HG3/apple/files/apple-thumb.jpg>
  + Image items should show their thumbnail image for their summary view and the full size image in the Document view.
  + Abandon the concept of a Resources page and storing data for resources like images within the items that use them.
  + Any use of images in the site should use an Image Item as source
  + Image items (or other media type items) can link to external media. These items will supply a link that looks like an internal ZuthaNova.com link but which actually pulls the data from the external source.
* The <document> element should store all information about the content of the item.
  + The content of the <document> element is what is determined by the item’s template.
  + Every Item must therefore be based on a template which defines the data format of the <document> element
  + Some elements inside the <document> element may not be used by the Summary or Document views of the item so they are effectively invisible. This functionality replaces the need for a <metadata> element.
  + The <summary> element is no longer needed as the Summary view is generated dynamically from potentially multiple fields from the <document> element
* The API should be a source of data. It should not be involved in the functionality of the site
  + Template Items are just more items for the API. It is the Client that pulls the XSL transformation data from the Template and applies it to an item to produce the output HTML to send to the browser
* Text that represents an Item ID needs to be able to be identified unambiguously
  + If a different action needs to be taken depending on whether the input is an ID or a field definition of something, there must be a clear way of identifying an ID.
  + An ID like PARENT could obviously be easily mistaken for the keyword “parent”
  + Therefore, wherever an item ID is used, it should be preceded by a hash (#) e.g. #J4F1
  + I am not yet certain this will be necessary. I would prefer if it is not since hashes will probably have to be escaped in HTTP GET data and there are other annoying things about using symbols. E.g. double clicking an ID text will not select that hash part, which makes copying and pasting Ids a slower process.
  + I can’t find a single symbol that Firefox selects with normal alphanumeric characters on double click – even underscore.
  + In a search string like “HR6G AND B3RA” there is no unambiguous programmatic way of identifying the Ids and the AND operator
  + I could force the use of && and || syntax in this case, but I don’t want to because I want to make advanced search syntax as accessible to non tech-savvy people as possible.
* The <title> element should not be used for computer parsed information
  + The Title of an Item should express what an item is for human understanding.
  + The <title> element appears as the HTML Document’s title and in the URL
  + Examples:
    - A User item should be named with syntax “User: barneyii”
    - A template should be named with syntax “Template: Book”
  + Such rigid naming schemes for Items can only be relied upon for items controlled at admin level, as the above examples are
  + The appropriateness of item names is left up to the people who name them
  + The paradigm of not using the <title> field to parse data from only really applies to items which are part of the site’s functionality (like Users and Templates)
  + Both Users and Templates should have separate fields in their <document> element defining their names
    - User items should have a <username> field
    - Template Items should have a <name> field
      * This name will be used when specifying a template in a query
* String-based Query Syntax
  + Words appearing on their own will be matched to any of the <title>, @id, or <document> parts of an item
    - I think there will be an issue with this as I don’t think Berkeley DB XML will allow indexing of the text content of all ancestors of the <document> element.
    - Therefore words appearing on their own will automatically have a “Yahoo:” or “Bing:” wrapper placed around them.
  + “Template:”
    - A single template name or list of template names
    - All results must be of one of the templates in this list
  + “TemplateID:”
    - A list of one or more template IDs
    - All results must be of one of the templates in this list
  + “Childof:”
    - A list of one or more IDs or another search string
    - All results must be a child of one of the items in the list or in the results of the query specified by the search string
    - When this is converted to an API Query, this filter is called “parent”
  + Yahoo: or Bing:
    - Any valid search string for the chosen search engine
    - Some keywords (like site☺ will be unavailable because the search must occur within the ZuthaNova.com site
    - The results shown will be filtered by the results from the search engine API
    - There will be an option to order items by the order sent by the Search Engine API
  + Field:<fieldname> e.g. field:publisher:
    - A simple search string: a set of words with optional AND, OR and “-“ modifiers
    - The custom <fieldname> must match the search string
    - Some fields will have indexes that allow searching within the field and some fields will require an exact match
    - This filter is only available is the results are filtered by a single Template
  + created\_after: , created\_before: , modified\_after: , modifed\_before:
    - an ISO formatted timestamp
    - e.g. 2008-09-23T18:30:00Z or 2008-09-23T18:30:00+10:00
    - for timestamps that don’t specify the time zone, the time zone of the user doing the search should be automatically added
  + level:
    - an integer, optionally preceded by > or <
    - e.g.
      * level:3
      * level:<5
      * level:>1 AND level:<4
      * level:>1 level:<4 (some as above)
  + worth:
    - a number, optionally (usually) preceded by > or <
    - e.g.
      * worth:1000
      * worth:>100 worth:<1000
  + owner:
    - a list of one or more owners
    - e.g.
      * owner:barneyii
      * owner:{zutha,barneyii,artemis}
  + title:
    - one or more words to be found in the <title> element
    - e.g.
      * title:{lord rings}
      * title:{apple –computer}
      * title:{(“nexus one” google) OR “google phone” review}
  + id:
    - a list of one or more IDs
    - if a single ID is given, obviously only a single result will be returned (if an item with that ID exists and is visible for the requesting user)
* I can’t think of a neat way to allow searching of all the text in an item’s Document via the internal Index.
  + There shouldn’t really be a reason to use the BDBXML index instead of a proper search engine when searching the whole text content of an item
  + The Database index should be reserved for searching within specific fields
* Events related to an item should be stored in an <events> element of their own instead of going in the <children> element
* Events should go in a separate container
* Events should be named <event> instead of <item>
* I can’t think of a reason why Events and Items should appear in the one query so I won’t bother allowing for this functionality
* The structure of <event>s may be slightly different from <item>s
  + <event>s will not have an <events> element
  + I might even remove <worth> and <level>
  + Perhaps I’ll even remove <owner> from <event>s because an Event isn’t really owned by anyone.
    - This would lead to a significantly different visual layout for Events compared to Items because the Owner avatar image and worth information would be missing.
    - Every event will be performed by a particular user though, so the <owner> concept is appropriate. I think I’ll keep it.

# Mon 1-Feb-10

* Query Syntax
  + Presence:<field name>
    - <field name> is present in the fields of the result items
  + Contains:<field name>:<search string>
    - <field name> contains a substring that matches <search string>
  + Equals:<field name>:<search string>
    - <field name> matches <search string>
  + Template:
    - A list of one or more templates
    - Template names with spaces should be enclosed by double quotes
    - E.g.
      * Template:character
      * Template:”Wheel of Time Character”
      * Template:{character, ”Wheel of Time Character”}
* Some templates will exist in a hierarchy
  + E.g. the Character template might have sub-templates like “Wheel of Time Character”
  + It should be possible to find all items whose templates are a sub-template of a particular parent template
  + I can’t think of an example where I’d want to filter by a general template, but exclude any items defined by a more specific template
  + When a Template filter is defined, any items matching children templates will also be selected
* All XML documents sent to a Browser from the Client should be linked to a static master XSLT stylesheet
  + The master XSLT stylesheet will dynamically load the needed template-specific XSLT stylesheets
* It should be possible to create a completely different look and feel for the site
  + Template-specific stylesheets should be stored in separate items than the Templates
  + The master stylesheet should store the data linking templates to template-specific stylesheet information
  + A Theme should be encapsulated by a single item.
    - This Theme item should store stylesheets for every template
    - It would be good if the stylesheets could be edited in the same way any other item is
    - There should be an item for each stylesheet that stores the stylesheet itself within the <document>
    - The stylesheet should still be accessible as a file through a url. This file will be generated dynamically from the data in the XSLT Stylesheet Item
* User should be able to create their own Theme Items
  + On creating a Theme item, subitems should automatically be created for each Template that currently exists in the site
  + The automatically generated Stylesheet items should simply copy the Template-specific stylesheets of the default Theme
  + There is still a question about what happens when a new template is created
  + There needs to be a default way to deal with item data if there is no stylesheet available for that item’s template
  + The default representation of an Item could be a simple table with a row for each field, or a set of headings and paragraphs for each field.
  + Each Theme should define its own default Stylesheet for dynamically handling unknown data
  + The default stylesheet might indicate on the resultant page that no stylesheet was found for this item’s Template
* When a User creates a new Template they should be reminded that they need to manually create Theme specific stylesheets for that Template, otherwise it will use that Theme’s default stylesheet.

# Tue 2-Feb-10

* I need to be able to do a query that returns the subitems of an item with the data for the Summary View for each subitem
  + It wouldn’t be practical to manually figure out which fields are required for each subitem from the Client end – this would take far too many API calls
  + I want to keep the functionality of the site separate from the API, but Templates are concerned with the data structure of items so it would be appropriate for the API to automatically handle the parsing of templates
  + When doing a query on the API, it should be specified whether Summary or Document view is required
  + If a 3rd party application only wants a specific field from many items, it wouldn’t be too inappropriate to make them parse the full Document view of each item
  + Alternatively I could allow another query method which specifies a specific field or fields to pull. This would be easy to implement later so I don’t have to worry about this functionality immediately
  + Fields that don’t appear in the Summary or Document view will not be available outside the API
* I want to be able to record data about how successful the site is at pulling notice to popular items
  + I could go back to recording all View Events, but there are many disadvantages to this
  + I could enforce a system of forcing links to go via an internal link that records when the link is clicked.
    - This would allow statistics to be recorded about how many people go to particular product web pages from ZuthaNova.com
    - This would be very difficult to maintain and would be complex to implement, though
  + I could rely on Server logs to glean data about traffic
  + I could rely on 3rd party data about internet traffic trends (I’m not sure how this would be gleaned in this case)
  + I could suggest that companies record their own statistics about where their customers were referred from and request that they publish the results to show how much traffic was sent from ZuthaNova.com
* When querying the API, results need to be pulled with worth data about the item Owners
  + In order to be efficient, pulling data about the Owners of each item in a result set must be done automatically by the API or too many Client->API calls would be necessary
* I need to design a query language that allows me to pull all the needed data for a page in a single query
* An API query can contain multiple <query> elements within the request
  + The resultset will be the results of each query appended to each other in the order of the given <query> elements
* API queries should specify which fields should be returned within a <fields> element
  + The <fields> element should contain a list of <field> elements which specify a field to return
  + The specification for accessing fields may not match the actual data structure
    - E.g. a <summary> field can be selected to be retrieved and the API will dynamically pull the appropriate fields from the <document> element to create this virtual field
    - Fields relating to the author of the item can be chosen even though these fields must be pulled from elsewhere in the database
  + <field> element has optional boolean attributes @inDocument and @include\_attributes, both of which default to 0
    - @inDocument indicates that the given field name is Template specific and appears in the <document> element of the item
    - @inAuthor indicates that the given field points to data to be pulled from the User item of the owner of each result item
      * I may abandon this attributes in favour of requesting owner fields with a prefixed “owner\_”
    - @include\_attributes requests that attributes such as @edit\_level and @levelup\_gap be included in the returned data
  + The “@ID” and “@template” fields never need to requested as they are always sent as attributes with the <item> element that wraps each result
  + Instead of using attributes @inDocument and @inAuthor I could allow a basic Xpath syntax: “document/first\_published” or “owner/avatar”
    - This syntax is much more extensible and would allow pulling data from deeper ancestors of <document>
    - Data within <document> or within <owner> should be returned inside so-named elements
  + Requesting the field <owner> should return an element containing <username>, <avatar>, <worth>, <level>
* If <start> and <end> are missing from a Query, <start> should default to 1 and <end> should default to some appropriate normal amount of results to return like 50
* API <filter> syntax
  + The <query> element should only contain one top level <filter> element
  + <filter> element can contain nested <filter> elements
  + <filter> has an attribute @type
  + @type=”and”
    - Must contain more than one child <filter> element
    - All child <filter> elements must apply
  + @type=”or”
    - Similar to “and” – functionality self-explanatory
  + @type=”not”
    - May contain one or more child <filter> elements
    - NOT is effectively like a NOT filter with an AND filter inside it because all children of the NOT filter must be false
  + @type=”parent”
    - May contain a string representing the ID of the parent that all results must have
    - May contain another <filter> from which the allowable parents will be taken
  + @type=”template”
    - Must contain a string representing the unique name of a template
    - All results must be of this template or of child templates of this template
  + @type=”templateID”
    - Same as “template” filter except the template is defined by its ID instead of its name
  + @type=”Yahoo”
    - Must contain a valid Yahoo search string
    - Results are filtered by the results of the site wide Yahoo search
  + @type=”title”
    - a string to match to the <title> element of items
    - requires the @extent attribute
  + @type=”level”
    - Must contain an integer optionally preceded by > or <
  + @type=”worth”
    - Must contain a number optionally preceded by > or <
  + @type=”owner”
    - A string to be matched exactly to the owner of result items
  + @type=”id”
    - An ID string
    - Only one item can ever match this filter
    - To produce more than one result with this filter type, an OR filter must be used in conjunction with it
  + @type=”modifed\_before”
    - An ISO formatted timestamp
    - E.g. 2009-11-23T13:33:00.000+10:00
    - I will possibly also allow a Unix timestamp, which will probably remain the internal format of storing timestamps
    - Date\_modified must be before this date
  + @type=”modified\_after”
  + @type=”created\_before”
  + @type=”created\_after”
  + @type=”field”
    - Must contain a second attribute @field=”publication\_date” (for example)
    - This type of filter (as well as @type=”title” filter) requires another attribute: @extent
    - The content of the element is a string to find inside that item (unless @extent=”presence” in which case the element should be empty)
  + @extent=”contains”
    - The field must contain the search string somewhere in it
    - No ancestor elements of the target field will be searched, result must be found in the text node directly inside the target field
  + @extent=”equals”
    - Same as above, but the given field must exactly match the search string
  + @extent=”presence”
    - <filter> element with this attribute should have an empty text node
    - The field specified by @field must be present
* Events should not be considered children of items they are related to
  + An event’s <parents> should be the “Events” item and the “User Actions” item
  + Events should not be linked to the item they concern through the <parents> field
* A <query> element in a Query request should contain an attribute @type which is either Event or Item
  + The results of a query can either be Items or Events but not both
  + Since multiple <query> elements can appear in a single Query request, it is possible to have Events and Items in the one result set, however
* When pulling the subitems of an item, only items of type Comment need to be excluded
  + Events will not appear in the resultset of a query that finds all items with a particular parent
  + Comment items should appear in a separate page than Subitems
* The Comments page should have the option to rank items by date or by worth
  + Comments should be ranked by date by default
* The Subitems page should be ranked by worth and not have any other options
  + Perhaps I’ll secretly allow ranking the subitems by something else using the “order” GET field
* The Search page should allow ranking by date, worth, Yahoo or Bing page rank and others
* If anyone wanted to view the subitems of an item ordered by something other than worth, they would have to do an advanced search
* All descendants of <document> should be under the same namespace
  + When searching within a specific document field (e.g. field:author:”J. K. Rowling”), all items containing that field should be searched
  + Fields should not be linked inseparably from their template
  + Some field names are likely to appear in multiple templates (like “author”) and it should be possible to search all items with this field, regardless of which specific template they are
  + A specific template can always be specified if so desired
* Caching
  + The client should store the HTML page of every URL that has been requested
  + The time since that URL was last requested should be stored with the cached data
  + There should be a default time limit within which a cached file will be returned instead of re-querying the API
  + After the time limit is passed, the next time that URL is requested, the Client will query the API for that data again
  + A User should be able to override the default caching time with their own setting
  + If a user is logged in who has changed their setting for cache time limit, then the Client will change its caching behaviour
  + This affects everyone of course. If a user who has set their cache time limit to 0, say, views an item, then the data will be updated for everyone
  + Images and other files like XSL and CSS files should be cached for a much longer period
  + When an Image, XSL, CSS or other file is changed, the API should send notification to the Clients (the private ones, not 3rd party ones)
  + It is the caching of commonly accessed files like CSS, XSL and image files that the most time will be saved
* A periodic script on the API should compile a list of all items (or parents of items) changed since the last time the script ran, and send this list to each Client.
  + This way the Client will end up with a comprehensive, up to date cache of xml and/or html documents of every item.
  + The Client should use the list to mark cached items as out-of-date, or maybe simply delete cached items that match any in the list. These deleted items will be replaced next time they are requested by a user.
* Fields like <publisher> should be linked to a Publisher item.
  + This will allow automatic pulling of the publisher name and generation of a link to the publisher item for presentation on the webpage.
  + It will also ensure consistency, because a particular publisher will be unambiguously identified by a unique ID.
  + Queries can then be made to find all books by a particular publisher and there won’t be need to worry about errors in spelling of the publisher name in some items and so forth.
  + Actually allow a choice between <publisher> and <publisherID> fields. It may be too much fuss to generate the publisher item just to add the publisher field to a Book item. In that case a simple text field can be used until someone gets around to linking the <publisherID> field to a Publisher item.
  + When the <publisherID> field is filled, the <publisher> field must be deleted. The <publisher> field will then be artificially made available by pulling publisher name from the linked Publisher item.
  + I could even have the publisher item generated automatically if it doesn't exist. A javascript generated picklist could appear when choosing the Publisher, to prevent too many duplicate publishers with variations in name.
  + There would still need to be a way to merge Publisher items. If the <publisher> field was indexed then all items that use the superseded publisher could easily be updated during the merge action.
* Maybe it wouldn't be so bad to delete items. the data would be stored in Event items, so the action could be reversed.
  + Deleting an item would look no different from the outside to making it invisible to the public
  + I will probably need to make some items invisible for copyright reasons, but I may not choose to delete them.
  + I can at least take it on a case by case basis as to whether I actually delete, or merely hide an item.

# Wed 3-Feb-10

* Do not store a user’s financial data in the site
  + I do not trust my ability to maintain the security of my site enough to let people trust me with their credit card details and such
  + Actions like Donations (which are pretty well the only financial action that will occur through the site) should be done through a 3rd party body like Paypal or Google Checkout.
* After a request, the result xml file should be stored in the cache
  + If a url has the name of an item wrong, or the case of some letters wrong, the request will still be sent
  + When the correct url is ascertained after the API request, it is compared with the actual url used, and the browser is redirected if the actual url is wrong
  + After the redirect, the cached result of the API request done on the incorrect url, should be available, so bareley any effort should be wasted
* When a page is to be converted to HTML at the server end, both the original XML document and its HTML transformation should be stored in the cache
* When a page is requested in HTML, both the HTML and XML cache should be checked
  + If an up to date HTML version of the page is available, just return that
  + If only the XML document is available in the cache, convert that to HTML and return that
  + If neither the HTML nor the XML version of the page is available in the cache, a new API call must be made
* I’ve decided the API should be written in C++
  + C++ is the native language for the Berkeley DB XML API
  + C++ will be more reliably supported by Berkeley DBXML than PHP (PHP 5.3 doesn’t work with DBXML at the moment)
  + The only IO mechanism for the API is raw XML HTTP Post requests. This should not be too hard to do with C++
  + C++ might not be an ideal language for web development, but the extent to which the API is a web app is minimal because it simply responds to a simple raw xml input with raw xml output. The fact that the communication is via HTTP Post requests is immaterial.
  + Learning C++ will be useful, and will get me ahead in my IT course at Uni as well
  + C++ is fast
* My attempts to squeeze all site functionality into some subpage of an item is beginning to become cumbersome
  + Now that I have clearly separated the data structure (the API) with the site interface (the Client) it would not be undermining my design model to have some pages on the Client which are not subpages of an item.
  + In fact, some functionality is simply not appropriate as a subpage of an item since it is specific to the interface and not the data
  + Search, Login, Mass Edit, Item Creation – all these functions should be separate from a specific item
* New url syntax:
  + Item Independent pages
    - Zuthanova.com/Login
    - Zuthanova.com/Register
    - Zuthanova.com/Search
    - Zuthanova.com/AdvancedSearch
    - Zuthanova.com/MassEdit
    - Zuthanova.com/CreateItem
  + Item specific pages
    - Zuthanova.com/items/4THB1/Harry-Potter/Document
    - Zuthanova.com/items/4THB1/Harry-Potter/Subitems
    - Zuthanova.com/items/4THB1/Harry-Potter/Comments
    - Zuthanova.com/items/4THB1/Harry-Potter/Edit
    - Zuthanova.com/items/4THB1/Harry-Potter/History
    - Zuthanova.com/items/4THB1/Harry-Potter/Investments
* The top level (item independent) pages will be sent as html pages
  + These may pull some data from the database. For example, a list of Templates to put in a combo box on the AdvancedSearch or CreateItem page
* I’m beginning to question why I ever need to send pages as xml with linked XSL stylesheets
  + If a developer desires raw xml data, he can use the API
  + Keeping the client side XSLT would be useful for anyone wanting to learn how XSL works so they can create their own Themes
  + It may be that using client side XSLT would reduce load on the server
    - The majority of pages viewed will be subpages of items, which are the pages that can be sent as xml
  + A lot of the html in a page will be stored in the XSL stylesheet and this only needs to be loaded once by the browser
    - There is almost no redundant data in the XML documents, whereas there may end up being a lot of redundant data in the full html pages
  + I think what was bothering me was the inconsistency of having the item independent pages always sent as html, but sending item specific pages as xml
    - Having thought about why client side XSLT makes sense, this no longer seems incongruous to me
* There should be links on every page to the top level pages (except MassEdit)
  + When clicking on a link from an item subpage to the CreateItem page, the CreateItem page should automatically have added to the <parents> field, the item from which the user pressed Create Item.
  + MassEdit page is linked to from the Search results page

# Thu 4-Feb-10

* Realtime information needs to be added to the header of every page
  + Whether a user is logged in and, if so, which user is logged in
  + A dynamically changing, user targeted message, like:
    - You are using internet explorer 6, which is not a standards based browser. This site will look and function better if you use a standards based browser like Mozilla Firefox or Google Chrome
    - You can earn Zuth by donating real money to a charity or cause you believe in. Click here to find items that you can donate to.
    - ADMIN ALERT: You have been accused of altering someone else’s work without justification. Click here to see the complaint. If you do not justify your actions you may risk demotion, or at the very least jeopardise your ability to reach higher privilege levels.
  + The set of messages to be cycled through should be loaded from the API on login
* The real-time information needs to be added to the xml document just before it is sent to the browser.
  + It no longer seems practical to cache both html and xml versions of pages
  + The xml document before the real-time information is added, should be cached always
  + If ever the html needs to be generated server-side, the raw xml document should be created or taken from cache, the real-time information added, and then the document transformed with XSLT before being sent to the browser as html.
* I could send image data in base64 text format with the xml document
  + This would make the design easier, potentially
  + It would be diverging from the norm, however, and I want to avoid that since doing strange things could lead to unforseen compatibility problems
* I could store images inside the xml database
  + Image data could reside in the <imagedata> field of a Image item
  + When retrieving a set of items with the Client, the image data for every owner avatar could be retrieved with a single query.
  + The image data would then be stored in the cache
  + The xml document sent to the browser would contain only links to the avatar images
  + The browser would request each of the avatar images from the Client
  + The Client would already have retrieved all the images and stored them in its cache so it can return them all quickly.
  + The disadvantage of this is that the Client then doesn’t take proper advantage of caching.
    - By requesting all avatar images every API call, it is potentially making redundant calls because it may already have many of the Avatar images in cache
    - Therefore I won’t retrieve all avatar images with each Client query
    - I may still store the image data as base64 data inside Image items though
* Caching is going to be more complex than I thought because I have to take into account user settings and get variables defining the document
  + I could cache only the default document. I.e.
    - Start = 1
    - End = 50
    - Order = worth
  + To make things easier, it would be best if any user settings that would affect the structure of a document were passed through GET variables
    - Apart from the header (with messages and login info), a document with exactly the same url should look exactly the same
  + I had intended to allow settings such as
    - Setting the date format
    - Changing the number of decimal places of seconds to show in dates
  + Settings that affect page appearance that cannot be put in GET variables should be avoided
  + Even selecting direct HTML output should be defined by GET variable “&output=html”
* Query field selection syntax
  + Top level fields like “title”, ”date\_modified”, ”worth” etc. Are selected with syntax <field>title</field>
  + Some fields, like “owner”, link to other items. Such fields will allow a syntax for getting data from the linked items
  + To access data inside the owner item, use syntax <field>owner/worth</field>
  + Only the data inside the <document> element of the linked item is available using this syntax
  + Process for looking up: <field>owner/worth</field>
    - look in the current item for the <owner> element
    - get the id in the <owner> element and look up that item (the owner’s user item)
    - inside this item look for an element called <worth> inside the <document> element
    - return this data with syntax: <owner><worth>4325</worth></owner>
    - i.e. exclude the <document> element from the hierarchy of the returned data
  + This syntax can be used to look through multiple layers of items
  + To get data about the avatar Image item of the owner of the current item
    - <field>owner/avatar/filename</field>
    - This looks up the owner item, then looks up the Image item linked to by the <avatar> field, then returns the <filename> element from the <document> element of the avatar Image item
* If I can only access data inside the <document> element of linked items (as in “owner/avatar/filename” syntax query), then I can’t access the <title> element, but that will be a crucial query target for such situations
  + It makes most sense to just access fields where they are found.
  + So an avatar’s filename would be found at “owner/data/avatar/data/filename”
  + I’ve replaced the internal <document> element with <data>
  + <document> and <summary> are both virtual elements that automatically pull the relevant fields by looking up the item’s template
  + <document> and <summary> do not allow access to individual fields inside them. For that, the <data> field must be used, since this field contains all the fields in either <summary> or <document>
* The “user’s items” item should be a Search item, which finds all items that have that user as their owner

# Fri 5-Feb-10

* I had considered making the <title> field independent of the item data. E.g.
  + User items would be named <title>user: barneyii</title> and the actual username would be a separate field in the <data> element
  + Image items would be titled like <title>image: frog</title> and the filename would be stored in the <data> element
* For fields that link to another item (like “publisher”), I need a convention for returning the id and name, rather than just the item
  + If some items were titled like: <title>publisher: Barallier Books</title>, then for getting the data for a named link to the publisher I would need to return 3 fields: <id>, <title> and <name>
  + I would then need a convention for storing the actual <name> field so that it can be accessed automatically for any item
  + This is like having another <title> element which defeats the purpose of the title element since titles like “image: penguin” could be made by the Client from a title like “penguin” or “Barallier Books”
  + Therefore <title> will be a field that is used programmatically
    - By this I mean that requesting fields that are links like the <publisherID> field, should return something like <publisher id=”5H1N”>Barallier Books</publisher>
    - This format should be the convention for returning id based fields (like <parent>, <child>, <publisher>, <template>, <owner>)
      * The automatically generated <breadcrumbs> field should return a list of <breadcrumb> fields with format: <breadcrumb id=”RV9WA”>Browse</breadcrumb>
      * The <owner> field should be changed to an ID based link
  + The Client can take the <id> and <title> fields and make a link to that item
  + Fields like <publisherID> can be resolved into a link to the publisher item with the name of the link simply being the actual publisher the way it would be written if it wasn’t a link. i.e. “Barallier Books”
* @id and @template should be turned into normal elements <id> and <template>
  + Attributes should be reserved for things like permissions and visibility
  + Actually perhaps id should be left as an attribute
  + Since id is an attribute in so many cases I should attempt to make it always seen as an attribute
* If even fields that link to other items with an id used the id as an attribute it would make it easier to identify such fields. i.e.
  + <child id=”U8F” />
  + <publisher id=”B2C” />
  + <template id=”TB2” />
* I’m really liking the design principle of linking to other items and automatically expanding the data when it is requested
* The Multi Part Document template table of contents design:
  + <toc> element only has direct children, but with a recursive links system a multi tiered toc can be generated
  + <toc> has <part> elements as children
  + <part> elements simply have an @id attribute which points to a Document or another Multi Part Document
  + If the <part> links to a Document its toc is generated from the headings in that document
  + The Client would turn the headings of the document into page bookmarks (accessed by #bookmark\_name at the end of the url)
  + If the <part> element links to another Multi Part Document, then the toc of each linked Multi Part Document is merged into a larger toc in the parent Multi Part Document
* To scale up the capacity of the API I could implement a system of API mirrors
  + The central API would have to receive any action requests
  + The events it creates could then be sent on to mirror APIs which would perform those events as well
  + There would need to be a way of verifying the consistency of the mirror databases
    - if all events that modify store the old value as well as the new (as in editing items or changing settings), then it might be possible to detect if an event is received that is dependent on another already taking place
    - a separate process could be used to check that all events had been received, but if the above was true, this process would not be time critical.
    - Events could be missed for a while and it wouldn’t risk the database being corrupted by a later event getting through that depends on the missing earlier event
  + The mirror APIs would be used only for querying
  + It is querying that needs to happen fast.
    - I wouldn’t mind that much having to wait a bit when creating an item or investing, since I would only be doing that occasionally
    - I would mind if I was delayed in loading a page, so this aspect is what needs to be fast
* Perhaps permissions on editing items should increase as the item level increases
  + I don’t want it to be possible for high level items to have important information changed by users who are less accountable (low level users)
  + While it is possible to alter the permissions level on parts of an item when it is deemed vulnerable to manipulation, this requires a user to notice
  + It would be very likely, if I relied on users to find fields of high level items that should be protected, that many would escape notice
  + It would end up being very easy for someone with the intent, to level many users up to about level 3 and use them to do a lot of vandalism
  + There should be the option to define permission by level gap or absolute level
  + Some fields like <level> should only be definable to level gap
  + There would be cases when permissions would need to be defined by absolute level
    - An important category item like Browse, may not have a high worth or level, but it would be important that low level users cannot vandalise it by changing its name or description or by adding rubbish children.
  + Actually both @plevel and @pgap should be usable in the one element
    - Sometimes it would be desirable to stop low level users editing an item when the item is on a low level, but the level required to edit it should start increasing once the item reaches a certain level
    - Whichever attribute indicates the highest required level is used.
  + @pgap can be negative. In fact its most common value should be @pgap=”-1”
    - This is because the owner of the item will always be at minimum 1 level behind the value of the item (since the owner earns 10% of item revenue)
    - A user who creates a popular item should, by default, be given the privilege of editing it
    - Actually a user may not be at least 1 level behind the item, but his worth will at least be the equivalent of 1 level behind, even if he has not received the level yet.
* The kinds of items (defined by their template) that can be added as children to a particular item is defined by the parent’s template
* Changing the permission level on the <children> element of an item affects the ability to create the kinds of items available for that item.
* The ability to create Comment items is not specified by the template
  + Comment items are always creatable by any user for any item
  + The permission level on <children> element does not affect that ability to create Comment items
* Some templates, designed for discussion only, may not allow any subitems, only comments
  + In this case, the default page to open when opening this item should be Document or Comments
* I don’t think I need a separate Comments page
  + Document page should be the one that has comments at the bottom, just like the way current internet articles do
  + The Subitems page should be the default for things like Category items
* Many items that are likely to have Subitems, may not have large Documents so the whole <summary> and <document> idea may be unnecessary, but I don’t think I can rely on it.
* The <title> field of an item is what it would be named as a link
* I don’t have to worry about special characters in the <title> field.
  + Special characters will be converted to dashes “-“ for the url
  + I can name Image items with their extension. i.e. <title>apple.jpg</title>
  + This is the most appropriate name for an image item since it is the name that would be shown in a link to that item

# Wed 10-Feb-10

* The <title> field could be pulled from template specific data
  + The User template would define that the field <username> is used as the item name
  + The User template would also define that <username> is returned when <owner> is requested as a link from another item.
    - <owner id=”5GD”>barneyii</owner>
    - where barneyii is the <username> of the “5GD” User item
* the data structure does not really require a <title> for every item in order to function
  + Comment items may not require a <title>
  + The Client could generate a title for comment items which have no specified title.
    - An unnamed comment item under “Harry Potter” might be automatically named “Re. Harry Potter”

# Fri 12-Feb-10

* When selecting <field>s to show in results, if <field> does not exist for a particular result, it is simply not returned – no error occurs.
  + This is to make it simpler to get results from multiple item types
* In the History view of an item, each action listed should have a “reverse” button next to it
  + The “reverse” button should only be visible to users with permission to use it
  + Reverse button should only appear (or at least only work) if the <new> value of the changed field is the same as the current value
* Comment items should have default permissions:
  + @pgap=-1
  + @plevel=0
  + @pgroup=private
    - This means only the owner can edit
  + A user should be able to edit his own comments
  + I can’t think of a reason why comments should be editable by anyone except the owner
* Many items should have default permissions: @plevel=3
  + This is to protect items against vandalism by default
* @pgroup could eventually be used to create sections of the site dedicated to particular groups, like companies
  + It could also allow items to be worked on collaboratively without the danger of vandalism from users outside the permission group
* All permission conditions (pgap, plevel, pgroup) must be satisfied in order for a user to be allowed to edit an item
  + I might make a @pgroup\_override attribute that specifies a level at which the @pgroup condition can be ignored
  + One of @pgroup or @pgroup\_override must be satisfied in order to edit an item
* Virtual fields
  + Breadcrumbs
  + Document
  + Summary
  + Allowed\_children
* Every user stores messages relevant to them
  + A certain set of messages are assigned to every new user (generally help messages)
  + Users can dismiss messages, which removes them permanently from their User item
  + Messages are cycled through in a message bar at the top of every page in the website

# Sat 13-Feb-10

* Allowed templates field
  + The virtual field “allowed\_children” returns Comment (and any other always allowed types), plus the types listed in the <allowed\_children > field of the Category item
  + Templates may define <allowed\_children > inside <data>. This determines the default value of <allowed\_children > for that item type
* A <permissions> field in Template items allows for specifying the initial permissions of fields for items of that type
* A category item is created with @plevel=”3” on <allowed\_children >, but it has no default allowed children
  + In order to create a category for which subitems can be created, a user must be at least level 3
  + <allowed\_children> must be manually specified for Category items
* An “Argument Map Node” item would have <allowed\_children> = “Argument Map Node”
  + This would be specified by the <default\_allowed\_children> of the “Argument Map Node” Template
  + The template would also specify high default permissions for editing <allowed\_children> of an “Argument Map Node” item
* I could make an <auto\_children> field in Templates that specifies a set of children to create automatically.
  + The field would list a set of templates
  + The kinds of templates that would work here would have to be really specific templates, designed for this purpose
  + E.g. a “Reviews Category” Template, which would define within the Template definition:
    - That only Review items can be children of this Category
  + For auto-generated children, I would need a way to automatically generate names for them.
    - The auto-name should probably be specified in the <auto\_children> field
* Investment events should be numbered with respect to where they lie in the list of investments for that item
  + This is to make it easier to detect if an event has been missed when Events are distributed to Mirror APIs.
* I need a way for link fields like “author” and “publisher” to automatically add relevant parents to the item containing the links
  + I need a field that would go in a “publisher” or “author” item (x) which specifies which parent item (y) to add to the linking item (z)
  + For a Person item (x) the linking item (z) might be of type “Book”. In this case, the parent item (y) should be a Category subitem of (x) such as “Books written by <person>”
  + The “Books written by <person>” item (y) might be created via <auto\_children>. In this case, the field in (x) that specifies (y) as the auto-parent for Book items linking to (x), must be automated also.
* It would be handy to have a template dedicated to “Authors” so I can make special functionality for the Author role, however, many people hold other roles than “Author” and I don’t want multiple items representing a single person.

# Tue 16-Feb-10

* I want to be able to change <default\_allowed\_children> in a template and have this change affect all items that use the template
  + I could do this by running an algorithm whenever the <default\_allowed\_children> field of a template is updated to update the <allowed\_children> fields of all items that use the template
  + I could force all items to reference the <default\_allowed\_children> field of their template and add the Types listed in this field to those in the item’s own <allowed\_children> field to come up with the final list.
  + The 2nd option would only allow adding to, not removing from the <default\_allowed\_children> list.
  + This is not a problem if <default\_allowed\_children> is only used for situations when certain children Types should always be allowed
  + Actually in most cases, the types of children allowed should be defined by the template, not individual items
  + Category items will always have <allowed\_children> defined at the individual item level. Hence the issue of updating their <allowed\_children> field when their template is changed, is not relevant.
  + I could allow adding and subtracting from the Template defined <default\_allowed\_children> list
  + <allowed\_children> could contain:
    - <allowed\_type id=”4B” />
    - <disallowed\_type id=”BV4” />
  + I can now allow a general Template like Document, but disallow specific sub-templates like Review, so that the allowed Types are all Document Types except Review.
* I could make Templates use an inheritance principle
  + If I put Comment in the <default\_allowed\_children> of the Template Template, then it would apply to all Templates, and thus all items
  + If I did it like this though, it would be possible for individual items to disallow Comment children, which I don’t want
  + I could allow an optional boolean attribute in <allowed\_type> and <disallowed\_type> like @can\_override=”0”, which would prevent sub-templates or items from overriding the <allowed\_type> or <disallowed\_type> of the Template.
  + When making <allowed\_type> elements in <allowed\_children>, a particular <allowed\_type> should only be accepted if that Type has not been defined in an ancestor Template with @can\_override=”0”.
  + I can now specify that all items should be allowed Comments as children by putting <allowed\_children id=”3A” can\_override=”0” /> in the Template Template
* I need a Reverse\_Predicate Type
  + Reverse\_Predicate item examples:
    - Books written by J.K. Rowling
    - People born in Australia
  + Reverse\_Predicate Items are associated with link fields (predicates) in that a Book item with an Author field pointing to the Person: J.K. Rowling, will be made a child of the Reverse\_Predicate item: Books written by J.K. Rowling.
* I would be good if fields were defined unambiguously
  + “Title” for a book should not be confused with a nobleman’s “Title”
  + I could allow for URI definitions of fields (i.e. predicates) in Templates
* I want to eventually link my database with other semantic projects like DBpedia.org and LinkedData.org
* Each Reverse\_Predicate such as “Books Written by <author>” would have multiple instances like “Books written by J.K. Rowling”
* I need a more flexible system for defining <title> field, especially for People
  + “J.K. Rowling” is a penname, but should probably be the title of the item that represents her
  + Similarly “J.R.R Tolkien” should be the item title
  + I want to be able to glean full names like “John Ronald Reuel Tolkien”; first names i.e. “John” alone; pen names/most-commonly-known-as names such as “J.R.R Tolkien”
  + I can solve this by making it possible to override the virtual <title> field with a <title> field defined in the item itself
  + Both the template defined and locally defined <title> field can reference other fields such as <firstname> and <lastname> to construct the <title> field dynamically
  + For famous authors that are better known for their pennames like “J.K. Rowling”, “J.R.R Tolkein”, “Robert Jordan”, their items can have the auto <title> field overridden locally with <title><field>penname</field></title> such that the field <penname> is used as the item title instead of “<firstname> <lastname>”
* Templates should have a <definitions> field which defines metadata about fields used in the template
  + The <schema> field defines the structure of the document with XML Schema
  + <definitions> defines such things as
    - links to external predicate definitions
    - allowed linked item types (if the field is a link)
    - initial editing permissions
    - a brief description of what goes in the field
  + <allowed\_types> has children <allowed\_type> or <disallowed\_type>
    - This allow for exclusion of Templates that fall under an allowed parent Template
  + Not all fields in the template should need to be defined in <definitions>
    - For some fields, everything might need to be known about them is defined by the schema
* There are items which I would be more inclined to call Category items than Reverse\_Predicates (a name I don’t really like), but which require similar functionality to the Reverse\_Predicate idea
  + The item “Fantasy Books” should automatically be added as a parent to Book items with <genre>fantasy</genre>
  + The item “Authors” should automatically be added as a parent to Person items with <occupation>author</occupation>
  + The Occupation item “Author” is distinct from the “Authors” item that has Author items as subitems.
  + The Occupation item “Author” needs to define that Person items with <occupation>author</occupation> need to have the “Authors” category item added as a parent
* I need a new Item type: the “Role” item
  + A Person item can have multiple Roles such as Author, Parent, Board Member, President etc.
  + The Person Template defines the available Roles for that item type
  + A specific item such as a Person defines which Roles are relevant to that item
  + Inside the <roles> element of an item are <role id=”5K” /> elements
    - Inside <role> elements are <autoparent> elements which define for each item Type (object) that might link to this item (subject), which Predicate item should be automatically added as a parent to the object.
    - E.g. <autoparent object=(book) predicate=(books by J.K. Rowling) />
* Predicate items should be divided finely into sub-types
  + There should be a Template for “has\_author” and “has\_occupation”
    - “has\_author” may have instances such as “works by Robert Jordan” or “works by Steven Hawking” (which would contain scientific papers as well as books)
    - “has\_occupation” may have instances such as “authors” or “lawyers” or “physicists”
  + Some Predicate items like “has\_author” may be broken down further to contain only items of a specific type. E.g.
    - “book\_has\_author” which might have instances such as “books by J.K. Rowling” or “books by Robert Jordan”
    - “series\_has\_author” which might have instances such as “book series by Robert Jordan”
  + Instances of general predicates like “works by Robert Jordan” should specify the relevant instances of type-specific predicates like “books by Robert Jordan”
* Item specific <roles>
  + For the Person item “Robert Jordan”: <role id=(author) predicate=(works by Robert Jordan) />
* A Role item would specify the item types that can have that Role
  + The Role: Author applies to Person, Organisation, Software etc.
  + This means that link fields can have the relevant Role item specified and the allowed Item types for that field can be pulled from the Role item
  + It may be prudent to allow defining of allowed\_types explicitly in case the Role concept doesn’t apply in some circumstances
* A Role item would be associated with a general Predicate item
  + The Role: Author would be associated with the general Predicate Template “has\_author”
* When a Book item (say Harry Potter and the Chamber of Secrets) is created with <author id=(J.K. Rowling) />
  + “J.K. Rowling” item is checked for the presence of a <role id=(author)> element
    - If <role id=(author)> exists, the value of the @predicate attribute is taken from this <role> element. In this case the predicate should be the “has\_author” item: “works by J.K. Rowling”
      * “works by J.K. Rowling” is added as a parent to “Harry Potter and the Chamber of Secrets”
      * “works by J.K. Rowling” is checked for a link to the type-specific predicate for the type: book. In this case that would be “books by J.K. Rowling”
        + If “books by J.K. Rowling” exists, it is added as a parent to “Harry Potter and the Chamber of Secrets”
        + If not, then the “has\_author” Template would be checked for a sub-Template: “book\_has\_author”

If “book\_has\_author” Template exists than a new item of this type is created: “books by J.K. Rowling” (and is added as parent to H.P.)

If “book\_has\_author” Template doesn’t exist, it indicates that the predicate “has\_author” does not need to be broken into categories by the Type of subject using the predicate.

* + - If <role id=(author)> does not exist then this <role> is added to the “J.K. Rowling” item
      * The Role: Author is referenced for the relevant predicate Template (“has\_author” in this case)
      * A “has\_author” item is created, in this case called “works by J.K. Rowling” (and added as parent to H.P.)
* I need a better name than Predicate for items like “works by Robert Jordan” which are based off Templates like “has\_author”
  + “has\_author” is a predicate, but it is not accurate to describe the item “works by Robert Jordan” as a predicate

# Wed 17-Feb-10

* How to automate categories
  + I could use the convoluted method I designed yesterday, with Roles and a hierarchy of Predicate Templates and instances
    - This method just seems too messy and convoluted – I don’t like it
    - I like the Role concept, but I think it will be useful for other reasons even if I don’t use it for automatically filling categories
  + I could make Category items Search items that pull their children in real time by doing a stored search
    - This would mean children of categories would be absent from the Hierarchy, so they would not appear in search results from an ancestor of the category
    - It would prevent manual addition of children
  + Category items could be Search items that save the search results each time the category item is opened
    - This method lacks “neatness”
    - After a Book item is created with a link to an author, the category item: Books by <author> will not be added as a parent to the new Book, until someone opens to category item.
    - Which user performs the necessary actions when someone opens a category item?
    - I want to avoid any editing of the database for View events (especially since View events don’t have to be associated with a User)
* Roles should define Role-specific fields
  + The Role: Author could define fields such as: <notable\_works>, <genres>, <influences>
  + When an item is given a <role> it opens up new fields to that item.
* There could be various different things that could happen automatically for linked fields (like <author>, <publisher>, <mother>, <parent\_company> etc.)
  + When a Person (A) has their <mother> field linked to a Person (B), Person (B) should automatically have a <child> field added linking to Person (A)
  + When a Book is linked to an <author>, it should be placed under the category items “works by <author>” and “Books by <author>”
  + When a Person is linked to <occupation id=(novelist) /> they should be added to category items “novelists” and “authors” (where the “novelist” occupation is a descendant of the “author” occupation).
* The concept of Search items is much more flexible than the Predicate item concept
  + With a flexible query syntax (the same used for the API), I can make Auto\_Category items that satisfy all the purposes I had thought to use Predicate items for.
  + An “Authors” item can search for all items for which the <occupation> field points to an occupation which is a descendant of the Author occupation.
* For many relationship links, it may not be suitable to design it as a one-way link like <author> and use an auto\_category for the specific->general link.
  + A Person could have a <father> and <mother> field
    - Should the mother have an auto\_category subitem pulling all Person items with her as the <mother>
    - I think it would make more sense for the mother item to have <child> fields
* Two way links:
  + <mother> ⬄ <child>
  + <member\_of> ⬄ <party\_member>
  + <member\_of> ⬄ <band\_member>
* There is a blurry line between which one-to-many links should be two-way links and which should be one-way links
  + <spouse> should definitely be a two-way link as it is one-to-one
  + <mother> ⬄ <child> should probably be two-way since it is neater to have a list of <children> within the item data than requiring navigation through a subcategory “Children of <mother>” in order to find the children items.
  + <member\_of> ⬄ <party\_member> makes more sense as a one-way link since the number of members of a political party could be arbitrarily large.
* For two-way links it would be desirable to automatically complete the other half of the link
  + They are going to be permissions issues to resolve here
    - A level 0 user could create a new Person and name his <father> as Albert Einstein
    - The permissions on one end of a two-way link need to affect the other
    - In order to add Albert Einstein as a Person’s <father>, one needs to have permission to edit the <children> element of the Albert Einstein item
  + Who is credit for the edit at the other end of the link
    - I see no reason why the edit of both ends of the link cannot be credited to the editing user
* If I tried to map friend connections I would run into a cyclic problem in the Hierarchy
  + <friend> ⬄ <friend> is a many-to-many connection
  + If <friend> was a field it would have to be a field for both Person items so there would be no cyclic issue in the Hierarchy.
  + If Person (A) had a subcategory “friends of <person>” containing a Person (B), and Person (B) had a subcategory containing Person (A), this would yield an infinite loop in the Hierarchy and my current data structure would not support this.
* I could change the Hierarchy dynamics to allow for infinite loops in a friendly way
  + Every time a child (B) is added to an item (A), the hierarchy under (B) has any instances of (A) removed from it.
  + This method should not hinder the ability to search all ancestors of an item.
  + All this does is removes redundancy
  + All items under (B) will still be searched in a query on (A) and vice versa
* The main purpose of placing all associated items in the one basket (<children>) is for the ability to search ancestors of an item
  + This does not mean there cannot be redundancy
  + Let’s say I want to link a company (A) to its partner companies
    - I can make a subcategory of (A) called “partner companies of <A>”, which contains the partner companies
    - I would want to describe these relationships semantically as well, though, and in fact it would make more sense if the “partner companies of <A>” category pulled its children from the semantic data
    - Company (A) could have a field <partner\_companies> which list the partner companies. Each partner company would also link back to company (A) in the same way
    - The “partner companies of <company>” categories would pull their children from the relevant <partner\_companies> field
    - This situation would create an infinite loop in the hierarchy, but this is no longer a problem
  + When searching ancestors of an item, what kind of relationships should define the items that are available for this search?
  + In general, subitems need to be more specific than their parent
    - It would not make sense to have the author of an item be an ancestor of that item
    - Even though the author is associated with a Book, it should not appear in searches under the book because it is not a more specific concept than the book.
    - If someone wants information relevant to the author of a Book, they need to find the author via the Book, and then search under the Author
    - Sideways relationships like <partner\_company> are more difficult
* When searching under item (A), the breadcrumb chains of the result items should make a course through the item (A)
* Templates store all possible fields, but some fields require a particular Role to be assigned to the item in order to be enabled
  + This will enable these Role-specific fields to be defined in the template schema and in the template’s stylesheet
* Sideways fields like <friend> and <business\_colleague> should not be part of the hierarchy. Such relationships will exist only through specific fields
* Roles define default auto\_categories to create, such as “works by J.K. Rowling”
  + The default auto\_categories should be easily modifiable and added to
* Within a <role> definition for an item, there should be a list of <relevant\_category>s
  + <role id=(author)>
    - <relevant\_category id=(works by J.K. Rowling) />
    - <relevant\_category id=(Books by J.K. Rowling) />
    - <relevant\_category id=(Articles by J.K. Rowling) />
    - <relevant\_category id=(Fantasy Books by J.K. Rowling) />
  + When a Book (A) is linked to a specific <author> (B) the linked Person item (B) has its <role id=(author)> element checked for <relevant\_category>s
    - Each of these <relevant\_category>s is checked to see if it’s conditions are satisfied by the Book item (A)
    - The “works by J.K. Rowling” category would specify only that children must have <author id=(J.K. Rowling) />
    - The “Books by J.K. Rowling” category would specify that children must have <author id=(J.K. Rowling) /> and <template id=(Book) />
  + Some categories will have conditions on multiple linked fields (Author/Template/Genre/Publisher)
    - “Fantasy Books by J.K. Rowling” would appear as a <relevant\_category> in the Author <role> of J.K. Rowling and the genre <role> of the “Fantasy” Genre item
    - Actually it won’t be practical to store all the <relevant\_category>s relevant to the “Fantasy” Genre item, because there could be an arbitrary number of category items that use the <genre> field in their filter conditions.
      * Perhaps this wouldn’t be such a problem
      * There’s nothing wrong with long lists
      * And I need a way to update the relevant Categories when the <genre> field of an item is changed (or when any field relevant to a category’s filter condition is changed)
* Autocategory item
  + children are populated only by a filter condition such as author=”J.K. Rowling” and Template=”Book”
  + the filter condition defines a query in the same syntax as the API query syntax
  + when the Autocategory is first created, its query is run and the results are added to the Autocategory’s children
  + for every link field (such as Author) used in the filter condition, a <relevant\_category> element is added to the relevant linked item (such as “J.K. Rowling”)
  + whenever a link field is altered in item (A) such that it points to item (B), the <relevant\_category>s in item (B) for which the conditions are met by (A), are added as parents to (A)
  + whenever an Autocategory item is visited, its query is run and the results compared to its existing children
    - if there are results that are not already children, these results are added as children
    - this is to allow for complex queries which may use static fields (like <age> or <birthdate>)
  + I need to be careful with who is allowed to create Autocategories and what exactly the scope of the queries can be, since I don’t want people creating Autocategories with extremely resource draining queries, or containing unreasonable numbers of items.
    - Someone could make an Autocategory that contains every item in the database. This could cause trouble.
* When an item (B) is added as a child to item (A), no ancestor of (A) may be a descendent of (B) or we will have an endless loop
  + With each such action, all ancestors of (A) must be removed from the hierarchy under (B), before the (B) hierarchy is appended as a child of (A).
  + This will require quickly obtaining a list of all ancestors of (A)
  + Perhaps I need another container which stores the parent “inverted hierarchy”
  + The inverted hierarchy would not have a single root
* There isn’t actually anything important about the structure of the hierarchy
  + All I ever need to know is an item’s children, descendants, parents and ancestors
  + Perhaps I could abandon the Hierarchy container and just store these 4 lists in every item
  + This might actually be more efficient than the hierarchy
    - For a given node in the hierarchy there may be a lot of redundancy in terms of duplicated descendants
    - I’m pretty sure that the Hierarchy must have more redundancy than lists of descendants in each item
  + The question is, will the 4 lists structure contain all the information necessary to update all the relevant parts of the database when a new item is created
  + I think the 4 lists structure will actually be much easier and simpler to implement
    - Item (B) is added to <children> of item (A)
    - Item (B) and all its descendants are added to <descendants> field of item (A) and all its ancestors
    - Item (A) is added to <parents> of item (B)
    - Item (A) and all its ancestors are added to <ancestors> field of item (B) and all its descendants
    - I think that’s it!
  + How do I make <breadcrumbs>?
    - I would have used the <parents> field anyway I think
    - To make <breadcrumbs> for item (A):
      * Pick highest worth <parent> of item (A) as item (B)
      * Pick highest worth <parent> of item (B) as item (C)
      * ...
  + If there were millions or billions of items in total, then the Top item at least might contain billions of <descendent> elements. This seems impractical.
  + The Hierarchy system does avoid some redundancy it seems; it incorporates the data about descendants of subitems into the data that describes the descendants of the parent
  + The hierarchy design may be useful if I want to find items that are a discrete number of generations beneath the base item
  + Finding all ancestors could easily be done by recursively looping through <parents>
    - This would be practical since the <ancestors> tree would be substantially smaller than some <descendants> trees could potentially be (such as for the root item)
    - A list of ancestors is needed when adding parents, to ensure no infinite loops are created
    - If ancestors are only needed on edit events, there is no problem if some processing is required.
    - It is for querying that speed is most needed, which is why the descendants structure must be computed and stored in advance as the Hierarchy.
  + Loops should be disallowed – they are messy and have no justifiable purpose
    - An item (B) is to be added as a child to item (A)
    - A list of ancestors of (A) is constructed recursively
    - Descendants of (B) are searched for ancestors of (A)
    - If any are found, the operation is disallowed

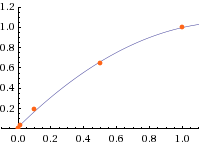
# Thu 18-Feb-10

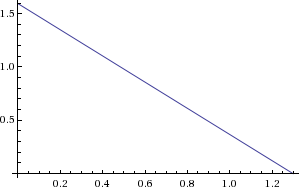
* I need a new permission type for parents defined by autocategories
  + I don’t want auto-parents to be manually editable, but I want them to be editable by autocategories regardless of who triggers the automatic action
  + @plevel=”auto” should be equivalent to @plevel=”admin” as far as users are concerned
  + Whenever a parent is added by an autocategory, that parent should be given @plevel=”auto”
  + Only the parent itself with @plevel=”auto” (the autocategory), can remove that parent
* Refreshing an autocategory needs to add new children and remove children that no longer satisfy the conditions
* In order to ensure autocategories remain up-to-date, any autocategories that depend on a field need to be refreshed whenever that field is changed in any item.
  + This would be very inefficient, since there could be an arbitrarily large number of autocategories that use a particular field
  + I could narrow the autocatagories that need to be refreshed by using the <relevant\_category>s of the linked items defined by the old and new value of the changed link field.
    - So if <author> is changed from “J.K. Rowling” to “J.R.R Tolkien”, the <relevant\_category>s of both “J.K. Rowling” and “J.R.R Tolkien” are refreshed
* A category like “Books” will already contain all items of type Book in an “all books” category. Adding more refined categories within “books” like “fantasy Books” or “books by J.K. Rowling” does not contribute to the set of items that are descendants of “Books”
  + If there were to be more refined categories like “fantasy Books”, it would be desirable for all parallel categories of that grouping type to be present (i.e. all divisions of Books by genre)
  + It could become very difficult to maintain such groupings cleanly, especially since many categories such as “fantasy books” would have to appear under multiple parent categories (“books” and “fantasy”)
  + It might be easier, and cleaner, to have 2nd level grouping done automatically on the fly
  + Within the “books” item could be an option to “group by” various different fields such as genre or author.
  + Grouping can be difficult with Xquery, but since I have a list of all possible values of the relevant fields (such as author/genre), I can do it easily enough.
  + When a “group by” field is selected in “Books”, all relevant categories are shown with the number of children and descendants for each category
  + When a category such as “fantasy” or “law” is opened, a Search page is opened with the relevant filters (i.e. template=Book, genre=fantasy)
  + I think this is a good design principle, because it shows off the flexibility of the Search functionality and will encourage people to use it for what it is capable of.
* Now that I don’t have to worry about autocategories with multiple parameters it will be easier to automate them
  + The “Books” category is easy to fill:
    - When an item is created, a Template is chosen
    - Say the Book Template is chosen
    - The Book Template will contain a list of <relevant\_category>s of which one (perhaps the only one) will be the “Books” category item
    - On creation of the new Book item, this <relevant\_category> will be found and added as a parent to the new item
  + “works by J.K. Rowling” category item:
    - “J.K. Rowling” is selected as an author for item (A)
    - “J.K. Rowling” Person item is checked for <relevant\_category>s
    - The “works by J.K. Rowling” is found and added as a parent to (A)
    - The old <author> (if the field is being changed rather than set for the first time) is checked for <relevant\_category>s and any found are removed from the parent list of item (A).
  + Since there will only be one condition allowed for autocategories now, an item can only have at most one <relevant\_category>
    - There will be a <relevant\_category> for every Role an item has
* I can now go back to using the Predicate concept
  + An item (C) called “Harry Potter and the Chamber of Secrets” is linked to the Person item (A) called “J.K. Rowling”, through the <author> field.
  + (A) has never been used in the Author Role before
  + The Role: “author” is defined for (A)
  + The Role item defines the format of the Predicate item that should be created, including how to name the Predicate item and how it’s condition should be defined
  + A new Predicate item (B) is created
    - The author Role item defines that (B) should be named with format: “works by <name>” which in this case resolves to “works by J.K. Rowling”
    - The author Role item defines that (B) should have a condition on <author> field.
    - The condition must point to the “J.K. Rowling” item so the condition on (B) is <author id=(J.K. Rowling)>
    - The Predicate item (B) is “refreshed” which means all items satisfying its condition are found and added as children
      * This is only necessary if items have been allowed to link to (A) through the <author> field before (B) was created
  + In (A), the <role id=(author)> element is given an attribute @predicate=(B)
  + Whenever an item is linked to (A) through the <author> field (connecting to (A)’s author Role), it has (B) added as a parent
  + When a link to (A) through the <author> field is removed from item (D), the Predicate (B) is removed from (D)’s children
* The link between a linked field and its relevant Predicate item can be two-way
  + If the <author> field is filled, a Predicate item is added as a child
  + If that Predicate item is manually added as a child, the <author> field is overridden with the appropriate value
  + To have permission to do either of these actions, you must have permission to do both
  + I no longer need @plevel=”auto”
* Investment rules
  + For the first investment in an item, 10% goes to item owner, 90% goes to Treasury
  + If an Investment is larger than the total current worth of an item, the excess goes to Treasury
    - This means that the most an existing investor can make from any one 3rd party investments is the value of his existing investment.
  + Whatever proportion of an investment would go back to the investor because he has existing investments, instead goes to Treasury
  + A user can invest in his own item, but the 10% commission he would receive as owner instead goes to Treasury
* With these rules, it may not be necessary to have a tax
  + For p=0.9 in
  + An item levels every factor of 10 increase in worth
  + This means that a user can make 200% profit on an investment every time the item increases in worth by a factor of 10
  + The tax may be useful to control the rate of inflation, eventually
  + I will implement the tax functionality into the site for later use
  + Initially, though, it makes sense to make tax=0
* A group of users collaborating can still take advantage of the system
  + User A owns an item
  + User B invests Z100 in it
    - User A receives Z10
  + User C invests Z100 in the item
    - User A receives Z10
    - User B receives Z90
  + User D invests Z100
    - User A receives Z10
    - User B receives Z45
    - User C receives Z45
  + ...
  + User A has gained Z30
  + User B has gained Z135
  + User C has gained Z45
  + User A invests Z30
    - User B, C, D each receive Z9
  + User B invests Z135
    - User C, D each receive Z41
    - User A receives Z13.5
  + User C invests Z45
    - cbf
  + assume all users invest their earnings once more
  + Z300 was spent
  + Total Worth of item = 300 +30+135+45+ 9\*3+41\*2+13.5+~30 = 662.5
  + Multiplier of somewhat more than 2
* What happens with a 10% tax?
  + User A owns an item
  + User B invests 100 in it
    - Treasury: 90
    - User A: 10
  + User C invests 100
    - Treasury: 10
    - User A: 10
    - User B: 80
  + User D invests 200
    - Treasury: 20
    - User A: 20
    - User B: 80
    - User C: 80
  + ...
  + User A invests 40
    - Treasury: 8
    - User B: 8
    - User C: 8
    - User D: 16
  + User B invests 160
    - Treasury: 16 + 29
    - User A: 16 + 12
    - User C: 29
    - User D: 58
  + User C invests 80
    - Treasury: 8 + 13
    - User A: 8 + 5
    - User B: 35
    - User D: 27
  + Assume all users invest their earnings once more
  + Initial investment size: 400
  + Total worth of item: 400 + 40+160+80 + 28+42+37+27 = 814
  + Multiplier of a little more than 2
* I can’t see that the 10% tax makes an astronomical difference
  + The multiplier effect here is unavoidable I think
  + If users are to be able to make profit from investing, they need to make money back on their investments, thus there will be a multiplier effect
  + If I reduce the ability of groups working together to rig the system, I am reducing the profitability of investing
  + There is a major flaw to overcome here, though.
  + I want it to be profitable to invest, but I don’t want users working together to be able to massively multiply their net worth
* I don’t think even a tax can prevent the ability to multiply worth massively
  + User A owns item
  + User B invests small amount, say 10
    - Only 9 zuth lost to Treasury
  + User C invests 10
  + User D invests 20
  + User E invests 40
  + ... each investment is equal to the total worth of item
  + With many investors, all with lots of zuth, the item can be given a huge worth
  + If tax=0, only 9 zuth was lost to Treasury, all the rest remains in this closed system (where all users investing are collaborating)
  + The collaborating group of users can then go and do the same thing on a different item
  + If there is a tax of say, 10%, then only 90% of the group’s initial net cash is available for doing the same process on the next item
  + With a retention of 90%, the ability to multiply worth is huge (up to about 9X)
* A possible new valuation system:
  + Users by “shares” in items
  + “shares” may not be discrete units
  + The first share is purchased for 1 zuth
  + After a share is purchased, the value of shares increases slightly, say to 1.1 zuth
  + To buy the 2nd share costs 1.1 zuth
  + To sell the 1st share has to still cost 1 zuth though
  + The 3rd share might cost 1.21 zuth (increase by 10%)
  + I could keep the share or “stock” value behind the scenes
  + When a user invests, they just chose an amount to invest
  + As others invest after them, the value of their “stock” in the item increases
  + If other users start selling their stock, the value of everyone’s stock decreases steadily
  + The stock value is essentially a multiplier on the value of an investment
  + The multiplier is defined as a function of total item worth
  + Worth is simply the net zuth invested in an item, without multiplier effects
* The best way to make money with this system
  + Find items that you believe will become more popular when more people discover it
  + If a lot of investment goes into the item after yours, the value of your investment will increase
  + If you pick the turning point, you can sell your investment at its peak value
  + Once others start selling, the multiplier decreases and each successive sale is valued with a smaller multiplier
* I can model the system as a function
  + I want stock to be measured as a continuous quantity, not as discrete objects, so I need calculus
  + When purchasing stock, the amount of zuth invested fills the area under the function for a certain distance along the x-axis
    - The width (along the x-axis) of the area made under the curve, is the size of the investment
  + when selling stock, your investment represents a length along the x-axis (of say “d”)
    - To calculate how much zuth is gained by selling your investment, integrate the function through length “d”, with the right terminal of the integral matching the current worth of the item.
    - The worth of the item is reduce by amount “d”
  + The function needs to have a positive gradient so that an integral of width “d” is larger the more to the right along the x-axis it is
  + For increment that an integral of width “d” moves to the right along the x-axis, I want the area of the integral to increase by a constant factor
    - Actually for an increase in total area under the graph (zuth) of factor “k”, I want the value (in zuth) of an investment of width “d” to increase by factor “k”
    - The principle I’m trying to achieve is that when the worth of an item increases by x%, the value of your investment increases by x%
  + The function I need is f(x)=e^x
  + The problem with this function is when the item’s worth is near zero.
    - The first investment will acquire an unreasonably large quota of the x-axis, which will become stupidly valuable once the item’s worth increases substantially
    - Beyond the very early stage, though, the value of investments should increase in proportion to the worth of the item
    - I’ve checked the function and it works perfectly once worth is large
    - For the first investment, though, it’s a mess
    - By changing the function to f(x)=Ae^(x), and making A somewhere close to an average investment size, the profit made by early investors is not so outrageous
    - Unfortunately, there isn’t really a way to make “A somewhere close to an average investment size)
    - If the first investment is significantly larger than A, this investor will effectively own the entire worth of the item for a LONG time.
    - If A is very large, it is effectively like a large initial investment (of about A)
      * This means the item needs to increase in worth by an amount equal to A for the value of the first investment to double.
      * A literally acts as a default first investment on all items by the Treasury
    - The problem is the concept that an investment increases in value by x% when the item increases in worth by x%
      * This just doesn’t work out sensibly for an item starting at zero worth – the first investors gain too much of a stake
* Investment system design principles
  + The return from selling an investment should be dependent on the worth of the item compared to the size of your investment
  + If the worth of the item is about double the size of your investment, you should receive a small profit when selling your investment
  + If the worth of the item is many times the size of your investment, you should receive a substantial profit when selling your investment
  + Actually the profit made needs to be dependent on the amount invested after your investment compared to the size of your investment
* Another Investment system
  + When you invest in an item, the current worth of the item (before investment)= i
  + The value of the investment = v
  + When you sell the investment, the worth of the item = f
  + The amount invested after you
  + profit =
  + example 1 (100% profit)
    - you invest 100 zuth
    - 1000 more zuth is invested after you
    - You sell at this point
    - G=1000
    - g/v=10
    - profit = v = 100%
  + example 2 (a loss)
    - you invest 100 zuth
    - worth of item decreases by 1000 zuth
    - you sell
    - g=-1000
    - g/v=-10
    - profit =-v
    - total return = 0
  + a problem in this formula
    - you invest 10 zuth
    - worth of item decreases by 1000
    - you sell
    - g=-1000
    - g/v=-100
    - profit =-2v
    - total return = -v (you owe money)
    - THIS FORMULA DOESN’T WORK
  + this concept is close though, I think
  + I need a formula which depends on change in worth compared to investment size
  + the formula needs to define the return from selling, which must be a positive value
  + return = e^(g/v)\*v
    - if g is –ve, return gets progressively smaller, but never reaches zero
    - this formula is too extreme: for large g/v, return becomes gigantic
  + formula properties:
    - return->0 as g/v->infinity
    - return->infinity as g/v -> infinity
    - return goes like log(g/v) for positive g/v
    - return = 1 for g/v=1
  + because of the relative way that returns are calculated, each investment would have to be treated an calculated separately
    - I’m not sure if it would be possible to allow partial sales of investments
    - It could probably be done with calculus.

# Fri 19-Feb-10

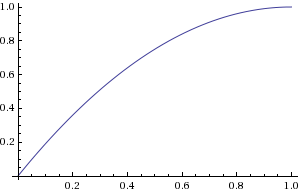
* My current formula properties do not anchor the function at worth=0
* Return needs to approach zero as worth approaches zero and reach zero at worth=0
* Investment system Design principle
  + The first 20% of investments (by value) are worth, say, 40% of item worth
  + The specific numbers (20% and 40%) can be fiddled with of course
  + I think the function that would do this is simply y=kx
    - No, this is wrong. I was thinking in terms of similar triangles, but I have to work with ratios of areas not ratios of lengths
  + This rule does not extrapolate in the way I wanted
  + For f(x)=mx
    - The first 20% of investment (by value) is worth 3.47 times its initial cost (69% of item worth)
    - The first 1% of investments (by value) is worth 19 times its initial cost (19% of item worth)
    - The first 0.1% of investment is worth 62 times its initial cost (6.2% of item worth)
    - First k\*100% of investment is worth times its initial cost ( of item worth)
    - F(x)=mx is the only function I’ve found so far that exhibits these properties
      * Unfortunately I cannot specify the above properties (“m” has no effect)
      * “m” is actually completely obsolete – it doesn’t affect the behaviour of the investment system at all
    - I need a function that has the property “the first x% of investment is worth y% of item worth”
      * F(x) = mx does this
      * But I need a function for which I can specify x and y myself
    - Assuming that all investments were sold in the order they were purchased
      * g(x)=1/sqrt(x)-1 for x between 0 and 1, defines a value distribution
      * If you integrate g(x) through a segment in the range [0,1], you get the value of that segment when everything is sold in the order it was purchased.
  + Rules
    - the first 50% of the item should be worth about 65%
    - first 10% should be worth about 20%
    - this means from 10%-50% is worth 40%
    - first 1% should be worth about 4%
    - first 0.1% should be worth about 0.8%

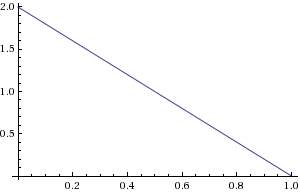
**A function that approximates the rules above**





* A cleaner function





* This function describes the behaviour I want for the value of investments when investments are sold in the order they were bought.
  + I now need a function to describe how to value an investment which is sold at an arbitrary time
  + I’m not sure if I need a new model
    - The old model equated initial investment with area
    - This area was “poured” into a shape defined by some function and the height gained is stored with the investment
    - When selling, that same height is taken off the top
    - If more investments have been “poured in” since your purchase, the value of a given height is greater now than it was at purchase time
* I like the function y=sqrt(x)
  + Y=x^(1/n) is appropriate for n in [1,3]
  + For n=1: value of investment doubles when worth is multiplied by 4
  + For n=2: value of investment doubles when worth is multiplied by 8
  + For n=3: value of investment doubles when worth is multiplied by 16
* It would be possible to alter n even after items have been invested in
  + It would just be a matter of “refilling the vase” as it were
  + The investments could be simply re-enacted, with new “heights” associated with each one

# Sat 20-Feb-10

* For every investment in an item, the investment amount and length it fills should be recorded
* It would be good if there were additional benefits to having an investment in an item apart from the fact that you can eventually sell it to make a profit
  + I could give dividends
  + Dividends should be divided proportional to amount of money spent buying the investment (not on the length value)
  + This would provide an incentive to invest large sums – so you get a larger dividend
  + Where would dividends come from?
    - If I was working with artificial money, I could generate dividends from nothing
      * This would be one way to increase the money in the system
    - If I was working entirely with real money, dividends would have to have a real money source
      * Once ZuthaNova.com is making a lot of profit, I could distribute some of that among investors
      * If I had a chunk of profit to divvy up among investors across all items, how would I divide it up?
      * I would want investments in popular items to be worth more than investments in unpopular items
      * Perhaps this is unnecessary; a uniform benefit to making any investment in my site should not be a problem
      * With the large pool of money that I would be holding on to, I would make some amount of interest, even if I simply put it in a reliable bank account
      * I could divide up the interest made on everyone’s investments and simply pay that back to the investors
      * This seems like a fair thing to do anyway
      * If people can invest in an item and get the same interest they would have if they invested in a bank, there is a greater incentive to use the site
      * When someone sells, what happens to the proportion of the item that other investors own
      * If I just store, in order, the lengths filled by each investment, I can gauge the proportion of the item owned by each user, by integrating along this length at its current position. For selling, I simply move this length to the rightmost edge of the function and integrate it there.
    - I could take an idea from the old investment system and take some money from later investors and give it to earlier investors
      * A tax on every purchase could be collected for dividend purposes
      * The dividend pool should build up for a time and then be divided out among investors say, each month or something
      * The problem with this is that, in the simplest case, all investors will simply get back the tax that was taken from them on investing.
    - With real companies in the stock market, the source of dividends is easy – it is profit produced by the company
      * Items in my website are not producing anything
* Artificial money system
  + How do I initially inject money into the system?
* The real money system has some major flaws
  + Damian warned me that it is probably illegal, or at least in the grey legally
    - The principle that you only make profit if others invest after you is reminiscent of pyramid marketing
    - What exactly are people buying when they buy an investment in an item?
    - Am I providing a gambling service? If so do I need a license?
    - do I need a license if I’m going to make money (by investing in the real stock market) out of the pool of money available to me from people’s investments, and then return some or all of that money back to investors
  + entities with a lot of money can easily boost the value of a chosen item excessively
    - this violates the principle that the valuation of items should be based purely on merit, not on the means of those with a vested interest in having the item rated highly
    - a company could create the item representing their company and immediately invest a massive amount in it
      * there would probably be very little incentive for anyone else to invest in that item, especially if it was obvious that one use owned all of its wealth so far
      * as the first investor, there is no way for the company to loose the money they invested
      * they can sell their investment whenever
      * as long as they leave their massive investment there, they can arbitrarily boost the value of the chosen item
  + users that get in fast and invest in a commercial item early, which is later invested in massively by a someone with a vested interest, could make a killing
    - After this user sells their investment to make their profit, though, the item is once more entirely controlled by the entity with the vested interest
* A system completely separated from real money
  + Users are levelled by me initially, for the degree and quality of their contribution to the site
  + Users are given a salary in zuth (generated from nothing), based on their level
    - Level 0 users get no salary
    - Once they have contributed some good items, a higher level user might level them to level 1 – then they get a small salary
    - With further contributions, perhaps including edits of other items now, they might be levelled again. On level 2, they get a slightly larger salary
  + There is no need for levels to be based on anything concrete
    - it would be a similar process to rising up the ranks in a business
    - your success would depend on how well you are perceived by your superiors
  + there would be no need for a top level; level can increase arbitrarily
    - actually, I would determine the top level, because only I can level the highest level users
  + there needs to be an incentive for user to level lower level users
  + at the moment, there is no way for a level 0 user to acquire zuth unless someone levels them to level 1
  + I could model user privileges much more on the real workplace
    - I “employ” users whose items I like
    - They employ users who are creating items they think will do well
    - Somehow I want to make it so there is an incentive for employers to find users who will create “profitable” (i.e. popular) items, and/or make constructive edits
    - If there was enough of a reason to have employees at all, I could make it that employers must pay their employees
    - There could be a large hierarchy of employees under the top level employer
    - Every user in the hierarchy pays the users below him for their services
    - I want to be personally watch over a small group of users who are held accountable for their own respective employees and so on down the line
  + I need there to be a reward for creating items that do well, as well as for making constructive edits to the site
* Circle of incentives
  + Assertion: people want to make zuth because it can make them real money
  + To make zuth, people can
    - engage in risky investments in items that they believe will be popular
      * In aggregate this leads to useful comparative valuations of various concepts
      * People find this valuation of ideas useful, so they are attracted to use the site
    - Contribute to the orderliness of the site
      * They are paid in zuth (perhaps indirectly), by Zutha (a sort of government)
      * People find the structured organisation of information in the site to be useful, so this attracts more people to the site
      * This is why it is in Zutha’s interests to reward users who contribute to the orderliness of the site
      * The more users who visit the site, the more money Zutha gets from advertising revenue
  + There is an incentive for companies to pay a subscription to have items that advertise their products to show their worth in zuth
    - The more people that come to the site, the larger the incentive to have their products advertised on it
    - The higher the worth of items representing their products, the better the advertising for them
      * Therefore there is an incentive for companies to spend money to increase the value of these items
      * They will do this by paying users with a lot of zuth to invest in these items (supports Assertion)
* Tasks I need done by a hierarchy of users
  + Merging duplicate items
* How to inject zuth into the system
  + Let’s say I pay a “wage” to users depending on their level
  + I cannot decide the level of every user in the site
  + I need to let users levelup users lower than themselves
  + If this is allowed, and the effect of levelling a user is that more zuth is generated from nothing, then there is massive scope for abusing the system
    - A level 3 user could level lots of level 0 users that they themselves own
    - The huge collection of now level 1 users start getting a wage
    - One person owns the revenue of all these users
    - The users could be new and without any credibility to be levelled
    - Sure, the culprit might be caught, but he can take advantage of his massive unjustified income for some time at least
    - I don’t want holes like this in the system
  + One option is that the higher level users must pay the income of the users they level
    - For this to be workable, there needs to be something given to the higher level users in exchange for the income
    - This is where the workplace model comes in
    - So what can these lower level “employees” do for the employer in exchange for income?
* I want there to still be a level system for users, as this will determine permissions
* A possible power hierarchy system
  + I level a small group of users to varying levels
  + I have responsibility for ensuring that these top level users act in the best interests of the site
  + Users are paid a salary from the treasury based on their level
  + Each of these top level users has a maximum quota of lower level users that they can “employ”
  + A level 5 user could employ, say, 10 level 4 users, or 8 level 4 users and 4 level 3 users
  + The employees of a user are directly attached to that user
    - If the boss is demoted one level, all his employees are demoted one level
      * The level 1 users at the bottom of the hierarchy “fall off” the bottom of the hierarchy
      * They become level 0 users again and cease to be “employed” by anyone
    - If the boss is “sacked”, he and all his employees return to level 0
      * The justification for this, is that if the boss is dodgy, his poor judgement will be expressed in the hierarchy of employees he has chosen
        + The only thing that has given the employees their level is the judgement of the boss
        + There is nothing external to the boss which justifies the employees retaining the level they had under that boss
        + If the bosses credibility is shattered, then his credibility to determine the credibility of his employees is also shattered
      * The individual merit of all the “disbanded” employees is determined by their deeds
        + If someone has been doing good work under a bad boss, they still have the history of this good work to show off
        + Ideally, other good employers would seek out users who were doing a good job from among the disbanded
  + Employers can vie for the services of the most sought after users
    - An employer might offer a user working under a different employer a higher level if they work for him
    - When a user changes employer, all his employees remain attached to him – the whole hierarchy under the moving user moves with him
  + What are these users all getting paid for?
    - At the top of the system, I will pick out users who match certain criteria, to be promoted
      * users who contribute constructively by adding lots of good items
      * users who show a desire to make the information in the site useful in the most objective way possible
      * users who show they can be objective and open in debates, particularly about issues regarding the organisation of the site
      * users who show they are good at managing other people
        + they need to be good at finding and praising people who are good at these things
        + They need to show they are strict in dealing with people who act immorally in the site (i.e. vandalising, pursuing an agenda...)
    - I will promote, demote or sack these top level users based on how well they perform their task of managing the hierarchy under them
    - To create a more concrete incentive to do what’s in the best interests of the site, I will eventually pay these top level users a percentage of the site’s real-money income
      * The merit of these top level users is based on how well the improve the usefulness of the site
      * If they do this task well, the site will become more popular, advertising revenue will increase and they will earn more real money
* If zuth itself is completely isolated from real money
  + This will improve the quality of the valuation of items because it will be less weighted on the vested interest of entities with financial means
  + I don’t think I can incorporate donations and affiliate programs, in terms of providing bonuses of zuth to donate and purchase products
    - Actually just having the affiliate links there might be quite profitable, even if I don’t provide an incentive in zuth for buying through my site
    - In fact, there doesn’t seem to be any kind of unified system for affiliate programs and for many, if not all, it would be impossible to connect a particular commission with the zuthanova.com user that purchased the product which resulted in the commission.
  + The only source of income for the site would then be subscriptions from companies wanting their commercial items to show their worth
  + Owners of items should receive a percentage of investments in items
    - Actually a percentage of each investment (the first and most valuable section) should be given to the owner
    - The owner can chose at his own discretion to sell his investments for cash (zuth cash)
  + Owner of commercial items then have an incentive to have the item paid for so it shows its worth
* Perhaps later on, I could introduce a separate, but parallel real money investment system
* Commercial items whose worth is hidden
  + Investing in such items is blind
  + After you’ve invested you can see the value of your investment and see how it changes
  + However, this doesn’t give you enough information to determine the total amount invested in the item – it’s worth

# Sun 21-Feb-10

* What should users do when they have complaints about the actions of another user?
  + First, communicate with the offender themself
  + If the conflict cannot be resolved, the offender’s employer can be contacted
  + Every user’s employer should be an easy to find link on their User item
  + The user with the complaint can continue to move up line of employers until their issue is resolved to their satisfaction
  + If the complaint is unreasonable, it would be expected that someone in the line of employers would set the complainer straight about what they can expect
    - If the complainer continues to be stubborn in their unreasonableness, then it would be expected that they would simply no receive any further help
  + At every level of power (except Zutha himself – me) there is an incentive to help the complainer
    - If a user does not help resolve the complainer’s issue, they should expect to be disciplined by their employer
    - If a given user in the hierarchy ignores the complainer, the complainer can go the next level of power and complain about being ignored
  + Users need to be encouraged to start at the bottom of the pyramid when requesting aid from authority
    - It will not be efficient if it becomes common for users with complaints to “go straight to the top”, instead of complaining to the most relevant authorities first and moving up the chain if they don’t get adequate help.
  + The first thing to do when requesting aid from authority, would be to post a comment under the item that requires attention
    - Examples:
      * An item you don’t have permission to edit should have something about it changed, such as parents or a specific field
        + Action: write a comment under that item requesting a change to a field in the item and flag the comment as just such a request
      * You find a duplicate item that should be merged with an earlier created item representing the same thing
        + Action: write a comment under the duplicate item claiming it is a duplicate, and pointing to the item it should be merged with
      * You disagree with some factual information stored in a particular item
        + Action: write a comment under the item requesting the correction
      * You believe a new category should be created somewhere for a concept not catered for by the site yet
        + Action: write a comment under the item under which the new category should go requesting the creation of the new category item
    - If no one steps up to help you, you can request help from a specific user by adding your comment item as a child to that user’s “notifications” Item.
      * The smart way to get attention would be to find a user whose notifications item is not very populated with requests
* Bullying
  + A level 2 user (the bully) vandalises a level 0 user’s item
  + The level 0 user (the victim) complains to a higher level user
  + The bully is watching the victim’s actions and quickly silences his complaint by deleting it, changing its parents to irrelevant ones, or marking the complaint item as resolved.
  + I need to make sure it is possible for even level 0 users to be heard when they have a valid complaint
    - There needs to be a way for low level users to directly communicate with specific higher level users without interference
    - Comments that are placed in the “notifications” item of a high level user (A), could be made editable only by (A), the creator (B), and users of a higher level than (A).
    - I could set this “notifications” item up like an email inbox, with features to mark possibly multiple comments at once as “read” or “completed”
    - This “Inbox” item would act as a combination of an inbox and a troubleticket system
  + At the same time I need to make sure spammers and deliberate troublemakers cannot overwhelm the system by flooding the system that allows level 0 users’ complaints to be heard.
    - The only way to prevent this on a mass scale is by having technical measures in place to block IP address responsible for the flooding
    - I don’t believe this will be a large issue though.
    - There would need to be an incentive to be disruptive in this way
    - Disgruntled people may want to get back at a user who they believe has wronged them by flooding their inbox
      * Level 0 users would have a limited rate at which they could post comments
      * If an IP address made lots of level 0 users and used up all their item creation quota on flooding a target’s inbox, then it could be detected, or at least reported
      * The IP address would then be blocked
      * If someone was really determined, they could keep getting new IP address and keep bothering their target
      * The impact this would have on the target would be that legitimate requests for their attention would be difficult to find amongst all the maliciously sent junk
        + Unless the attacker got really creative, though, it would likely be possible to set up a filter to exclude the spam messages
        + The worst consequence I can think of for such an attack is that the victim stops looking through items sent to their inbox by level 0 users
* It is important that users only rise in the ranks slowly
  + If it is possible to gain a high level quickly for a few well received actions, then this can be repeated multiple times from level 0 users
    - This is a problem since it allows people to be bullies with a high level user
    - If it is very difficult to get a high level user, and takes a lot of time, then there is more to lose by abusing the privileges of a high level user
    - The more people have to lose by abusing the privileges of a high level user, the less likely they are to do so
  + This principle may already be an inherent aspect of the system
    - In order to get to level 2, you need to have shown yourself at level 1
    - In order to get to level 4, you need to have proven yourself at level 2
    - ...
    - A user is limited in his ability to show his usefulness as an editor and as a leader while on a low level
    - To show that you can command a large hierarchy beneath you, you need to be a high enough level to have that large hierarchy beneath you.
* Wage system
  + Level 1 users earn 1 zuth per day
  + Level 2 users earn 2 zuth per day
  + Level 3 users earn 4 zuth per day
  + ...
  + Level 2 users can employ up to 4 level 1 users
    - The sum of wages paid to the level 2 user’s employees is 4 zuth per day
  + Level 3 users can employ up to 4 level 2 users or up to 8 level 1 users
    - Sum of wages = 8 zuth per day
  + Level 4 users can employ up to 4 level 3 users, up to 8 level 2 users, or up to 16 level 1 users
    - Sum of wages = 16 zuth per day
  + ...
* It should be possible to reach a high level by being very good at either making contributions to the site on your own, or creating and leading an effective hierarchy of useful contributors
  + A level 4 user may chose not to employ any underlings. Instead, he shows his worth by being very active in the site
    - This user will gain credibility based directly on the quality of his contributions to the site
  + Another level 4 user may prefer to be valuable by finding and leading a team of employee users to contribute to the site
    - This user will gain credibility through the value contributed by his employees
* New permissions system
  + @plevel=”all:3;owner:0;parent:0”
    - the owner can edit the item at any level
    - the owner of a parent item can edit the item at any level
    - anyone else must be level 3 to edit
  + @plevel=”all:admin;group-3H5B:0”
    - Anyone in the group 3H5B may edit the item at any level
    - Everyone else must be at least level 6 to edit the item
    - This functionality is to allow groups to collaboratively work on items, without just anyone being able to interfere with their work
  + @vlevel=”all:admin;group-3H5B:0”
    - Only people in the group 3H5B can see the item
  + By using the group identifier in @plevel and @vlevel, companies and other bodies could have their own private section of the website
    - This might be used as a forum for sharing ideas concerning the improvement of the company
    - Groups like this might want to have their own isolated power hierarchy
    - It would also be desirable for them to have an isolated currency distribution within their niche section of the website
      * I can’t think of a clean way to isolate zuth for use only on certain items
      * Perhaps providing a private niche section of the site to companies will be impractical
        + The other way to provide the same functionality would be to license the software to companies to use privately
        + The advantage of having a private subsection of the hierarchy would be that the benefit of the rest of the site is close by
        + If companies licensed the software privately, the entire hierarchy would be their own, but then employees would have to visit ZuthaNova.com separately to get information from there.
        + Now that I think about it, this may not be such a problem
  + Without need for private sections of the hierarchy for use by private groups like companies, the @vlevel attribute would not need the group identifier “group-3H5B:0” functionality
  + @vlevel common values
    - @vlevel=”all:0”
    - @vlevel=”all:admin;owner:0”
* There’s a lack of flexibility in organising one’s employees
  + Imagine an employer (A) on level 5 has 4 level 4 employees
  + (A) finds a level 3 user (B) somewhere that he wants to employ at level 4
  + (A) is underwhelmed by one of his level 4 employees (C)
  + (C) is still a valuable employee, but (A) thinks (B) would do a better job in his place
  + (A) wants to demote (C) to level 3, promote (B) to level 4 and give (C) to (B) as a level 3 employee
  + Problems:
    - When (A) initially promotes (B), he will have 5 level 4 employees, which is more than his quota
    - (B) will probably already have 4 level 2 employees. When (B) is promoted to level 4, he may want to simply promote all his level 2 employees to level 3. In this case he won’t have room to take on (C).
  + In a real business
    - I think it is uncommon for a manager to be demoted and placed underneath a person who is promoted to take his place
    - Because of the nature of people’s ego, a demoted person is likely to harbour bad humour towards the person who takes their place
    - Demotion does happen though. It would be inefficient to be forced to either retain or sack an underperforming manager.
      * By sacking the manager, you may lose an asset that would have been valuable in another position
      * By retaining the manager you are unable to replace him by someone better able to perform his duty.
    - The usual reason a manager is replaced is if he leaves, or is sacked (I think)
  + My site needs to be treated as one big company, with Zutha at the top
    - If a high level user is “sacked” it is not workable for him to retain his hierarchy of employees and his level, without anyone above him
  + An employer should only be concerned with his direct employees
    - Each user handpicks their direct employees
    - Those employees have each handpicked their own employees
  + If the hierarchy anywhere is to change, someone has to be demoted or sacked
  + There needs to be a buffer between the normal amount of employees one has, and the maximum amount possible
    - This will allow more room to shift the hierarchy around
    - The normal number of employees should be 4 (of the level below you)
    - The maximum number should be 6
    - Users are only allowed to go over their quota of employees for a limited period of time, say 3 days
    - After the “overflow” period is up, there needs to be a strong incentive to reduce employees back to the normal quota
    - The incentive could be that, until the correct quota is restored, wages dry up for the user with too many employees as well as the whole hierarchy beneath him.
  + It will be difficult to rearrange the hierarchy when the norm is for everyone to have their full quota of allowed employees
    - I need the employee quota concept to be more flexible
    - Some users may not be allowed to have as many employees as others
    - Actually the total allowed employees could increase with level
    - This would mean that when a user is promoted they would have more slots available for extra employees, even if all their existing ones were retained and promoted with them.
* This power hierarchy system is very much like a bureaucracy
  + I would like to find a system that keeps the effective parts of the bureaucracy system and discards the bad parts
  + Actually one major distinction that already exists is that my system is a completely “open government”.
  + I wonder whether this alone is enough to make it work, even if it copies the bureaucracy system in every other way
* It should be easy to view all actions performed by any given sub-hierarchy. E.g.
  + find all event items owned by a particular user or any of his direct employees
  + find all event items owned by a particular user or anyone in the first 3 levels of the hierarchy beneath him - employees (of employees (of employees)) of him
  + find all event items owned by a particular user or anyone in the hierarchy beneath him - employees (of employees (of employees (...))) of him
* What happens when a high level user with lots of employees is sacked?
  + The sacked user should go back to level 0
  + It seems unfair that his employees should all be returned to level 0 (sacked as well in effect) if they have done nothing wrong
  + I’ve been trying to treat employees as loyal to their direct employer only - not employers higher up the chain
    - This is not like a real business
    - In a real business, the people employed by (A) are really employed on behalf of (A)’s employer (B)
    - When (A) is sacked, his employees become the responsibility of (B) temporarily
    - (B) then employs some new, (C), to take (A)’s place managing these employees
    - If (A) has made poor decisions in the people he has employed, it is (C)’s responsibility to sort out the mess
    - If a worker is “poached” from somewhere else in the hierarchy, he leaves his old employees behind and takes on new ones in his new role
  + The reason I wanted to avoid this system is I think there would be a benefit if a user’s employees stuck to him where he went
    - The employees and their boss would get to know one another after working together
    - it would be a shame if this bond had to be broken whenever an employer moved jobs
  + maybe I can have it both ways
    - when employing someone (A), the employer (B) should be able to choose whether to let (A) retain his own employees
      * In general it would be preferable if (A)’s employees were retained
    - when sacking someone (A), the employer (B) should be able to choose whether to sack the entire hierarchy under him, or retain (A)’s employees under him
      * In general it would be preferable that (A)’s employees not be sacked
      * Actually, I can’t think of a situation where sacking all employees would be justified
      * The user that takes over (A)’s job, (C), should have the responsibility of dealing with (A)’s employees
      * (C) may bring his own employees with him, and want to keep them
      * (C) should be given some time to rearrange the hierarchy beneath him to cater for the extra employees
* What happens when a high level user with lots of employees is demoted
  + One idea is to cause all employees beneath him to be demoted as well
  + This seems unfair on these employees, given that they are not necessarily responsible for their boss being demoted
  + However, the credibility of the employees to be at the level they are comes from their employer.
  + If the employer is demoted, does that imply that his employees deserve to be demoted as well?
  + If a level 5 user (A) with 4 employees is demoted, he cannot keep all 4 employees because level 4 users are only allowed 3 Level3 employees.
    - (A) is responsible for reducing the number of his employees back to the normal amount
    - He may
      * sack one employee
      * demote 2 employees
    - If (A) were being demoted from level 3 to level 2, he would have to sack 1 of his 2 employees
* There should be a way for users to be “looking for work” while retaining their level and employees, and possibly their income
* Privileges by level
  + Level 0
    - Create items only – no editing
    - Unemployed
    - No income
  + Level 1
    - Editing of items will now be allowed
    - Employed
    - Income: 1 zuth per day
  + Level 2
    - Income: 2
    - Employees: 1X Level1 employee allowed
    - Total income of employees: 1
    - Total income of hierarchy: 3
  + Level 3
    - Income: 4
    - Employees: up to 2X Level2 / up to 4X Level1
    - Employee income: 4
    - Hierarchy income: up to 10
  + Level 4
    - Income: 8
    - Employees: 3X Level3 / 6X Level2 (no level 1 employees should be allowed)
    - Employee income: 12
    - Hierarchy income: up to 38
  + Level 5
    - Income: 16
    - Employees: 4X Level 4 / 8X Level 3
    - Employee income: 32
    - Hierarchy income: up to 168
* Each individual employer should be able to choose how he deals with the hierarchy beneath him in his own way
  + Your options when sacking an employee (A)
    - Sack (A) and keep his employees as your own
      * In most cases it will be necessary to employee someone else or promote someone to take the place of (A)
    - Sack (A) and everyone beneath him
      * This action is likely to cause bad feelings among those beneath (A) who may believe themselves unfairly punished
  + Your options when demoting an employee (A)
    - Demote (A) only and take all his employees as your own
      * This will likely leave you with more than your quota of employees
      * Someone else (B) will need to be put in (A)’s old place and (A) added to (B)’s employees – at least this is one way to resolve this
    - Demote (A) and everyone under him
      * Many of the users under (A) will be forced to cut down their own employees

# Mon 22-Feb-10

* I really want there to be a reward for people buying a product through an affiliate link on my site
  + I feel I will be able to justify making a rule that all affiliate links on the site must be mine then
  + Problem is I don’t think it will be practical to detect which user deservers the reward
  + If people were rewarded with a very small amount of zuth, then I don’t think the ability to get hold of zuth with real money will unbalance the integrity of the voting system much
  + The only real issue will be if a company buys its own product a really huge number of times to get a lot of zuth
    - I think this problem will be unlikely as it is exceedingly dishonest and could likely be detected
  + Ideally I would set up my own custom affiliate standard which would allow detection of user
    - The simplest thing I can think of is to put the onus on the buyer to enter his ZuthaNova.com username when purchasing the product so that he gains an investment in the relevant item for that user.
  + I would prefer it if affiliate purchases did not generate new currency
    - I want a system analogous to the one I had for the old investment system, where users could be given an investment in an item without that generating new currency
    - The value of the investment should be zero initially, and increase steadily from zero as the item worth increases above the point it was at at the time of affiliate purchase.
  + It would probably take a large scale initiative to implement an affiliate system that allows detection of users
    - Many companies would have to be convinced to use this system before it would be worth implementing the reward system with zuth
  + In the meantime, I still want to make use of affiliate programs
    - I could personally collect affiliate links and change the links to product pages to affiliate links for each affiliate program I set up
    - The bulk of visitors to the site are unlikely to have user accounts,
    - People with no user accounts won’t care if there is no reward for purchasing a product through a ZuthaNova.com affiliate link
    - So the most important thing is to have the affiliate links there
    - I need to make it a significant offence to add your own affiliate links to the site
      * To justify this morally, I need to explain somewhere, my plan to make a reward system for buying through affiliate links
        + Also, I should make it clear that some of the money earned through affiliate programs will be given back to contributors of to the site
      * It may not be practical to stop level 0 users from creating new product items and linking them with their own affiliate links
      * This should not be too much of a problem as long as the affiliate links are eventually replaced by my own
      * The only action that needs to be considered a great offence is for a high level user with adequate privileges to change a ZuthaNova.com affiliate link to their own
      * Since high level users have a lot to lose, I should be able to trust them to maintenance affiliate links for me
        + required maintenance might include changing links when the product page moves
        + it would be important that users high enough to change affiliate links (which will have a high permission level put on them) know to keep the affiliate information in the link when they change the link base.
        + This would only be relevent to affiliate links that use GET variables to identify affiliates
        + I could store such GET data in a separate field with @plevel=”admin”

This way I could allow the base url to be edited and the affiliate GET data would be kept

* There could be some benefits to regular bulk lay-offs
  + If sacking someone causes everyone in the hierarchy beneath him to go back to level 0 too, then it will allow a lot of reshuffling of the hierarchy
  + This could improve efficiency
  + I think it would still be good if bosses had the option to sack a user but keep their employees as their own
* There should be a way for a user and the hierarchy under him to be cut off from the main power hierarchy, without this hierarchy collapsing
  + Income should cease for users in this cut-off hierarchy
  + They should be put in an “unemployed” state which prevents them from using the privileges normally granted them by their level
  + When a high level user is sacked, his employees and their hierarchies should be placed in this dormant state
  + Users who are looking for new employees should look first in the pool of dormant workers
  + In order to encourage recruitment from among unemployed users, rather than poaching, I should create some hindrances or disincentives to “poaching”
  + Poaching means taking an employee from another employer
    - This should of course only happen at the employee’s consent
    - This should not happen for no good reason
    - It should happen if the employee generally believes the new employer is better and would prefer to work for him
    - There needs to be a slight disincentive for one or both of the poacher and the poached
    - There should be a period of “transfer” time after a worker is poached, during which the employee is in a limbo state and does not get paid or have editing priviliges
    - This should be enough of a reason for workers to hesitate a little before accepting a change of employer
  + If a new level 5 employee is desired, the options should be to find a level 5 or level 4 user from among the unemployed, or poach a level 5 or level 4 user
  + It should not be expected that users in the dormant (unemployed) state deserve to retain their level
    - This level was given to them by someone who may have been sacked
    - The new employer should decide if the unemployed user deserves the level they had when they lost their job
    - unemployed users will often have to choose between losing some levels and being employed again, or waiting longer in the hope that someone will employ them who will let them keep their current level
  + there is a distinction between “sacking” and “laying off”
    - Sacking means the user should return to level 0. This is a major punishment
    - an employee (A) would be “laid off” if
      * the employer (B) has run out of his employee quota
        + this will happen if (B) has been demoted and now has a smaller employee quota
      * the employer believes there is someone better to fulfil (A)’s job and wants to replace him, but that does not mean (A) has done anything wrong. (A) might be entirely deserving of his level.
      * The employer wants to slim down on the number of employees he has
        + (A) might simply be the worst of a good bunch, and doesn’t really deserve punishment
        + This is the purpose of the “unemployed” status – for laid off users who haven’t done anything wrong
* Power hierarchy modification options
  + Sack
    - User is returned to level 0
    - His employees retain their level and their own hierarchies, but are put in the limbo state with “unemployed” status
  + Lay off
    - User retains his level and he, with his hierarchy, is put in the limbo state
  + Demote
    - User’s level reduce by 1
    - User’s employees are demoted also
    - User will have to choose an employee to lay off if he had the maximum allowed employees at the time he was demoted
  + Promote
    - User’s level increased by 1
    - His employees are not automatically levelled
    - User may choose to levelup all his employees as well
    - He will now have a slightly increased employee quota
  + Poach
    - User is asked to join a different employer
    - He may choose to refuse
    - If he accepts, he and his employee hierarchy will be put in transit mode for a period of time, during which time they will not receive income or have editing privileges
    - After the transit period is over, the user will have a new employer
    - His employee hierarchy will have moved with him
* Real money based investment system
  + This should be an entirely separate system from the real-money-disconnected zuth market
  + The two currencies for investing: real money and zuth, should be run in parallel
  + Items will have two scores, their zuth worth and real money worth
  + Purchasing an item with an affiliate link supporting ZuthaNova user detection will provide the purchasing user with an investment in the relevant item.
  + I’m still weighing up whether the real money system should allow money to flow out as well as in
    - With money flowing out, it would really become like a gambling system
    - if money could only flow in, paradoxically, it might result in less ill will, because people would be spending money knowing it is being spent
    - if people could sell their ZuthaNova currency again, some people might begin to believe that an investment in a ZuthaNova item is actually an investment like buying a house or stocks
  + There needs to be a real incentive for spending real money just to be able to “vote” on an “item” in cyberspace
  + I could use the dividends concept, even if ZuthaNova currency was not sellable for real money
  + I could divide up a proportion of profits among real money investors, proportional to their wealth in virtual currency terms
  + This way, you could purchase Zutha currency with a small initial investment of real money
    - By investing in ZuthaNova items intelligently, you could make a large amount of Zutha currency compared to the amount of money spent
    - You would then receive larger dividends
    - I think the prospect of an increasing real money income from the site by playing the investment game well, would spur many people to engage in the game
  + It might actually be more intoxicating to people if only the best players of the Zutha game actually got paid real money
    - The amount they received would then be able to be much greater
    - People would see some people making large amounts of money and want to get in on the deal
    - If the money were too spread out, no one would be earning a stunning amount, so it wouldn’t be as intoxicating
    - This is a similar psychology to what attracts people to lotteries
    - If the profits from a lottery were divided up evenly among everyone who bought a ticket, they would be vastly less popular - indeed, everyone would lose money.
    - If, instead of making it a chance game, I simply paid large sums to a top percentage of successful investors in my site, this could be even more attractive to people since they would feel they are in control of “winning” rather than chance.
  + I would need to think carefully about who deserves to be making the money
    - Someone willing to spend large amounts of real money, could easily become the richest user in zuth currency
    - In order to actually make profit from ZuthaNova, you should need to multiply your Zutha currency many times beyond the amount you bought with real money, otherwise the amount you win would be less than what you spent to win.
    - If this is the case, it may deter people from spending exorbitant amounts of money to be the richest person in ZuthaNova
    - Then again, some people might decide to spend that money and become the richest person in ZuthaNova, even if the winnings they receive does not exceed what they spent
      * If this happens, there is likely to be ill feeling among the users who multiply their wealth really successfully, but do not receive their due reward because someone with a lot of money is taking it from them
      * As long as a relatively large proportion of the top echelon of the ZuthaNova rich are rewarded with real money, plenty of those humble (in real money) user who make a killing in ZuthaNova, will receive their due
      * In fact, if a large amount of money is spent by people trying to become rich in ZuthaNova, then the extent of the rewards can be increased because there will be more money available.
      * Any users who spend much more than they earn back from rewards, will be contributing to the amount earned by the users who legitimately deserve the rewards – i.e. they earned their ZuthaNova money by being smart investors in ZuthaNova, not by buying the bulk of it with real money.

# Wed 24-Feb-10

* I’m beginning to think it makes sense to store different kinds of <children> under different fields
  + Comments on an item should be stored under <comments>
  + Employees should be stored under <employees
* Even in the Hierarchy, each item node could have its children divided into categories
  + <item id=”34”>
    - <comments>
      * <item id=”F5” />
      * <item id=”T32” />
    - <categories>
      * <item id=”2B” />
* The Hierarchy may not actually be that helpful
  + A collection of all descendants (by a particular field), can be made recursively
  + I don’t think this will be much slower than using an indexed Hierarchy
  + The recursive method will definitely be faster for only going down a few levels (rather than all levels – for all descendants)
* The main purpose of the Hierarchy is to quickly create a list of all descendants of an item so that this list can be used for searching under an item
  + For this purpose, all kinds of items need to be jumbled in together
* Perhaps the <children> field should jumble together all other fields
* <subitems> (of a User item)
  + <comments>
    - <item id=”F5” />
    - <item id=”T32” />
  + <employees>
    - <item id=”2B” />
    - <item id=”7N” />
  + <actions> (actions performed by this User)
    - <event id=”e5BV2” />
  + <items> (items owned by this User)
    - <item id=”2BV” />
* Here I’ve replaced <children> with <subitems> because <children> might be used as a specific field actually intended for biological “children”
* All the sub-categories of <subitems> would appear as sub-pages of the “Subitems” page
  + The Subitems page itself should show a list of the available kinds of subitems for this item
* External links should all be stored inside Link items
  + links that appear in non-Link items should have two parts: a link to the Link item, and the external link
  + this way, all web pages linked to will be made into ZuthaNova item
  + Link items on ZuthaNova can have a comments page and have investments put on them
* If I found descendants recursively,
  + I would be able to allow infinite loops
    - When getting the children of an item, it would be a simple matter of checking whether those children are in the list of selected item yet
    - The recursive process would only continue with new items that have not been processed yet
    - E.g. Item (A) has subitem (B) and (B) has subitem (A)
      * Get all subitems of (A)
      * With each of these, of which (B) is one, get all subitems
      * With each of these subitems check if it is in the found list yet
      * For (B), the subitem (A), has already been found, so (A) is not added again to the found list, nor is (A)’s subitems checked again
  + I could find descendants by more than one field
    - For a User item, I could find
      * All descendent comments (...comments about comments about the User)
      * All descendent employees (...employees of employees of User)
      * All actions done by descendent employees
    - For an item like “Linux”, I could find, and search within
      * All descendants by the <subitems> or <comments> fields
  + I could incorporate Search items into the searchable hierarchy
    - By Search items I mean items that don’t store the results of the Search in the item, but find them in real time when the item’s children are requested
    - Search items would still have static comments on them
    - When searching for descendants of “J.K. Rowling”, the Search item “Books by J.K. Rowling” might come up
      * Instead of checking an internal field for the subitems of “Books by J.K. Rowling”, the subitems will be found in real time by the query defined by the Search item
* It could be quite relevant and useful in some contexts to allow investing in a negative way
  + Sometimes people will want to express how much they hate or disagree with something
  + I could allow 2 separate pools for investing in, for every item – a positive one and a negative one
  + Positive investments would give a percentage to item owner
  + Negative investments would not give anything to item owner, but perhaps should have a tax taken out instead
  + Actions done by users can have positive or negative investments in them
    - This will allow employers to be notified of the Actions they should be taking note of
    - Actions that are strongly voted negative or positive should be emphasized
    - Anyone wanting to get an idea of how well received the actions of a particular user are, can check the positive/negative weightings of his Action items
  + Negative investments should receive a larger overall tax than positive investments
    - Hopefully this will create a disincentive to use negative investments spitefully
    - If the first person to make a negative investment in an item was not taxed, there would be no disincentive to putting in a huge negative investment out of spite, because you would know you can get it all back when you chose.
* There are list fields, like <employees> that it would make sense to store inside the item as well as inside a child category item
  + The benefit of storing fields like <employees> or <authored\_works> inside an item are semantic
* Search items should be set up so that requesting a new subitem of them automatically fills in the fields that would allow the item to appear in the Search item’s results
  + The definition of a Search item might be:
    - Template=”book”
    - Data/genre=”fantasy”
  + If a new subitem was requested for this Search item it would be forced to be of type: Book and have genre=”fantasy”
  + Actually only certain items need to allow this
  + There are some items that should remain read only Search items
    - Adding to the results of such items should be done by separate means
    - E.g. a Search item containing actions by any User in the Employee Hierarchy of a particular Employer
* Functionality that the architecture must allow for
  + Search items associated with the current item, optionally with a specified number of degrees of separation
  + Find items linked to the current item through a particular field, possibly through multiple degrees of separation
    - Find all employees (of employees (...)) of an employer
  + View a set of items broken up into several different kinds of groupings
    - E.g. Books by Genre, Books by Author, All Books
    - E.g. Organisations by Region, Organisations by
    - These groupings should be made complete automatically
      * Books by Genre should automatically contain all the genres that have Books linked to them, and no more
  + View different kinds of subitems separately
    - i.e. Comments, Categories
* There are sections of the site that need a lot of manual labour to organise
  + e.g. the Discussion hierarchy under a Book or Movie item
* sections of the site that can be described semantically should be almost exclusively organised automatically
  + e.g. “Books by J.K. Rowling”
  + e.g. “Fantasy Books”

# Thu 25-Feb-10

* I want to be able to set competitions that have a deadline, and use my site to take advantage of the wisdom of crowds
  + For competitions that will on some date have a result (such as the result of a soccer match), I need to decide what should happen to the investments in the competition’s Candidate items.
  + Once the competition is over, investors are likely to want to take their money out of the Candidate items they invested in
  + It would be ideal if the investments were left in the Candidate items so that the result of the crowds decision can be exhibited
  + The “prize” for picking the correct Candidate should require the investments to remain in the item
  + The prize system should work for the real money system as well
  + It would be great if I could allow anyone to create Competitions
    - If the prize system had a chance of draining money from the treasury, this may not be practical
    - Users could create Competitions that have an obvious result, and make money from the treasury
    - The prize system should not allow this
  + Competitions such as who will win a sport match should have an investing deadline, after which all investments are locked in
  + Investments in such Competitions should be then frozen for ever after to exhibit the results of the competition
  + When the result of the Competition is known, all the zuth/money in the losing Candidates should be bundled with the money in the winning Candidate and divided up among investors in the winning Candidate
  + This prize system does not seem to make any use of the “filling the vase” based investment system
    - The benefit of investing early should only be relevant if you decide to pull out of the Competition before it is over
    - Actually I’m not sure I like the idea that someone can invest early in a Competition and then take away lots of other investors’ money just before the Competition closes
  + How should the winnings be divided?
  + Perhaps this kind of “competition” should use different investment dynamics entirely
    - I could remove the ability to withdraw your investment
    - There would then be no need for the “filling the vase” concept
    - The value of everyone’s “wager” would be recorded, and no alterations to their value would occur
    - This is just like a normal betting process
    - The main difference would be that winnings come from and are limited to, the investments of the losers
    - By ensuring that the “bookkeeper” (in this case the ZuthaNova.com site), cannot lose money, it is safe to allow anyone to set up “Bets”
    - I could potentially take a small percentage of every bet, just like the tax on investments
      * This wouldn’t actually be necessary though, unlike the tax on investments
      * It could be a beneficial statement of good faith to make no profit from these Gambling items (in the real money system that is)
      * In real betting, bookkeepers do a lot of calculating to make sure they get profit
      * My system will be a certain way to avoid losing money
      * Some people may become suspicious that they are not guaranteed a specific amount of winnings if they win
      * If I make it clear why I am choosing to design the prize system this way, and also take no profit as a measure of good faith, hopefully this will assuage people of their suspicion.
* There are two very different kinds of Competition
  + Competitions that are opinion based, such as “what is the best romantic scene in any movie?”
    - In this case the “competition” is between candidate items
    - This kind of Competition could be instead called a Poll
  + Competitions that will have a result, such as “who will be the winner of sport match?”
    - In this case the “competition” is between investors to guess the right answer
    - This probably should not be called a Competition, but perhaps something like a “wager” or “gamble” or “venture” or “speculation” or “bet”
* I want to add a kind of Gamble item where Users bet on the result of an event, where the result could be any of a continuous range of options
  + E.g. the margin in a sporting match
  + E.g. the opening day Box Office revenue of a new movie
  + These kinds of bets should be confidential
  + Betting should be closed some time before the result will be discovered
  + The Crowd’s prediction can then be published in advance of the event occurring
  + The prize system for this kind of Gamble could be that the closest guesser receives all the winnings
  + There could be a range of prizes, like 1st, 2nd, 3rd... prize of decreasing amounts
  + I’m not sure whether it would be more enticing to people if a really small proportion of betters won, or if, say, the top 10% closest guessers won something
* The investment system would better make use of the “wisdom of crowds” if investors were not as influenced by previous investors in making their decision
  + If a few User decide a particular item is good and invest in it, then subsequent investors will be influenced by the earlier investors to think better of that item
  + This will be particularly detrimental for situations where the first few investors may be investing in the item because of special interest in seeing that item become popular, rather than a genuine belief that the item deserves the investment.
  + Items whose worth is hidden, such as “unpaid commercial” items, may make better use of the wisdom of crowds because the secrecy of people’s “votes” will increases diversity, and this will increase the quality of the collective decision.
  + It would be good if I could take advantage of these benefits while still retaining the main purpose of the voting system, which is to display the results of the collective decision
  + I could present the results of the collective decision in instalments
    - I could make information about previous users’ actions available only after a delay period
    - When a person is deciding whether to invest in an item, they would know that its current worth is out of date
    - When an item is first starting off, there should be an opportunity for many people to decide it is worth investing in, before their decision becomes influenced by knowledge that others have invested in it.
    - When new information becomes available to someone that causes them to reduce their valuation of an item and want to sell their investment in it, their decision should not immediately influence others to do the same, based only on the first person’s evidence.
      * Say a major flaw is reported in a product through an obscure news source
      * Some people might hear about it and decide to sell their investments in the product’s associated item
      * If their actions were known immediately, other people might see them selling and decide to copy them without knowing why they are selling
      * If the sellers’ actions were not known immediately by everyone else, there would be no reason for others to copy them
      * The information source of the sellers may turn out to be false, or at least, most others never find out about it
      * As much as possible, people should not be induced to sell based on the actions of others
      * If some people know some information asserted by an obscure news source that a product is flawed, then this knowledge’s influence on the value of the product’s item should be weighted on how many people know it.
      * The aim is to aggregate the information known by lots of people, without the information known (or believed) by some, unduly influencing the outcome.
    - Technically, I could implement this delay by simply not running Investment event item immediately
      * The event items would be collected invisibly for a period of time, then run all at once when the time comes to reveal the results
      * This cycle would repeat continuously
      * The period of time between revealing item worths could be something like a few days
    - It might make sense to vary the interval somewhat randomly
      * If there was a known “refresh” time for all items, it would create a big event
      * People would focus on the “refresh” event and try to find some way to take advantage of acting when it occurs
      * I’m not sure what harm this might do, but I have a suspicion that it would be better if the crowd did not know when the newest aggregated information would come out.
      * Each time a refresh occurs, the time before the next refresh for that item would be decided randomly (within bounds)
    - What should happen when users sell between refreshes
      * The value of their investment could be based on the last refresh
      * This would mean sell events that affect the behaviour of the buy events, would occur before them, even if the buy events were requested first.
      * The only problem is, if the sell events happen immediately, they will affect future sell events
      * It would be necessary to inform users of the value of their investments if they decide to sell them
      * When sell events occur, they will affect the value of other users’ investments
      * Perhaps sell events must occur at the refresh moment as well
      * It wouldn’t be too big an issue for users to have to wait a while for their money
      * Actually, some people would be quite annoyed by this, particularly if they had to wait up to several days
      * It would also upset people if they decided to sell, believing they were making a profit, but when the sell event was run, it was run after many other sell events and was therefore worth less than when the sell was requested.
    - I think it will be necessary for the events to be run in real time
    - Only the aggregated “worth” of the item can be realistically withheld
      * This means that some people will be able to glean information about the progress of the item’s worth by watching the value of their investment
      * Anyone who figures out the maths will be able to calculate the worth of the item in real time by watching how the value of his investment changes.
      * If only a minority of people bother to do this, the effectiveness of withholding information from the masses will be retained
      * There may be a risk that someone will publish their calculated item worth to the public
      * There would a strong incentive to keep this information secret for the discoverer’s own benefit, but I cannot rely on this
* I need a convention for when information should be stored inside the item and when it should be stored in subitems of the item
  + Only data that is directly about the item itself should be stored in the item
  + Information about associated ideas should be in subitems
  + Most lists should be stored in subitems, especially if they can be of arbitrary length
    - This is the part I’m most unsure about
    - Should <employees> be stored in the User item?
    - Should “books written by <author>” be stored inside the Person item
    - Should “books published by <publisher>” be stored inside the publisher’s Organisation item
      * This list at least would be too long to put inside the item
      * It would make sense to be able to view the list of books published by a publisher as a list ranked by worth
* There may not be such a big problem with programmatically working with associations that are linked through Predicate item like “books written by J.K. Rowling”
  + A Person item “J.K. Rowling” specifies that the item containing books written by her is the “books written by J.K. Rowling” item
  + A query algorithm looking for the <authored\_works> data for “J.K. Rowling” would find a reference inside the “J.K. Rowling” item directing the query engine to look in the item “books written by J.K. Rowling”
  + If I base my architecture on this approach, I think I’ll find it will open up a lot of new possibilities
* I think I’ll adopt the convention that any lists of items should be stored as subitems
* Any fields that link to another item should correspond to a subitem of the linked item
  + Item (A) specifies item (B) as its <employer>
  + Item (A) is added to the “employees of B” subitem of (B)
* For one-to-many relationships, this convention makes sense
* For one-to-one relationships?
  + Fields like <spouse> only make sense as fields in both items linked, never as subitems of one
* For many-to-many relationships?
  + A field like <friends> would contain a list
  + At the other end of the link, the <friend> would also be one of many
* I could store both sides of all links fields inside the item
  + This would replace the need for an index on these linked fields
  + It would be worth checking whether there is some benefit to using Berkeley DB XML’s built in indexing system over making it myself in this way
  + I would still use Berkeley DB XML for many indexes, I just wouldn’t need to index fields like <author> if I already have, for every Author, a list of all items that have it as an <author>
  + This data wouldn’t necessarily need to be visible inside the item
  + Books by a particular Publisher might actually be viewed via a Search item called “Books by <publisher>”

# Fri 26-Feb-10

* Investments in events should all go to the event owner and not be sellable
  + Their purpose is to praise the owner for this action
  + It should be impossible for a user to invest in his own Action/Event item
  + It would be possible for someone to use an alternate user to invest in an Event/Action item owned by another of their users
  + The tax on investments needs to remain. This will create a slight disincentive to invest in your own Event item, because you will still loose some money.
  + A small disincentive may be enough
    - I can’t imagine that there would be a lot of people that would corruptly invest in their own actions to give themselves recognition
    - They would have to invest in actions that will be considered good by others as well
    - There should be major consequences if someone is discovered investing in the Actions of their alternate users
  + The purpose of investing in Event items is two-fold
    - It is a gift to a user who has done something you approve of, or a bonus for the same reason if it comes from an employer
    - It is a way to broadcast to others what you think was a sensible action
  + I am still breaking a core principle here
    - I decided that currency must not give value by being flowing through an item
    - it must only give value in one way at a time
    - currency can either be liquid, and available for investing, or it can be invested, in which case it gives worth to the item
    - if I allow investments in Events to give worth to the event while also giving the currency to the event owner, I am breaking this principle
  + I could divide investments in Event items in half, giving half to the owner, and leaving half in the item
    - The zuth left in the item should still not be sellable, as Events are one off and their value does not evolve with time as Companies do, for example
* Items that do not represent a concept that evolves with time, should not allow selling of investments in them
  + Events are static
    - They have already occurred
    - There is no need to allow people to modify their decision about whether they believe it was a “good” event
    - Actually, there may be reason to do this in some cases, but there is another reason to not let users sell their investments in “static” concepts
    - A user might invest in an Event to show they approve of it, but after the Event is long gone, they will want to retrieve their money
    - I want investments in such things as Events to remain as a historic record
  + Items that are candiates for a Decision become obsolete once the decision has been made
    - Investments are votes on which decision should be made in a particular scenario
    - Such investments should not be sellable, because the vote should be made for the record
* There should be a way for employers to praise their employees by giving them bonuses
  + I could make the User item like an Event item
    - Investments in a User item would be divided in 2
    - Half would be given to the User
    - Half would remain in the User item as a testimony that the gift had occurred
  + The incentive to give bonuses would need to come from up the line
    - A high level employer should give bonuses to an emloyee if his “team” is producing good work
    - Ultimately, the incentive to encourage good work needs to come from me, and (hopefully as losely as possible) I need to define what is considered “good” work
* The items users create should be included in what defines a worthwhile employee
  + Indeed, initially, the only thing to go on when recruiting a level 0 user is the items they have created
  + Employers should be praised for putting a team together that creates popular items
  + It would be useful to quantify success in this area more definatively
  + It may be realistic to quantify objectively how well a user’s “team” is creating popular items
  + It would be far more difficult to quantify how successfully a user’s “team” is doing at making constructive edits to the site content
  + Other actions that should be valued are comments that are constructive but may not receive anywhere near the amount of investment that simply creating the item “Harry Potter” would glean
  + Users should be praised for answer people’s questions
    - I could use this architecture to do what services like Answers.com and Yahoo Answers do
* There should be many different ways in which users can contribute in order to be employed and earn an income
* Ways to earn money
  + You can earn money by creating popular items, because a proportion of investments will go to you
  + You can earn money by investing in undervalued items and then selling when the value of your investment increases because of subsequent investments by other users.
  + You can earn money by being employed and rising up the ranks
    - You can gain credibility by:
      * Creating popular items
      * Giving constructive feedback about the running of the site
      * Answer user’s questions to their satisfaction
      * Improving the organisation of the site
      * Finding and posting bugs or improvements to the code
      * Making new Templates and Themes and other technical aspects of the site
      * Convincing companies to pay a subscription to have their products show their worth on the site
      * Recruiting new people to the site
        + In order to know who is responsible for recruiting a new user, the new user would have to mention the person who recruited them
        + Actually this cannot work because there is no way to establish that a new user is a new person
        + The best way to benefit from recruiting a new person who contributes actively to the site would be to employ them and reap the benefit for your own recognition of having an employee who does good work.
      * Finding and employing users who do a lot of the above and encouraging them to do more of it

# Sat 27-Feb-10

* The fact that there is no objective way to determine the value of an item means that to a large extent people’s actions will be dependent on other people’s actions
  + This dependance will reduce the accuracy of the investment scheme in valuing items according to what the average person actually thinks
  + Instead, valuation will be weighted towards what, at any one time, people believe other people are valuing.
  + There is a mechanism of my site which may reduce the harm of this fact compared to the impact this principle has on the stock market
  + The benefit of investing in an item reduces as it increases in worth because of the dynamics of the “filling-the-vase” model
  + Therefore there is an increased incentive for investors to go against the crowd because there is greater benefit in picking valuable items that have not yet been given a high worth
* There seems so much in common between Events and normal items that it might make sense to go back to incorporating Events as just another kind of item
* Each Template should determine the dynamics of investing in that item
  + It should be determined whether investments can be sold after they are bought
  + The proportion of investments that should go to item owner
* Templates created by 3rd parties will need to be verified and locked before they are available for use in new items
  + Otherwise it would be possible for users to create Templates that allow undesirable things, and make these undesirable items immediately
  + Undersirable actions would include setting the proportion of investments that go to owner as 100% and then creating a popular item with this dodgy template so that people get tricked into giving money to the owner.
* There is still a question about how to deal with the database when the number of items becomes really huge
  + Even though View Events are no longer recorded, Action Events will still make far more than half of all items in the database
  + It would still make sense to put Events in a separate container
  + If I’m going to start using multiple containers, I need a very efficient way to combine the results of searches on all containers
  + I also need to know which container to look in when an item ID is looked up
  + I could use conventions in the ID system which would allow identification of the appropriate container
* I could let the first digit of the ID represent the container to look in
  + If an ID must begin with a number then I could distinguish between IDs and words
  + This would allow me to have both of these URLs:
    - ZuthaNova.com/0HB4/the-item-name/page
    - ZuthaNova.com/login
  + In order to cater potentially for more than 10 containers, if the first digit of the ID is 9, then the first 2 digits are used to identify the container
    - This could be extrapolated: if the first 2 digits are 9 then the first 3 digits are used to identify container etc.
    - This would mean that the ID would increase in length by 1 for every 10 containers used
    - Since a container should fit billions of items, I shouldn’t have to worry about this as a problem
  + Event items could all have an ID starting with 1 representing container number 1
  + Initially all normal items would initially have an ID starting with 0 representing container number 0
  + If either of these containers ever filled up, I could add a container number 2 and start a new line of IDs beginning with 2
  + Once there are more than 1 container containing normal items, it would be necessary to do all searches on both of them and merge search results
    - I don’t think this would be too difficult since XQuery is designed to allow for multi-container queries
* I like the idea that users have complete autonomy over what they do with their own employees
  + A user’s boss should have no say over which employees he picks
  + The boss can and should advice his employees about the way they deal with their employees
  + The boss should not be able to do an action to affect his employees’ employees
  + A boss should promote, demote, sack, and employ employees based on how well they manage their own employees, but he should not directly interfere
  + A boss may demote an employee (A), employ someone else (B) to take his place and then ask (B) to employ (A)
    - He may not make (A) an employee of (B) by himself

# Fri 5-Mar-10

* My data structure should combine graphs and hierarchies
  + Specific fields like <publisher> and <author> represent links that form a graph
  + <childitem> and <parentitem> fields will be consistent throughout all item types and represent a hierarchical relationship
  + Part hierarchies and Class hierarchies will both be represented with these same fields
  + Actually perhaps it would be useful to define even hierarchical relationships with specific fields
  + There will be hierarchies that are defined by specific fields, like <employee>
  + An “Animal” may be classified at various levels of abstraction
    - A Red Kangaroo might be classified as a Kangaroo or more abstractly as a Marsupial
  + It would be useful to be able to easily generate a list of all Kangaroos or all Marsupials, without having to transverse a hierarchy to get all the leaf nodes
    - Actually the list of Marsupials should probably contain the next level of abstraction like Kangaroo and Bilby, as opposed to containing all the specific breeds of Kangaroo and Bilby and other kinds of marsupial
    - Perhaps it does make sense to arrange such data in a Class hierarchy

# Tue 23-Mar-10

* Social Networking features
  + Ability to tweet from website
  + Users can have friends
* Comments can be filtered by owner level

# Wed 24-Mar-10

* Option should be available to access every item in RDF format
  + RDF data would be generated automatically when it is requested
* Instead of having a single <template> field, items could have a list of <ISA> values (“is a”).
  + I thought of this idea after reading about perl’s method of multiple inheritance
  + Multiple inheritance could replace the idea of “roles”
  + J.K. Rowling could have <ISA> fields: “Person” and “Author”
    - The fields defined by these two templates would be combined
      * The Author template would permit fields like “pen name”
      * The Person template would permit fields like “spouse”
    - When linking “Harry Potter” to an author, the criteria for the selection would simply be that the linked item must be an “Author”

# Mon 26-Apr-10

* I’ve decided to change Databases to a Graph Database
  + I think I’ll choose InfoGrid database, but there are a few reservations I have about it; mainly its apparent lack of an indexing system
  + It might be practical to use a separate indexing system with InfoGrid such as Lucene.
  + Neo4J, another Graph Database, integrates Lucene.
  + HyperGraphDB has integrated fullfledged indexing and an advanced querying mechanism which utilizes the index.
* Things I like about InfoGrid
  + It uses Viewlets to transform data for various kinds of output including HTML.
  + It has a sophisticated system in place for making the database Distributed.
    - Given that it will be a long time before I may need to consider a Distributed design principle, I probably shouldn’t get hung up over this
  + It seems to allow easier import and export, though I can’t see why I would be prevented from importing/exporting data from the other databses.
  + I like the feel I get about the developers’ mindset. I don’t know how to make this less vague, but I like their approach. On several occasions while reading the documentation I was struck by how perfectly they had implemented an idea I thought I had invented myself, most noticeably the concept of Roles in the context of relationships between nodes.
* Things I don’t like about InfoGrid
  + It doesn’t appear to have an Indexing system, so I wouldn’t be able to even lookup items with a particular name

# Tue 27-Apr-10

* There needs to be a way to hinder the direct trading of Zuth
  + This is because if it is easy to give Zuth to other users, it will be easy for it to be traded for real money.
  + If the only way to trade Zuth for money was by investing in a particular item, then there would be a disincentive to sell yourself out for an unworthy item
  + In the new system, however, it would be easy to transfer Zuth directly to another user.
* It could make sense to withdraw a tax at the time of selling an investment.
  + This would give some additional disincentive to sell.
  + It would also mean the “winners” would be taxed more.

# Tue 20-Jul-10

* I want to integrate Freebase data somehow into ZuthaNova
* I am considering abandoning the idea that an item is owned by someone
  + This would make it easier to integrate with other data sources including Freebase
  + I could deeply integrate my system with Freebase then; merging all its data with mine
* Ownership of an item is supposed to entail rewards when that item becomes popular
  + This creates an incentive for people to find concepts that they think deserve to be popular
  + This system is open to being exploited by users who algorithmically import massive amounts of data into the system, thus gaining rewards for all the chance items among that data that become popular
  + A user should have to invest something in an item to show he thinks it worth others investing in
  + Removing the direct reward for ownership of an item does not remove the incentive to create new worthy items
    - If someone wants to invest in an item that doesn’t exist, he has an incentive to create it, because being the first investor is a massive advantage
* Some types of items inherently need to have an owner such as Comment and Article
  + Such types will be defined as such and will obtain their “owner” property at creation time
* Every item needs to have a property defining the “creditee” or “beneficiary” of investments in that item i.e. the person who receives a proportion of every investment in that item
  + For Comment and Article items, this would be the “owner” or “creator” or “author” or whatever I decide to call the field
  + For Book items, “author” would be the beneficiary
  + For “Web Article”, “author” would be the beneficiary
  + for “Movie” items, “Studio” would be the beneficiary
  + There would be times when it would not be clear who should be the beneficiary for an item
  + Perhaps the “beneficiary” field should be empty unless there is a specific user who has been authenticated as representing the item in question
  + I am undecided as to whether revenue should accumulate for items that have not yet been claimed
    - If I was to keep track of revenue for unclaimed items, I would probably need to create a temporary user to receive the revenue
    - Perhaps I could always keep track of revenue for an item in an item-specific account item.
    - When someone authenticates himself as the deserved beneficiary of that item, they can withdraw from this associated account
* I could use Freebase solely with the javascript front-end of my website, to auto-fill fields for newly created items
  + This is a feature that can easily be added at a later date, which is desirable
  + As long as it is easy to access my database programatically, anyone can design systems to get data from elsewhere and put it into my database
  + I would need to figure out a way to reliably match the results from a Freebase search with the underlying items in my own database.
  + If I wanted to use javascript to find the author of a book that is being added to my database, getting the name of the author would not be sufficient. I would have to find the corresponding item in my database representing that author. If there are multiple authors with the same name, this would become difficult.
  + If the author didn’t already exist in my database, I could create it automatically, possibly pulling some data from the entity just located in Freebase
  + The fact that items don’t have to have owners anymore would make the process of auto-adding items much neater
  + The main benefit of this paradigm is that the Freebase integration can easily be implemented later regardless of how I design my architecture
* It could be useful to integrate Freebase and other data sources using the Infogrid Probe Framework
  + I am not yet really sure how this system works so I don’t know how well I could merge data across multiple databases, but that appears to be the intention of the Framework
  + I would want to be able to match items in my database with entities in the Freebase database, and update my fields from the Freebase data in real time
* Whenever an item is created which is of a universal type (defined outside the scope of ZuthaNova) such as a Book or Movie, it should be linked to a corresponding Freebase entity
  + Extension: whenever an item is created which is the sort of entity that would be found in a database of things like Freebase, it should be linked to as many of such databases as possible
  + If the created item does not yet exist in Freebase, it should first be added to Freebase and then linked to it

# Sun 15-Aug-10

* It should be possible to create isolated sub-graphs that are owned and completely controlled by a particular power hierarchy
  + This would allow companies to create their own private (possibly publically hidden) subgraph.
  + The company would authenticate ZuthaNova users as their employees and allow only these users into the company controlled power hierarchy
  + Only this hierarchy would be allowed to edit the company-specific subgraph
  + Links to outside entities would still be policed by high level users in the public power hierarchy
  + Such public links could be permitted to be one way. i.e. link from the company graph to the public graph, but not the other way around
  + This whole system would be useful for governments and other groups as well
  + It could be used as a feedback and idea generating forum, and as a way to internally and/or publically organise and present the organisation’s information

# Wed 25-Aug-10

* Allow users to connect their Zutha.com account to many other online accounts like Facebook, Google, OpenID etc.
* Allow users to share on their various connected social accounts, the items that they have invested in or starred
  + Users can star an item if they like it but can’t afford to invest in it
* Create addons for browsers that allows in-browser access to a hierarchical arrangment of all starred ZuthaNova items
  + The user’s own starred ZuthaNova items would be synced with the browser’s bookmarks for efficient access
  + Access to the entire ZuthaNova database could potentially be allowed, but it would involve realtime API access and be slower
* I want to make my power hierarchy take advantage of some of the principles that I think make startups more effective than big companies
  + The current system is too restrictive
  + The current system is too individual-centric
  + I’m thinking of making each node in the hierarchy a group instead of an individual
  + The core unit for constructive contribution to the site should be a collaborative group
  + It should still be possible for a group to consist of just one user, though
  + The main benefit of a hierarchy is that I only have to manage a small number of people directly below me, and wages can and power can be filtered down the hierarchy without any one person having to be responsible for too many people at once.
  + The main obstacle for adopting a model of many independent groups, is that the wages are coming from a central government-like source
  + If I could think of some natural, evolutionary type way to distribute new currency to those that win out in some kind of competative game, that would be ideal
  + I do need a power hierarchy to settle disputes and prevent vandalism, though
  + The assumption behind a power hierarchy seems to be that each worker needs a specific superior entity to watch over it in order to prevent him mucking up
  + But in reality, it doesn’t seem that having a direct superior watching over each worker prevents them from poor behavior or performance
  + Perhaps the way to escape from a hierarchy is to use a model in which misbehavior by workers is proactively sought out by a large group of superiors, and finding and punishing such misbehavior is rewarded by the higher level superiors in the same way that finding and rewarding good behavior would be.
* The power structure could be an arbitrary quantity of independent collaborative teams
  + Each group can be promoted or demoted by any of its superior groups
  + The highest level group (the ZuthaNova admin team), would raise a small number of the best teams to the 2nd highest level
  + Each team may be rewarded with promotion by any higher level team for doing such things as
    - Finding and promoting good lower level teams
    - finding and punishing misbehaving lower level teams
    - resolving conflicts
    - creating good items
    - general maintenance on the site, including the semantic relationships between items etc
    - answering questions well
    - giving good feedback to lower level groups, site users, higher level groups or the site admin
  + A team may be punished with disolution (meaning loss of all editing privileges), demotion or simply constructive criticism for such crimes as
    - apparent favoratism of lower level groups
    - demoting a group without merit
    - failing to constructively resolve a dispute with another group
    - vandalising the site
    - promoting a group that has been comitting any of these crimes
  + It is likely that vicious conflicts will arise with this model, as it is inevetable that different people will have different opinions about such things as whether another group deserves to be demoted for a particular crime, how a domain of the site should be organised, etc.
    - It would be useful for a higher level group than those involved in a conflict to be able to suspend privileges of those groups involved in the conflict until the conflict is resolved
  + Groups/Teams will be paid a wage determined only by their privilege level
    - The collective wage will then be divided up among the users in the group
    - It should be possible for a group to divide up its income disproportionately among its members based on how much each user is contributing
    - In order to facilitate such decisions within a group, there should be a framework for groups to elect a leader if they chose, and to vote on such things as how much income each member deserves
  + Every user in a group would have the same editing privileges as the group
    - Perhaps it would be useful for a group to create its own internal power hierarchy
    - This would allow the hierarchical structure of governing to be tested against the independent small team model
  + I want to provide as much flexibility as possible in order to allow the evolutionary algorithm to do its work
    - I will specify in a lose way the kinds of things I want my army of teams to achieve, and then let evolution determine the best way to achieve them
    - I imagine my influence will be blurred by the time it reaches the lowest level teams
    - Therefore, I need some kind of more natural measure of a team’s success
    - The most obvious measure to link a team’s success to is the success of the site itself
    - If I pay the top teams a proportion of the site’s profits, then they have a direct insentive to make the site more profitable
    - It is better if I let evolution work out the best way to make my site profitable than try to specify my own metric for that end

# Fri 27-Aug-10

* Search engine plugin to find the category of each link in Zuthanova – like Xmarks
* Fund making data public and organising it on Zuthanova
  + This could be such data as government statistics, or statistics from various publically or privately funded research projects
  + I would pay, through ZuthaNova, to have this data made public, and as part of the deal, links to the data would be placed in an appropriate place in the site

# Fri 3-Sep-10

* When a Relationship is created, a link should be added to both item pages to the other item
  + E.g. “Robert Jordan” is added as the author of The Wheel of Time series
  + A link is added to Robert Jordan item such as “works by Robert Jordan”
  + This link should go to a search page showing everything authored by Robert Jordan
  + The search page might automatically be set to group by type
  + Grouping by type should be an option on any page, but it should be the default in this situation

# Sun 5-Sep-10

* I’ve decided on a paradigm of focussing on relationships between concepts rather than metadata about them
  + While trying to design my schema I encountered a conflict between a desire to extensively use InfoGrid Types to model allowed fields and relationships on items, and the desire to retain simplicity of the core schema and allow the greater flexibility of a user defined secondary schema, which would not have as direct an impact on the functionality of items.
  + I’ve realised that my second desire makes a lot more sense
  + The paradigm that will make it work will be a focus on relationships and largely an abandonment of metadata like date\_of\_birth and cost\_of\_construction.
  + Metadata like this is much better left to third party services (which can be linked to directly from each item)
  + For example I don’t need a “population” field for items of type “Region” which have a user defined category of “Country” – I just link to external websites representing that country and such metadata as population (plus a lot more) can be found there.
  + I’m not trying to make an encyclopaedia, or to be the sole database of the world’s knowledge. ZuthaNova is intended to be a way to find and compare concepts; other services can provide information about the concepts.
  + With this approach, I can make most of my type hierarchy impotent in terms of field functionality and relationship restriction, but thereby allow it to be extended by the community.
* There will probably be times when it will be desirable to make a user-defined type into an internal type
  + E.g. The Type “Organisation” might have a user defined sub-type: political party. Initially this may not have any special fields or relationships beyond what the “Organisation” type has, but later, a “Policy” Type may be added and it would be desirous to make a relationship between “Policy” and “Political Party”. A relationship between “Policy” and “Organisation” would not be restrictive enough. Therefore a new type: “Political Party” would need to be created and all the “Organisation” items currently with the user-defined type: “Political Party” would need to be updated to have the internal type: “Political Party”.
  + It would be ideal if there was a significant conceptual overlap between user-defined types and “Internal Types”
* Perhaps I can use some of the principles of an earlier model I had envisioned back when I thought I was going to use an xml database
  + Then, I hadn’t really conceived of using multiple “Types” of item, apart from the distinction between “item” and “event”
  + All the semantics were going to be described be the hierarchical arrangement of items
  + Perhaps I can keep the idea of arranging most of the semantic information in a lose way, just by using a graph of categories
  + Only some additional information would be desirable to describe more structuredly – such as the author of a Book or the brand of a product.

# Mon 6-Sep-10

* When creating an item, the name textbox should connect to the Freebase autocomplete feature as well as check for existing items in ZuthaNova
  + If the item doesn’t exist in ZuthaNova, but does in Freebase, the new item should be auto-filled with data from Freebase
  + If the item already exists in ZuthaNova, any new information from Freebase should be filled in. The prospective item creator should be notified that the item already exists, but the certain fields will be updated with new information from him and Freebase if he continues.
* In order to deeply integrate with Freebase, I need my data model to align with it pretty well
  + I quite like Freebase’s data model, so this should be practical
  + The only thing I can’t see being practical, or necessarily desirable, is the degree of realtime editability of the model
  + I think it would actually work quite well to have my model slowly develop, following behind Freebase’s most stable structures
  + I will advertise major additions to the model and try to make these events significant
    - Major new additions to possible item types would be a great opportunity for investors to make money, by being quick off the mark to invest in popular items of the new category
    - Making each incremental change to the model a big event, could be an effective way to keep enthusiasm alive – it would similar to the concept of bringing out regular patches in World of Warcraft.
  + In Freebase, Types are first class citizens and are directly editable as objects
  + My understanding of InfoGrid so far is that Types are underlying data structures which don’t manifest themselves as accessible nodes in the graph
  + I want to have this feature of Freebase that Types are viewable as Objects, so I will shadow every Type as a specific item which has a unique 2 way link with the Type
    - Every time an Item is blessed with a new Type, it should be linked with the associated Type item
* Categories need to be distinct, but related to Types
  + Categories may potentially be seen as prospective future Types
  + There will still be many categories, though that would not be applicable to making into a Type
  + The category item itself would not become the new Type item
  + A Category item will define the Types of item that are allowed as subitems
  + A Category item “Political Parties”, might specify that only items of type “Organisation” are permitted as subtypes
    - Organisation is chosen because it is the most specific type that exists at this moment that covers the desired subject matter to be captured by the Category
  + Later, the Type “Political Party”, might be added
    - The Category item “Political Parties” will then be able to make its subtype requirements more restrictive and require that all subitems be of type “Political Party”, instead of just “Organisation”
    - I would need to find a good way to update all the subtypes of the “Political Parties” category to have the Type “Political Party”
* Actually maybe every Type item could also be a Category item
  + When a new Type is added, the relevant category item would be made into a Type item as well, and linked to the underlying Mesh Type
  + All existing subitems of the Category could be made into the new Type
  + I need to find out what InfoGrid does when an item is blessed with a subtype of a Type it is already blessed with
    - It would probably be cleanest if the supertype is overridden/replaced by the new type
  + Items should be linked to every Type item that applies to them, including supertypes of all their Types.
* One of the reasons I’m hung up on providing functionality to store a lot of semantic data with each item, is that it would help in doing advanced searches and generally in locating a target item
  + I could potentially offload a lot of the data description responsibility to Freebase (or potentially other semantic data stores)
  + I could integrate the Freebase API directly into semantic searches for items on ZuthaNova
  + There may be fundamental relationships that I would want to maintain in my own database, like Author and Genre, say, because I would want to show Book items as direct Subitems of an Author item, rather than having to do a search to find all items with a particular Author and then sort them by their Worth.
    - For core relationships like this, I would want the efficiency of lookup that could only be achieved by storing the relationships in my own graph database
    - For more obscure metadata, that might still be desirable to search on – like “number of pages” – I can search Freebase, and filter by the resultset.
  + In order to use the resultset from Freebase, I need to have reliable one-one links between Freebase Objects and ZuthaNova Items
    - I can do this by storing the MID of the corresponding Freebase Objects for each item within ZuthaNova
    - I could also try to add an external ID to the Freebase item, linking to the corresponding ZuthaNova item, every time an item is created in ZuthaNova that has a corresponding Object in Freebase
    - For reliability purposes, though, I should store an index linking Freebase MIDs to ZuthaNova IDs

# Wed 8-Sep-10

* Monetization Progression
  + Manually connect as many items as I can to affiliate programs
  + Start paying high level users a proportion of the site’s income
  + Once viewer base becomes sufficiently high, start requiring commercial items to be paid for by a subscription
    - The subscription cost should start very small – much smaller than the value of the advertising
    - As popularity of the site grows, price should also grow incrementally
    - Subscription price should also be proportional to the worth of the item
    - Perhaps every time the item worth increases by a factor of 10, the price should double, or something like that
    - The price increase should be incremental, rather than continuous
    - The price might start at $15 per month for items of worth <100
    - Once the item reaches worth of 1000, subscription price increases to $30
    - When item worth reaches 10,000, subscription price increases to $60 etc.
  + Once I’ve worked out legal issues and the site is well known, set up the real money investment system
    - Users get taxed a largish percentage (maybe 10%) when they first invest and all of this goes to the item beneficiary
    - If the item beneficiary has not been determined, the money is associated with the item and will be ready to collect as soon as the deserving beneficiary is determined
    - Another option is to only take out that tax if there is an existing beneficiary, but I don’t like the asymmetry of this (that some items would be cheaper to invest in than others)
    - When a user sells an investment, a small tax is taken out by ZuthaNova (maybe 5%)
    - Actually it might be better to take out both taxes when the investment is sold, just because the incentives are weighted better then
    - So there would be a net 15% tax when selling an item, 10% of which goes to the item beneficiary
    - That 10% is effectively put in the ZuthaNova bank until a beneficiary is found, so there isn’t really any reason to literally give all 15% to ZuthaNova if there is no beneficiary
    - If a beneficiary is never found for an item, the money sits in an account accruing interest or just remains a debt to an unknown purpose and might be used in the mean time for ZuthaNova expenses or other projects.
  + Set up “Bet” items which allow one-way investments
    - Bet items will expire when the true result of the Bet has been determined
    - Bets could be on anything bets can be normally placed on like sporting events
    - Bets could also be set up for things like the outcome of future events (like election results)
    - When the outcome of the Bet is determined, the investors in the winning item get all their money back, plus 95% of all the money in all the losing items
    - 5% of the combined value of the loosing items is taxed by ZuthaNova
  + Set up the affiliate system so that users receive a proportion of ZuthaNova’s affiliate remuneration as an investment in the item (in the real money pool)
    - They can sell this investment immediately if they wish, and get taxed some more on it, or they can leave the investment in the item and hope it increases in value from others investing
  + (possibly) allow companies to choose between a subscription and the affiliate system in order to permit their item to show its worth
    - For this to work well, it would probably be necessary to use a standard affiliate system
    - For a start, it would become unwieldy to deal with many different kinds of affiliate trackers
    - I could choose one good affiliate tracker and require companies to use that one for my site, or possibly set up my own custom affiliate tracker
    - The tracker would need to be able to determine the user that made a purchase as well as the Zuthanova item corresponding to the specific product purchased
* The main discerning feature of what ZuthaNova offers is that the content is all excellent
  + I should not be aiming to organise all the world’s information – that is the role of Google and others
  + I do not want, therefore, to have categories swamped by spam
  + A new user can only gain zuth initially by getting employed, and only that by creating a good item
  + I could potentially allow only comment items to be produced by level 0 users, but that would make it unduly difficult to create something of value that a higher level user might find
  + In order to prevent spammers from swamping categories with crap items, I could cause items with zero worth to become hidden after a period of time
  + Zero worth items would need to last a long enough time to reasonably expect an investor to find it, though
  + It should be possible to show hidden items on demand, or by default in a user’s settings
  + When creating an item, hidden items should come up in the list of possible existing candidates
  + If someone is seeking to create an item and it already exists as a hidden item, the hidden item should be unhidden and given a 2nd chance to be invested in
  + Some items that are unlikely to be worthy of investment, like comments, should not ever be hidden
  + Spam comments and other comments that never get invested in would just sit at the bottom of an item’s pool of comments
  + I might add a single click “This is Spam” button to comment items to allow a degree of clean up
  + Being able to mark comments as spam should require a minimum level to prevent malicious vandalism or bullying

# Fri 9-Sep-10

* Create a feature to show the worth of items as it would be only investments from a given set of users was counted
  + This would allow searching for items based on the preferences of a desired group
  + E.g. feature that implements this tool: “People who like this movie also like…”
  + It should be possible to go into a site-wide mode which calculates item worth only with investments by the specified group of users
  + The amount of calculations to sort large sets of items (like all movies) in this mode could potentially be huge
    - One optimization technique would be to start with only the set of items that has an investment in it by one of the specified users
    - This part would be fast with a graph database
    - If the group of users was huge, and the set of items they had invested in was huge, it could still take a long time to calculate the worth of every one

# Mon 13-Sep-10

* As much information as possible should be publically available by default
  + The amount of money earned by every item benefactor should be publically available
  + This would allow people to find out how much other creators of content are earning and be enticed by it
  + For example, if a popular song writer is earning a lot of money through ZuthaNova, it will be possible for others to see exactly how much that song writer is earning. The media will be able to report on it, and people will be able to exclaim about it with social media like twitter and Facebook
  + This will be good advertising for the site, because it may encourage more content creators to be willing to make their content free and earn their income through ZuthaNova
  + In order to further encourage this, I think even media content that is not free should require a subscription for the item value to show
    - This probably follows from the existing rule anyway (that commercial products require a subscription), but I hadn’t thought about it applying to songs and movies
  + Of course, there would be a potentially long buffer period (perhaps even arbitrary), before the item would actually have its value hidden
    - Basically, the action of hiding the item’s value would be a threat (I’m sure there would be an appropriate euphemism in business terminology), to encourage a successful content owner to pay for the advertising (and possibly the income) they are receiving on ZuthaNova
  + Actually I could even make it that only free content can earn money through ZuthaNova
    - The only real reason I can see to give further income to commercial products is to entice them to advertise on ZuthaNova
    - If commercial product owners were paid for income from their items, it would reduce their incentive to make their product free
    - The main paradigm I want to promote with my site is that free content can make money
    - Therefore I want as much incentive as possible for people to make their content free
    - If making their content free will allow them to make money through ZuthaNova, then that is an incentive
    - They shouldn’t need to see their own income from ZuthaNova to see how effective the system is, because they can look at any other item’s income
    - Even better: the income accrues for every item that is invested in (with real money), but is unavailable unless and until the item becomes free

# Wed 15-Sep-10

* The ability to filter items by more than one criteria should be built into the item browsing system
  + I had previously envisioned that finding an item of interest would involve a linear path through a hierarchy
  + There are inherently multiple ways to categorise many concepts, though
  + If in a category representing the concept of “statistics”, one may want to narrow the related concepts
    - by type (say “articles”, “web services” or “academic disciplines”)
    - by a more specific category (say “crime statistics” or “wealth distribution”)
    - or by geographic scope (say “world” or “victoria”)
  + It would not be practical to have specific items for every combination of categorizations for any given concept
    - i.e. an item for “web services related to Victorian statistics”
    - or “academic disciplines about crime statistics”
    - or “Australian made black and white films about indigenous Australians”
    - or “physical products relating to sleep ailments”
  + However, I want it to be possible to browse from “statistics” to “web services relating to statistics” and have the new context feel like a new concrete location in the graph
  + I think the solution is that a particular page or “location” be composed of multiple nodes such that the results are items with links to all the filter nodes
  + To find “web services related to Victorian Statistics” one might
    - browse to the item representing the concept “statistics”
    - Ask to filter results by a type and then browse through the type hierarchy to “web service”
    - Ask to browse within related concepts, and then browse through a hierarchy which is filtered to only contain categories which contain “web service” items, to find the item representing the region “Victoria”
  + Each of the 3 filters would be listed at the top of the page
    - It would be possible to remove any filter and then items would only be filtered by association with the remaining filters
    - It should be possible to click on a filter and browse from that item with the other filters still active

# Sun 19-Sep-10

* I want to be completely immune to hacking by making absolutely all information that I store freely available anyway
  + I can use OpenID to authenticate users
  + I don’t need to store anything confidential to accept payments; I just need to associate a merchant account like Paypal with an OpenID like a Google account and then ask for authentication from the merchant account.
  + There would still be the issue of hacking to modify data rather than steal it, of course

# Mon 20-Sep-10

* I want to be able to make ZuthaNova open source, but still have ultimate control of the business
  + Making it open source would allow me to make use of the platform of distributed workers I’m trying to create
  + There could be an issue with security though, because allowing just anyone to edit the code (even if I had permission levels) would make it easier for hackers to modify such sensitive data as connections between users and merchant accounts.
  + It might be possible to use encryption appropriately so that it would be impossible to manually change the merchant account connected with a user without the password of the user and the merchant account, even if one had complete editing access to all information stored on the site.
  + It would still be possible to do a great deal of damage if someone malicious got editing access to the code though.
  + To be on the safe side, I could set up an open source sandbox (or three), where anyone can contribute code improvements and new ideas
* I want to encourage ZuthaNova to be a platform for a new kind of economy
  + For this to work, I would have to step back from a position of absolute control, or too many people would be reluctant to endorse the platform for fear of one person having too much control of it
  + The issue is, there would be a heap of money flowing through the system (most of it owned collectively by users), that would have to be managed by someone
  + It is likely that whatever I do, if I make money with this thing, other people will want to copy the idea and get in on the cash
  + The issue is that fragmentation of the platform would significantly reduce its effectiveness
  + An analogy might be: if there were three equal sized independent Wikipedia like services, all three would be much less useful because there would be overlapping conflicting information and some information available only in some of the services. Either individuals would have to search manually through all three – defeating the purpose of having a one-stop encyclopaedia, or there would have to be some kind of metasearch engine which would have to deal with the messiness of incompatible systems and duplicate data. Then there might be competition for the metasearch engine and we start all over again.
  + The reason Wikipedia has worked, though, is because it got to be the biggest first and it is now really hard to compete because people want to use the service that’s ubiquitous.
  + It could be possible to get ZuthaNova into an equivalent position (as Wikipedia, Facebook, Windows…) such that it’s impossible to compete because everyone wants to use the service that already has lots of users.
  + That doesn’t solve the problem of people resisting monopoly control of the platform
  + And in fact, monopoly control of the platform I’m trying to create would be undesirable (even if the dictator is me ☺) because it would undermine its integrity, contradict the ethos it is trying to promote, and limit the ability of the evolutionary algorithm to function to make the platform better.
  + Even if it were possible to somehow give away the platform to the public, I would be reluctant to because I want to be able to use it as a resource to achieve lots of other philanthropic ambitions, like improving education in the world, financing political reform towards openness and public collaboration, investing in great projects etc.
  + In a certain sense my reluctance to give up control of the ZuthaNova platform has a selfish motive: the desire to be able to give with a lot of resources behind me so I can feel good about it and be rewarded with my own and others’ approval as well as the joy of seeing people’s lives improve and knowing I was responsible.
  + I wonder whether if I maintained the right image from the start, if it could be possible for the world to accept ZuthaNova as a massive scale new economic platform, even if it is controlled by one man. I doubt it.
* I think it could be realistic to run ZuthaNova without employing any fulltime employees.
  + In fact I like the idea of promoting an ethos of outsourcing – only to the best services in a field of course
  + I can outsource hosting, perhaps to multiple different web hosts around the world.
    - Each node would have a HyperGraphDB instance in it, and would be connected to the others using a P2P distributed system
    - There isn’t any reason I can think of why using different hosts in different parts of the world would interfere with this distributed system
  + I have already worked out how to outsource a lot of labour through the site itself
    - Tasks like resolving disputes, cleaning up data, and providing support to other users can easily be performed by the hierarchy of “employed” users
    - There would be more sensitive tasks, though, like connecting items to their external owners (or “beneficiaries” I have called them), so that money earned by the item can be paid to the deserving party.
    - For such sensitive tasks, it could be practical to set up legal agreements with higher level users that I trust, and allow them to perform such operations. With a legal framework backing it up, it should be possible to avoid personal liability and also strongly discourage any of these more formally employed users from doing anything dodgy like connecting a popular movie item to an account they own and stealing its income. If such behaviour is unambiguously illegal, it shouldn’t be any different from white-collar crime in a bank or something similar.
  + There would eventually be matters like government relations and relations with other companies to deal with. It might be possible to manage such matters through the internet via the swarm of part-time employees, but there would be a lot of challenges.
  + I’m sure there are a heap of aspects to running a business that I’m not aware of which would make what I’m imagining extremely difficult, if not impossible, but it’s worth thinking about the possibilities.
* Actually there’s no reason why I shouldn’t employ full-time workers
  + It could actually work quite well to recruit people through ZuthaNova itself
  + Users could rise up the ranks within the ZuthaNova power hierarchy and eventually get to a high enough level that they might be asked to enter a more formal relationship with the company.
  + This would be a really effective recruitment system because it would be possible to watch potential employees work on the sorts of things they are going to be hired for, before actually committing to hiring them.
  + I like the system of paying high-contributing users a percentage of the site’s income
  + It would be great to somehow extend this system to full-time workers
  + Full-time workers could simply be assigned a high enough rank in ZuthaNova that the percentage of income they receive is enough for a full time wage
  + The prospect of employment could also be a great incentive for people to contribute their best to the site, especially once the company becomes large and well-regarded as I intend it to.

# Wed 22-Sep-10

* I’ve been warming towards the idea of dropping the “Nova” from the name of my website
  + The domain name I have purchased is zutha.com
  + “Nova” seems somehow corny and irrelevant
  + Zutha was intended when I first invented the name to be compatible with the name of some kind of being
  + I kind of like the idea of using a user called “Zutha” as the main admin user
  + By making zutha.com my primary site, I can still adopt the model Google uses with its site urls, namely using subdomains of the main domain name for extra services
    - i.e. mail.google.com
    - I could create services later and call them <service>.zutha.com
* JSF doesn’t really support sending search queries from a form by HTTP GET
  + This at first seemed to me a major problem because I hate websites that don’t allow you to bookmark pages and I don’t want Zutha.com to be like that
  + I’ve realised, though, that it could be a neat paradigm to make search result pages non-boomarkable, and thus force people to use and think in terms of concrete relationships
  + The whole process of finding an item of interest should be oriented around triangulating its position in the graph through its relationships with other items.
  + Using POST requests for search results initially would also allow me to smoothly and neatly move to ajax search results eventually (or immediately if it is easy enough) while retaining the POST queries for non-javascript users.
  + Using POST search requests would probably also allow more complex search queries
* It should be possible to create a “Search” item from a search results page and thus generate a permalink
* What should appear on an item’s page when no parameters are added?
  + One possibility is that the most appropriate kind of subitem is determined by the item type and those subitems are shown beneath the main item
    - There could be situations where the appropriate kind of subitem is ambiguous
    - I don’t like the inconsistencies that this approach is likely to create
    - It is possible that the most appropriate kind of subitem might change
      * E.g. from “comments” to “instances” if an item becomes a Type
      * This would mean that the nature of the plain item page would change and I find this prospect ugly
  + Another possibility is to only list the options for the kinds of subitems/relationships this item has
    - Examples
      * Epoch: comments, instances
      * Toyota (company): comments, products, external ids
      * J.K. Rowling: comments, works written, external-ids
      * Building: comments, subtypes, instances
    - When a particular relationship type link is selected, a new page is opened which specifically determines that items with that particular relationship with the focus item should be listed
      * Examples
        + Building->subtypes: “zutha.com/browse?type=<type>&supertype=<Building>
        + J.K. Rowling -> Comments: “zutha.com/browse?type=<comment>&about=<J.K. Rowling>
      * I’ve realised that I want relationships like “supertype” and “type” to be items as well which may mean they have to be specified in the GET request by their ids
        + This means things will really start to look ugly
        + Urls to show the comments about an item would look something like: “zutha.com/browse?TH3=8EFW&43D=U89CVW
        + This is not what I want
        + This would be terrible for SEO
      * It is really important that I hide the internals of the data structure to normal users
        + It should not be too hard to define human-readable names for commonly used relationship types and items

Relationships like “type=”, “supertype=”, “about=”, “author=” should all be specified like that in the url, rather than by their id numbers

* + - * + The comments about an item should intuitively be a subpage of the relevant item
      * New examples
        + Building->subtypes: “zutha.com/view/05GH72/Building/subtypes
        + J.K. Rowling->comments: “zutha.com/view/07H34A/J-K-Rowling/comments
        + Microsoft->products: “zutha.com/view/0MR5G/Microsoft/products
      * The subpages like “subtypes”, “comments”, “products” will be defined in item type definitions
        + The Company type (of which Microsoft is an instance) will define a relationship called “products” which is used to link to items of type Product which are “made by” this Company
    - This system is starting to look like it will extend to what Freebase does which is list all the roles an item has and all the relationships given by that role (and Freebase even lists the first few specific items for each relationship type)
* Ids should all start with 0 to distinguish them from human-readable item indicators which will be used in urls and probably elsewhere
  + E.g. 05D3E, 0G112J, 0KR44Q
  + This way if an id starts with something else it should be treated as a human-readable id
  + In the case of “type=comment” in a url, both “type” and “comment” will be found by looking up a different type of id than the universal id
* URL structure
  + Plain item page: “zutha.com/view/05G34/Bronze-Age”
    - This should show the full content of the “Bronze Age” item as the main focus of the page
    - It should also show the list of relationship types this item participates in
      * E.g. “comments”, “events in this period”, “things existing in this period”, “works about this topic”, “events” (internal site events like Edit or Investment)
    - The main item page could also have the interface for investing in an item
  + A particular subitem page of an item: “zutha.com/view/05G34/Bronze-Age/comments”
    - This should show the snippet view of the main item, followed by the list of items that apply to the specified relationship (comments)
  + Item subitems page filtered by relationships: “zutha.com/view/04FG/Time-Period/instances?type=<king>&rel=<Troy>”
    - This should show a filtered version of the set of instance subitems of the Time-Periods item that contain (recursively) items of type “King” that are related to the item “Troy”
    - Instances of Time-Period might include “Stone-Age”, “Bronze-Age”, “Iron-Age”
    - In this case only “Bronze-Age” should be shown because it is likely the only Time-Period that is an ancestor item of items that are both of type: King and related to Troy
    - Only links that are considered parent-child links (like item->comment, type->subtype) should be followed to determine if Bronze-Age is an ancestor of one of the set up items being filtered by.
  + Item edit page: “zutha.com/edit/01FG8/Pretty-Faces”
    - If someone goes to this page who doesn’t have permission to edit, they should be shown an error message
  + Item Investment management page: “zutha.com/invest/01FG8/Pretty-Faces”
    - This should allow investing and selling investments in both the positive and negative pots with both real and virtual currency
    - It should also have a link to a page showing the history of investments in the current item
    - This page may not be necessary – the investing interface could simply go on the item’s main page
  + Login page: “zutha.com/login”
  + Registration page: “zutha.com/register”
* Example Graph Traversals
  + Find Kings of Troy during the Bronze Age (this was an example given by Damian last night)
    - Start at ZuthaHome
    - Navigate through the graph to “Bronze Age”, perhaps with path: Abstraction/subtypes -> Measure/subtypes -> Time Period/subtypes -> Archaeological Time Period/instances -> Bronze Age
    - Click the link for the relationship type: “things existing in this period”
    - Now at “zutha.com/view/0J4RT/Bronze-Age/things-existing-in-this-period
    - Click “Add to filter”
    - Now at “zutha.com/view/00/ZuthaTop?existed-during=Bronze-Age”
      * The item that should be put in focus here is the most distant ancestor to an item that matches the filter, which has more than 1 subitem which is also an ancestor of an item matching the filter
      * Each subitem (and there should be at least 2) should follow almost the same rule: among the descendants of the node now in focus, each subitem should either be an item that matches the filter or be the most distant ancestor of an item which matches the filter, which has more than 1 subitem which also follows this rule.
    - Navigate through the graph to King. Example paths:
      * Physical Entity/subtypes -> Living Thing/subtypes -> Organism/subtypes -> Person/roles -> Ruler/subtypes -> monarch/subtypes -> King
      * Physics Entity/subtypes -> Causal Agent/subtypes -> Person/roles -> Representative/subtypes -> Head of State/subtypes -> King
    - It might be possible that some of the links in the above paths will be missing because they only have one subitem that is in the filtered graph
      * E.g. “Representative” might be missing from the path such that Person/roles contains items: Head of State, Warrior (and perhaps others), but not Representative because it would have only had the one subitem in the filtered graph (namely “Head of State”).
    - Go to the King/instances page
    - Now at “zutha.com/view/09KM3/King/instances
    - This should show a list of all Kings that existed during the Bronze Age
    - Add King/instances to the filter
    - Now at “zutha.com/view/00/ZuthaTop?existed-during=Bronze-Age&type=King”
      * Could possibly be at a lower order item as per the rules above
    - Search for “Troy”
      * Search function uses ajax to render results on the fly or POST if no javascript
      * Only items that match the filter will appear
      * Not sure yet whether ancestors of matching items should appear
      * The purpose of showing ancestors of matching items in the browsing process was to facilitate “browsing” through a hierarchy
      * With Search it would probably make more sense to simply have all results match the filter (in this case “existed during Bronze Age” and “is of type King”)
      * Select the item representing the city of Troy
    - Now at “zutha.com/view/07H4FE/Troy?existed-during=Bronze-Age&type=King”
    - Likely the only relationship-types available will be perhaps “ruled-by” and “people-born-here”
    - Select the “ruled-by” relationship type and the list of results should be the list we were looking for
  + Find Web Services that synchronize contacts between Google Contacts and Windows Mobile and support category sync
    - Search for Web Service and open its main item page
    - Select the “instances” relationship type
    - Add this to filter
    - Search for “Contact Synchronization” and select the item corresponding to the concept of Contact Synchronization
      * This should be a subtype of the concept “Synchronization”
      * I’ve just realised that search results should contain ancestors of items that match the filter because otherwise we couldn’t do this; we couldn’t search for other concepts which are associated with our target concept which aren’t already contained in the existing filter.
      * In fact search results should even return all ancestors of items that match the filter, whereas when browsing a filtered graph some redundant items can be removed as described above
    - Add this to filter
    - Now at “zutha.com/view/00/ZuthaTop?type=Web-Service&related-to=Contact-Synchronization”
      * Related-to means related by any type of relationship
      * Related-to was used because no specific relationship-type was chosen for “Contact Synchronization” when it was added to the filter
    - Find the Google-Contacts item
    - Possibly go to a specific relationship-type; perhaps it would be “supported-by” or something in this case
    - Add to the filter
    - Do the same with Windows Mobile
    - The ability to filter by items that support category sync would probably require a lot of specialised properties to be created for this knowledge domain
    - But perhaps it would suffice to use the concept of “Supports” and “supported-by” as we have for Google-Contacts and Windows-Mobile
    - Perhaps all it would take is to find the item representing the concept of “Category” and go to its “supported-by” relationship-type and add this to the filter
      * My only concern with this is that the “supported-by” relationship type would be confusing in the Category item to people not thinking about Category as a concept which is supported by a particular service
      * Even worse, if a specialized domain like Contact Synchronization is allowed to add a relationship-type to a concept as ubiquitous as Category, then how many other such specialised relationships might be formed? Allowing this to get out of hand could quickly overwhelm a lot of items with a mess of inexplicable (to most people) relationship-types.
      * Perhaps there could be a way to define certain relationship-types as “obscure” or something and have them be hidden unless there is an active filter which is specific enough to the knowledge domain that the obscure relationship-type applies to, to make it visible.
      * It should also be possible of course, to ask to view the set of obscure relationship-types explicitly.
      * In fact, this might be all that is necessary: a division between primary and obscure relationship-types such that you have to really be looking to find a particular obscure relationship-type
    - The results set should now contain Web Services in the genre of Contact Synchronization that “support” Google Contacts, Windows Mobile, and Categories
* There should be a setting specifying whether, on visiting a plain item page that only has subitems connected by one relationship-type, the user should be redirected automatically to the appropriate subpage.
  + This is particularly relevant for when the graph is filtered such that many of the normal list of relationship types will be missing
* Not requiring search queries to be GET requests will add a lot of flexibility
  + I could create an advanced search interface which would allow javascript adding of fields and other complex tools and not have to worry about how to render this as a flat GET string
  + If I used POST requests I could use XML for really complex queries
* One of the benefits of the old linear system of browsing through the hierarchy was that one could go up one level from an item and find other items that fit in the same specific category as it
  + E.g. If you found NuevaSync, you could look through its list of parent items and find the item representing Contact Synchronization Web Services
  + It should be possible with the new system to have all that power and more
  + The difference now is that there will be no item called “Contact Synchronization Web Services”, but there may be an item called “Contact Synchronization” and one called “Web Service”
  + NuevaSync would have type “Web Service” and genre “Contact Synchronization”
  + The goal is to have a very easy way (preferably one or two clicks) to go to a page which shows Contact Synchronization Web Services
  + Perhaps next to instances of upward relationships like “type” and “genre” there could be a checkbox
    - You could check boxes next to the type “Web Service” and the genre “Contact Synchronization” and click a button to perform that filter
    - The form would send its information by POST
    - You would be redirected to a page like “zutha.com/view/00/ZuthaTop?type=Web-Service&genre=Contact-Synchronization”
    - Or “zutha.com/view/0JH4E/Web-Service/instances?genre=Contact-Synchronization”
    - Or “zutha.com/view/04FV3/Contact-Synchronization?type=Web-Service”
    - It would feel inelegant to use either of the latter 2 options, especially since it would be unclear which should be chosen
    - The reason the latter 2 options are listed is because it would likely be desirable to see the list of all items that match the filter immediately instead of having to wade through the hierarchy again
    - The solution appears to be that the first option should not necessarily show a hierarchy of ancestors of the items matching the filter
* When a user asks to add the current view to the filter, they should be sent to a page which doesn’t correspond to an item in the graph
  + There is something inelegant about a location like “zutha.com/view/00/ZuthaTop?type=Web-Service&genre=Contact-Synchronization”
    - I have been working with the idea that this page would show a set of the highest level ancestors of the items that match the filter (+ some other rules discussed above)
    - This is not consistent with the behaviour of an item main page in other contexts
    - Normally an item main page shows the relationship-types of its subordinate relationships (like “comment”, “instances”)
    - The page that a user is sent to after adding something to the filter should be targeted towards browsing the result set, searching the result set, or searching for another item to add to the filter. It should not be targeted towards browsing for the next item to add to the filter.
  + The url could look like “zutha.com/view?type=Web-Service&genre=Contact-Synchronization”
    - This could simply show the set of items matching the filter without any focal item at the top as there would be on an item subpage like “King/instances”
    - The page “zutha.com/view” would thus show the set of all items in the graph
      * The items would be sorted (by default by worth) and only a limited set would be shown on one page of course
  + From this page, it should be possible to either search the set of items in the page, or search through all these items and their ancestors
  + It should also be possible to “group by” a particular attribute of the items in the result set
  + This results page could have options on it to
    - Group By
      * With javascript enabled this would open an in-page interface to find an attribute to group by
      * The attribute can be a multivalued one
      * I’m not sure yet whether it will be necessary for the attribute to apply to all items in the result set
      * I think it would make sense for every attribute that applies to at least one item in the current result set, to be available
      * If an attribute is selected which some items don’t even have, then those items just won’t show up under any of the attribute’s values
      * Grouping could be done on something like “type” or “author” and perhaps even subordinate relationships like “instance”
      * If grouping was done on “type” the set of groups would be constructed by finding the union of the “type” relationships of every item in the result set
      * The generated set of groups could be rendered the same way any other list of items is rendered
        + This set of groups (each of which is represented by an item) is a subset of the ancestors of the filtered items
        + It’s just another way of collecting together a set of items of which one or more can be chosen to add to the filter
        + The behaviour when clicking on an item in the group-by set would be same as clicking on an item while Browsing the Filtered Graph or in search results of the Ancestors of Filtered Items set
      * The url for a group-by page might look like: “zutha.com/search?group-by=type&author=J-K-Rowling
    - Browse the Filtered Graph
      * This would go to the TOP item with the filter data included in GET parameters
      * The TOP item is a category item just like any other which will show a set of relationship-types of its associated items
      * The relationship-type: subcategory is most likely going to the only one that yields items in the set of ancestors of the filtered items
      * I think I’ll abandon the idea of hiding redundant nodes in the graph (i.e. ones which will only have on child) because its only purpose would be to reduce the tediousness of browsing the graph and Search is the better way to avoid this tedium. Also, it is likely to be more confusing than anything else, because items can have a different set of subitems in different contexts.
    - Search the Descendants of Filtered Items set
      * Go to “Browse the Filtered Graph”
      * Use the search bar
      * Search results will only be for descendants of the current item
      * Actually, given how frequently I foresee it would be desirable to search the whole filtered graph (including ancestors) immediately after adding something to the filter and reaching the Search page, I should add a button next the search bar to “Search Ancestors”
        + For non-javascript users there would be 2 buttons: one normal search and one “Search Ancestors”
        + For javascript users… actually perhaps javascript users should have the two buttons as well, just for clarity, and the instant search defaults to normal filtering of results
        + Perhaps after “Search Ancestors” is clicked for the first time, instant search could be switched to apply to this search as well
    - Search within results
      * This is just done using the search bar. Results should be filtered instantly using ajax.
  + I think this page that doesn’t have a specific item in focus should be called a search page
    - Instead of “zutha.com/view?type=Web-Service&genre=Contact-Synchronization”
    - url should look like “zutha.com/search?type=Web-Service&genre=Contact-Synchronization”
* It will be pretty easy to have multiple filters on the same relationship type
  + E.g. “zutha.com/search?type=author&type=singer”

# Thu 23-Sep-10

* I’m still thinking too much in terms of a hierarchy and I think that is holding me back from a better paradigm
  + The first issue that came to mind is that there will be ambiguities in which relationships should be considered parent-child relationships
  + Thus there is going to be ambiguities in defining which items should fall within the set of ancestors of items matching the filter
  + Another issue is that constructing and searching the filtered graph (defined in this way as ancestors of matching items) is going to be extremely computationally intensive
  + I think I need to rely more on the Group-By feature and search
    - Group-By will be incredibly powerful since it will be a really effective way of narrowing down a set of results by a specific criteria
    - Group-By will actually be sufficient for finding any item in the graph
    - Search will be used in conjunction with Group by
      * It should be possible to search the set of relationship-types to group on as well as use instant search within a set of groups and within the result-set
    - Search should also have a button to Search Everything (this should be activated by pressing ENTER, whereas instant search only filters the current results)
    - By using Search Everything while there is a filter active it will be possible to find other outside concepts to add to the filter
    - It will also be important to be able to create filters that don’t return any results (which wouldn’t have been possible when searching the filtered graph)
      * This is because I want the Create Item button to appear on search result pages and be able to take in initial parameters using the specifications of the current filter
      * It may be that someone wants to create an item which has a set of relationships which is at that moment unique in the graph
      * If only the filtered graph could be search, it would not be possible to construct the desired filter
  + Search Everything will be a kind of alternative to Group-By
    - Their functionality will overlap
    - Some examples to illustrate the difference
      * Find Kings who ruled Troy (using Search Everything)
        + Search Everything for the King item
        + Go to Instances page
        + Add to filter
        + Search Everything for Troy
        + Go to “Ruled-By” page
        + Add to filter
      * Find Kings who ruled Troy (using Group By)
        + Search Everything for the King item
        + Go to Instances page
        + Add to filter
        + Now at “zutha.com/search?type=King”
        + Group By: find relationship-type “kingdoms-ruled”
        + Use Instant Search to find Troy
        + The link to Troy should send user to “zutha.com/search?type=King&kingdoms-ruled=Troy”

There should be a more obscure link that takes the user to the Troy main page

* The Edit page should not have a distinct URL
  + This will also fit in with the eventual paradigm of using javascript to enable and disable editing of an item
  + The Create Item button should perhaps only exist on the Search page
  + It will take its initial parameters from the current filters (not from any search strings)
  + The URL will not change when opening the Create Item dialogue
  + Once the item is created the user will be directed to its newly created main page
* The current filter data should be carried in the server side application state
  + There may be some situations where the current filter should not be carried in the URL, but should still be shown at the top of the page and still be active once the application returns to the right context
* There should be a way to find all relationship types between two sets
  + When I was writing the above example graph traversal “find Kings who ruled Troy”, I realised that it might be difficult in some cases for users to work out what wording to use to find the desired relationship-type to Group-By or to add the filter
  + Imagine someone has added King/instances to the filter
    - They now want to find Kings that ruled Troy
    - They find Troy using Search Everything
    - They aren’t sure now which relationship-type to use to filter Troy with
    - What would be ideal in this situation is if only relationship-types that lead to items matching the filter should be shown
    - Furthermore, when a relationship-type is selected, the result-set should be similarly filtered by the active filter
    - This seems a natural property of what having an active filter should do
    - As long as Search Everything is available, there is no reason why the filter should be active in every other way – indeed that is the behaviour that would seem most intuitive
  + This solution doesn’t make it easier to select the right relationship-type in Group-By
    - This isn’t really necessary
    - If the searcher is looking for a relationship between King and Troy they should use the above method of adding King to the filter, then finding Troy separately (or vice versa). This will show the relationships between the two in the Troy main page
* It could be useful to be able to disable a filter temporarily
  + I’m using “filter” here to refer to a particular filter within a set of filters (such as type=King)
  + If you were on the Troy main page with the filter type=King active, you might want to deactivate that filter so that you can see all relationship-types from Troy
  + A good example of when you might want to do this is if there are currently no Kings who ruled Troy stored in the graph but you want to create one
  + You would have to first disable the filter type=King so that the relationship-type “ruled-by” appears for Troy
  + You would go to the Troy/ruled-by page and then re-activate the filter
  + There would be no results in the page now
  + Then you select Create Item and the properties type=King and kingdoms-ruled=Troy would be already added to the form
  + It is possible that the King you want to add already exists, but has not been given the attribute kingdoms-ruled=Troy
  + When you start typing a name for the new item, ajax should grab a set of search results for that name
    - These could perhaps have their relevance rank affected by the existing properties specified in the Create Item form
    - This would mean that an item of type King that matches the name being typed would be ranked very highly
* Item Subpages should have Create Item buttons as well
* I should add lots of configurable hotkeys to control the site and generally work with a paradigm of enable keyboard use as much as possible
  + It should be possible to
    - press a hotkey to go to the search bar
    - type a search string and press enter
    - use tab or arrow keys to select a result
    - use tab or arrow keys to select a subpage (a relationship-type) and press enter to go to it
      * It might also be that the search bar searches relationship-types on an item main page
    - use a hotkey to add to the filter
    - use arrow keys, tab, page down etc. to browse the result-set if desired
    - OR: use a hotkey to go to the search bar
    - Type a new search query and press enter to Search Everything
    - Etc…
* When a user is first created, a tutorial mode should be enabled
  + Tutorial mode will show help bubbles automatically when new functionality is encountered
  + For example when a user first clicks in the search bar, a help bubble should appear explaining that if they just type, the current result-set will be filtered by their query, but if they press enter, the entire graph will be searched.
* In any page showing a set of items, the appearance and layout of the actual set of items should be the same
  + Each item has a small thumbnail
    - The thumbnail default will specified by the item’s type (this might require items to have a primary type if multiple inheritance is allowed)
    - E.g. if the item representing the book “The Lord of the Rings” has not had a thumbnail uploaded for it, then a generic image of a book will be used
  + Above the thumbnail is a link to the item owner (if there is one)
    - Items like Comments and Documents whose value is fully contained in the content stored within the item, the owner is the user who created the item
    - All other items have no owner by default
    - An owner can be added when one is identified
    - For a product, the owner will be the company that produces the product
    - For a book, the owner will be the author that wrote the book
    - The owner can take the place of the identity I was calling the “beneficiary”
    - Both zuth and real money invested into an item has a proportion given to the item owner
    - Before an item can be assigned as the owner of another item, it must be turned into a user and authenticated with the real entity that it represents
    - E.g. In order for Windows 7 to be assigned the owner “Microsoft”, the item Microsoft must first be turned into a user which is controlled by the actual real-life company Microsoft.
  + The title is a link
    - Usually this goes to the item main page
    - In the case of the result-set of a Group-By, the link will be to a search page with that item added to the filter using the relationship-type that was being grouped on
  + Just below the title is a list of the types this item has
    - If there are more than a certain threshold, only the first few are shown with a link to “show more”
  + Top right is the timestamp
  + Along the bottom of each item is two bars
    - One green bar representing positive investments
    - One red bar representing negative investments
    - Each bar should show the actual worth value on top of it
    - The scale should be a percentage of the highest valued item in the result set
      * The anchor value which will represent 100% on the percentage scale will be the higher of the two max worths (max positive and max negative)
    - The kind of worth to use (real money or Zuth), will be determined by the Order By choice
      * Then what happens if results are ordered by relevance?
      * Perhaps there should be a separate drop down to select the kind of worth to use
      * Actually I think it would be better if both forms of worth were visible at once so they can be compared
  + Along the bottom there should be 4 bars
    - 2 for each kind of worth
    - A green bar for positive worth in zuth
    - A bright red bar for negative worth in zuth
    - A gold bar for positive worth in real money
    - A bronze bar for negative worth in real money
  + Between the title and the worth bars and to the right of the thumbnail is a short description of the item
* Should I put a Reply button on comment items?
  + A more general approach would be to add an “add comment” link under every item in every result set page
  + This would reduce the number of clicks required to add a comment from 3 to 1
    - Without the “add comment” link you would have to go to the item main page, then the comments subpage, then Create Item
  + This shortcut could allow a simpler item creation interface to be used for comments that doesn’t have all the bells and whistles of the normal item creation form
  + I think I would rather users be forced to see the existing list of comments before creating a new one
* There will be many situations where it will be annoying to have to first go to an item main page and then click a link to the desired subpage
  + Examples
    - Browsing through settings
      * Start at the Settings item
      * We want to see all the subcategories of Settings so we go to the subcategories page
      * We click on a particular subcategory, say User Interface Settings
      * We now have to select either the instances subpage or the subcategories subpage
  + It would be better if certain subpages are opened by default
    - The Settings item should open the subcategories page by default
    - The default page should probably be specified explicitly for each item
    - Perhaps it could also work to be able to specify a default subpage for an item type so that all items of that type default to having that subpage as their default
    - The default subpage for particular items should also be able to be defined explicitly, overriding the default inherited from the item type
  + It should be easier to move between subpages
    - Currently it is necessary to go back to the main item page before you can branch out to a different subpage
    - If you go to the Settings item and it defaults to the subcategories page, but you want to go to the instances page so you can search all Settings instances, you should be able to get there in one click.
    - My model a while back was to have a horizontal navigation panel under the blurb for an item
      * This would contain links to Main Page, Comments, Events, Investments, etc.
      * It would be impractical to extend this navigation panel to all relationship-types because items can have arbitrarily many relationship-types
      * There are some primary relationship-types, however, such as the ones just listed
      * I could add a page to this navigation panel for “Relationships Types”
        + This page would show all relationship types the item participates in
        + It would not be necessary to navigate directly from one obscure subpage to another – it would be sufficient to be able to go back to the “Relationship Types” page and go to the other subpage from there.
    - I really want this sideways navigation (one subpage to another subpage) panel to be more extendible
      * Some items will have a relationship-types: subcategories, instances, subtypes, comments, investments, events
      * Some items might only have: comments, investments, events
      * A horizontal navigation bar doesn’t lend itself to variable numbers of destinations
      * A vertical navigation list would be more appropriate
        + The problem is I don’t know where such a vertical navigation list would fit into my vision of the page layout
        + Perhaps I could make it a dropdown menu
* With JSF, when a browser visits a web application for the very first time, all the links in the resulting page will have jsessionid appended to them
  + I don’t like this behaviour
  + I’ve searched for a way to prevent this, but only found people complaining about the same thing with no solutions
  + One way to avoid it might be to (for virgin requests like this) put a javascript function in the page which will immediately redirect the browser to the same page without jsessionid parameter in the url
  + By this time the cookie will have been set and the JSF framework will receive the cookie on the next request and not need to rewrite the links to contain jsessionid
* There are security risks associated with using URL rewriting for session management
  + I will be outsourcing authentication for financial transactions so this won’t be placed in jeopardy
  + The main issue will be hackers getting access to accounts with high level editing privileges
  + I could require SSL for editing actions that require a high level
  + For low and middle level edits, it might also make sense to authenticate the user with each edit request
    - I’m not sure how this could be done without it being a pain in the arse
    - I have to research how OpenID works
  + Many security vulnerabilities with URL rewriting and Session Management in general can be avoided by making every session identifier IP address specific
  + If I really want high security, I Shouldn’t use “Remember Me” functionality
  + Users would then have to log in again each time they visit the site
  + This means sessions would expire in a short time frame (say 5-20 mins)
* The measures that would need to be taken to make my application truly secure would be really annoying to the average user
  + Remember Me functionality is a huge convenience
  + I only really need high security on high level users
  + I could require these additional security measures only for users that are logged in with high level editing permissions
  + I should make even high level users default to only having low level editing permissions
  + In order to perform high level editing actions, users would have to log in again and they would enter a high-security mode
  + High security mode would involve such things as
    - HTTPS
    - Regular Regeneration of session-id
    - Sessions expire in 5 mins (no “remember me” functionality”
    - No session tracking by URL rewriting
    - Re-authentication before high-level actions

# Fri 24-Sep-10

* It could be possible to use the search results from Google (or another search engine) for the whole web, and use the urls of the results to find items in Zutha
  + This would make it possible to find web pages about a particular topic using Google’s technology, and then find where that web page fits in to the Zutha hierarchy
  + Using Zutha, it would then by easy to find other relevant (and highly rated) information about that very specific topic
* Only a specific set of subpages will be available
  + Subpages will all be arrayed in a horizontal navigation bar on every item page
  + When an obscure relationship type link is selected in an item’s properties page (like “is supported by”), the set of items related by that relationship type is constructed in a search page using appropriate filter parameters
  + Subpages will include Investments, Comments, Subtypes, Subcategories, Instances, Events
* When navigating a hierarchy of types, it would be useful if clicking on a subtype took you straight to that subtype’s subtypes page
  + The referring page should be taken into account when deciding which subpage to show by default
  + The main page (the “Details” page) should be shown on most occasions
  + It would be necessary to perform a redirect to implement a default subpage, so this behaviour shouldn’t be too predominant
  + The only rule necessary might be: if the referring page was a Subtypes subpage, show this item’s subtypes subpage
    - This rule could be easily implemented without requiring redirects, by specifying the appropriate subpage of the target item in the links of the referring page
* Store tooltips for many controls and site features, in items
  + This might involve creating “Tooltip” items, whose sole purpose is to carry the value of certain tooltips in the site
  + Some tooltips will be gathered from the “description” property of non-Tooltip items
  + For example, there should be a tooltip on property names in an item’s Detail page. This tooltip should be pulled from the description property of the Property item which defines the named property
* I think I should abandon the idea of supporting complex content like Articles and Tutorials
  + I think one of the reasons I thought this would be a good idea was so I could have the content of those articles indexed by my site
  + Another reason may have simply been a desire to kind of “own” or have control of more content
  + I’ve realised, though, that I should focus as much as possible on the new value that I am trying to offer
  + There are other services (like Google Docs and blogging services and such) that have rich interfaces for creating rich content
  + I have already decided I want to adopt a model of outsourcing as much functionality as possible
  + I’ve just realised recently that abandoning rich text support in any item is in alignment with that
  + I’ve just started to remember some cases which I think might need more than a simple plaintext Description property
    - I want to write extensive documentation about all aspects of the site
    - I want to write blog-like posts about site-related issues and decisions I make
    - I want to facilitate discussions
  + Perhaps the answer is to deeply integrate one or more outsourced rich-text publishing tools (like Google Docs)
  + The main issue with having to go to a 3rd party site to view a document is that you lose the convenience of the surrounding navigation tools of Zutha.com
  + I could add a “Document” page for items that represent an outsourced document
    - This page would embed the external document the item represents
    - I could create all my documentation in public Google Documents and have them be embedded in associated items in Zutha
* I noticed a problem with my first idea about how to define the scale of the worth bars
  + What should the scale be for an item main (or Detail) page where there is only that one item on the page?
  + I also think it would be confusing for the scales to be changing all the time
  + It would be better if people could begin to get an intuitive sense of how valuable an item is given the length of its worth bars
  + A better way might be to use a logarithmic scale for worth bars and have the 100% point anchored at the highest worth item site-wide
    - If an item y has double the worth bar length of x, y’s worth is x’s worth squared

# Sat 25-Sep-10

* Topic maps seem to inherently be unconstrained in that it any kind of relationship could be created between any set of concepts
  + I could potentially define constraints in my application using properties and relationships on Type items
  + One possibility is to actually allow massive flexibility the way Freebase does, and use the permissions system for maintaining data integrity
  + This would mean that a high level user would be allowed to do nonsensical things to the data structure, but they would be violating their responsibility as a high level user by doing so and wouldn’t retain that high level permission for long
* How to determine which topics to rank highest when there are multiple unrelated concepts that much an input keyword
  + User interests could be taken into account
  + If a user has a known interest in sport, then a search for “squash” should prioritise the sport rather than the plant (and vice versa)
* Example n-ary associations
  + Authorship: work, author\*
  + murder: victim, perpetrator, cause of death
  + Parenthood: father, mother, child\*
  + Ranking: country, sport, rank
  + Subject Result: student, subject, mark
  + Plays For: Player (Person), Team (Team), Season (Year)
  + Vote by committee: decision, “committee member”\*
  + Has flight from to: origin (Airport), Destination (Airport), Time (Weekday and Time)
* Events should be hierarchical
  + An event can have multiple sub-events
  + For example, if a user creates a new item which requires additional items to be created, there should be a parent event for everything that results from the user’s action, and children events for the subsidiary events
  + It should be possible to undo and redo large compound operations
  + A user might import an entire Topic Map into the database
    - All the new item creations and merge operations will occur
    - The whole import will be represented by an overall “Topic Map Import” event
    - Every fragment of the process, such as individual item creations, will be captured by child events
* It is essential that Events capture information in as general a way as possible
  + I want to be able to completely redesign the data structure of the site
  + I should be able to run a set of events through a new implementation that were generated in an earlier implementation
* I’ve decided I should at least start by using Ontopia instead of HyperGraphDB as my backend
  + Ontopia has awesome and incredibly extensive documentation
  + Ontopia has a lot more development activity going on
  + Ontopia is keeping up with the latest standards in Topic Map technologies
  + I should keep my web interface implementation (in JavaServer Faces) separated from my database management logic so that I can change backend technologies later
  + One day when I’m making lots of money and I am much more experienced in this field, I might try to encourage Ontopia and HyperGraphDB to collaborate to produce a really fast, flexible and distributed backend for Topic Maps

# Sun 26-Sep-10

* TMAPI seems to specify item IDs internally
  + I want to be able to uniquely identify Items with my human friendly ID system
  + There will probably be a need for metadata-like topics in the topic map that should not be items
  + I want to be able to merge topics and move both IDs into the new topic so that links to the removed topic don’t break
  + If there was a more durable internal ID than mine for every topic, I could literally move my ID from the removed item to the surviving one so that the surviving item of the merge now has two of my IDs
  + This is similar to the MID (machine id) system that Freebase uses: they have a durable UUID for every conceivable entity in their graph, and MIDs for every entity that is more likely to be specifically referenced. When entities are merged, all the MIDs will point only to the new entity. You have to use a UUID to access the “deleted” object.
  + My human friendly ID can be an unscoped Name of the item it identifies
    - This will allow me to prevent merging on certain kinds of items (like Comments) for which having the same name in the same scope does not indicate equivalence of identity
    - Comments would be primarily identified by their ID
* I’m not sure how I should scope the item name so that it is always visible, but doesn’t indicate the item should be merged with other comments with the same name in the same scope
  + Comment names should at least be scoped by the Item they are about
  + That leaves the question: do they have a different name if they are found outside the scope of their item?
  + It could make sense to put the Item name of the “About” item into the Comment name for the unconstrained scope
  + If you searched for Comment items with “I like” in the title, you would find a lot of comments which would be difficult to make sense of without knowing the context; that is, the item they are about
  + The about item name could be prepended (or appended) to the comment name
  + “Elayne is sexy” would become: “The Wheel of Time::Elayne is sexy”
    - Or maybe “Elayne is sexy [The Wheel of Time]”
  + This doesn’t yet solve the scope issue, because multiple comments about the same item can have the same name
  + I could also scope the name with the Comment owner (the writer of the comment)
  + This would mean putting the comment owner name in the Comment name in the unconstrained scope
  + So in the about item context: “borris: Elayne is sexy”
  + In the “borris” context: “Elayne is sexy [The Wheel of Time]”
  + In other contexts: “borris: Elayne is sexy [The Wheel of Time]”
* There are a lot of cases where item names would have to be automatically generated for the unconstrained context in order to prevent duplicate names
  + There needs to be a consistent way to choose a scope qualifier which will allow the extended name to be unique
  + Adding [<type>] to the end of the name might work
    - Apple [Company]
    - Apple [Fruit]
  + What if an item has multiple types?
    - The first (and primary) type should be used
* Should generated names appear in the url?
  + Zutha.com/view/0B4H6G/borris:-Elayne-is-sexy-[The-Wheel-of-Time]/
  + Or: Zutha.com/view/0B6H6G/Elayne-is-sexy/
  + The latter clearly looks cleaner
  + The name in the url is really just for search engine optimization; the ID in the url is what identifies the item
  + I think adding the scope qualifiers to the name in the url would just muddy the clarity of the concept for search engines, so I’ll leave them out
  + What about in the case of Apple?
    - Here, qualifying whether I mean Apple the company or Apple the fruit would help SEO
    - Zutha.com/view/0N4AQ/Apple-[Company]
* Which url format is best?
  + Zutha.com/view/0B4H6G/borris:-Elayne-is-sexy-[The-Wheel-of-Time]
  + Zutha.com/view/0B4H6G/borris-Elayne-is-sexy-The-Wheel-of-Time
  + Zutha.com/view/0B4H6G/borris\_Elayne\_is\_sexy\_The\_Wheel\_of\_Time
  + Zutha.com/view/0B4H6G/borris:\_Elayne\_is\_sexy\_[The\_Wheel\_of\_Time]
  + Zutha.com/view/0N4AQ/Apple-[Company]
  + Zutha.com/view/0N4AQ/Apple\_[Company]
  + I think replacing spaces with underscores and keeping most other formatting is best
  + What about Question item: “How do I find the x-intercepts for x^2/9+y^2/4=1 ?”
    - Zutha.com/view/07BH8J90/How\_do\_I\_find\_the\_x-intercepts\_for\_x^2\_9+y^2\_4=1\_
    - Zutha.com/view/07BH8J90/How\_do\_I\_find\_the\_x-intercepts\_for\_x^2\_9+y^2\_4=1\_
    - Zutha.com/view/07BH8J90/How\_do\_I\_find\_the\_x-intercepts\_for\_x^2%2F9+y^2%2F4%3D1\_%3F
* Search Everything should always have Ctrl-Enter shortcut
  + Since I will not implement ajax instant search immediately, for a while, users will be using Enter to search within page and Ctrl-Enter to Search Everything
  + Non-javascript users will retain the two Search button method after I implement Ajax instant search
  + Perhaps I should even keep the Search button as well as the Search Everything button even for javascript users once I implement Ajax instant search
* Every item should have a URL name that is generated automatically when the item is created, but can be modified later
  + If the main (unconstrained) item name is modified, the URL name should be regenerated, but can be modified manually again while modifying the main name or later
* What should happen if someone wants to create an item with a name that matches a name in another item in the same scope
  + The creating user should be notified that their name matches another in the same scope
  + They should be shown the other item and asked to decide whether their item should be merged with it
  + If they are creating a new item that just happens to have the same name as another in the same scope, they should be forced to specify a new scope for the name, or change the name (e.g. add a middle name)
  + Example
    - There exists already in the graph, a Person item with a name “John Smith” that is qualified by the Person scope (this would happen by default by using the item’s main type: Person)
    - Someone wants to create a new “Josh Smith” Person item
    - The Person scope will automatically be added to the new item
    - The application will then discover that another item named “John Smith” exists in the Person scope
    - The creator will be asked to either merge their new item with the existing “John Smith”, change the name, or add a qualifying scope to the name
    - So the creator could rename the new item to “John Lawrence Smith”
    - Or he could add a qualifying scope to the name “John Smith” of Architect
    - If he adds a qualifying scope to the name, a new name should be generated automatically for the unconstrained scope, which adds the chosen scope (Architect) into the suffix
  + Actually I’m thinking that it might not be ideal to automatically add any scopes (even the main Type)
    - Some types of items (like Comments), should have rules that automatically generate the name for the unconstrained scope
    - A Person item, though, should not have a scope added by default
    - So the first “John Smith” item would have one name in the unconstrained scope
      * Of course, it would make sense for the initial creator of this item, to give it a scope (like Scientist), because they know it is such a common name
    - When the second “John Smith” item is created, it would have a name collision with the other “John Smith” in the unconstrained scope
    - The creator should add the scope Architect to the name “John Smith”
    - At this point, a new name should be generated of the form: “John Smith [Architect]”
* Why am I using square brackets for scope qualifiers in names?
  + I was thinking maybe some item names might naturally have round brackets in them that are inherently part of the name and don’t represent qualifiers
  + A subtitle of a book might be a good example
  + E.g. “The Inheritance (The World Obama Confronts and the Challenges to American Power)”
  + This bracketed subtitle appears naturally in mentions of this book and does not represent a scope qualifier
  + If a scope qualifier needed to be added to this book (doubtful), it would look like:
    - “The Inheritance (The World Obama Confronts and the Challenges to American Power) [Book]”
  + It is examples like this which make it clear that scope qualifiers should not necessarily be added automatically
* The “Add Item” button should be the portal to creating items and adding associations
  + The “Add Item” button should appear on any page showing a set of items
  + “Add Item” should open a form to enter the name of the item to add
  + The name entered should be used to search for existing items
  + It will be tricky to determine how much search results should be filtered by context
  + Obviously the full context cannot be used, otherwise the item would already exist in the set
  + In fact, items that are already in the set must be excluded from the search results
  + Items that match more of the context should be made more relevant and appear higher
  + These search results should not be ordered in any way by worth
  + Even items that don’t match any of the properties of the current context should be returned, but they should appear at the bottom of the list
  + If the desired item to add to the current context is found, all the necessary relationships are added to it and it will appear in the current set
  + This means that any item can be added to any context and thus have any arbitrary relationships added to it
  + The user doing the action will have to have the required permissions to make all the necessary changes, of course
  + If the user selects “create new item”, the create item form should open with the required information filled in that would put the new item in the current set
    - This information should not be editable
    - This means that, when creating a new item, one must first find at least one appropriate context for it to reside in
  + There should be no “Add Item” button on the blank Search page
    - It will appear once at least one filter has been added
* In Ajax mode, relationships are added to items during creation time by typing the name of the item to relate and an Ajax drop down will show instant search results
  + If the item is not found, there should be a button or link that will open in a new window to allow the non-existent item to be created
  + The context for creating the original item, as well as the name given for the non-existent item, should be automatically filled in in the 2nd create item form
* In no-javascript mode, relationships are still added by giving a name for the item to relate
  + When the form is submitted, associations to other items will be resolved by searching for items with the given name in the appropriate context that have the right type
  + If multiple matches are found, the form should be redisplayed requesting the user to select from among the matches
    - The user can specify that the desired item to link is not among the options and wants to create a new item
    - If the user elects to create a new item, he should be sent to a new tab to do so
    - Again, as much information should be drawn from the context as possible to prefill the new item
  + If no results are found, redisplay the edit form and specify that there were no results
    - There should be a “create new item” link below the error message of the same kind as the one that appears when multiple results are found
  + If one result is found, the user should still be shown a confirm page that shows the details of the item that is going to be linked
    - From the confirm page, they can chose to go back and edit the choice of linked item
* Items that represent addressable resources should have an icon-sized link next to their name that links directly to the external resource
  + Example: Soocial Homepage [[\*](http://www.soocial.com/)]
  + The “Soocial” Item which represents the Company/Product/Service, will have a SubjectIndicator point to the “Soocial Homepage” Item
  + In the Soocial Item Details page, there should be a set of External IDs listed, of which “Soocial Homepage” will be one
    - The main body of the link to the “Soocial Homepage” in this case should go directly to the external resource
    - There should be a smaller icon link to go to the “Soocial Homepage” Item
* I don’t think it will work to turn existing items into users
  + I had considered doing this because it seems appropriate in a situation where a company owns a product
  + In that case, making the company item the owner of the product item is elegant
  + But the owner here should refer to a user, because ownership of an item carries with it editing privileges and the right to earn money and zuth from the item
  + I think it would be simpler, and not unacceptably inelegant to just make a user the owner of products
  + That user would have to be able to authenticate himself as having the right to be the owner of that product
* How to authenticate users as deserving to own an Item
  + Call the company that owns the product in real life and ask them to confirm which user they want to be the owner of the product
  + A company (or any other real life owner of a concept) should be able to, at any time, contact Zutha (by Phone, email or in a comment in the Zutha graph) and complain that the wrong user owns an item representing one of their products
    - If this happens, a responsible high level Zutha user just has to call the company again and ask them to confirm which user should own the product in question
  + I’m sure there will be far more difficult cases to identify than companies
  + An Article owner
    - Contact the publishing website and ask them to provide contact details for the article owner
  + The owner of a YouTube video
    - Send a message to the user that created the video and again just ask them to choose a Zutha user to own the corresponding Zutha item
  + What if the author of something cannot be found, but someone contacts Zutha claiming to be that author
    - Ask the claimant to identify some authoritative body which can independently get us in contact with the author of the work in question
    - This 3rd party body should also be able to verify that the person they are getting us in contact with is indeed the author of the work
    - I think there will be a lot of cases where this won’t be too difficult
    - The claimant can identify a university or the business they work at (if it is reputable enough), and that organisation can in turn get Zutha in contact with the prospective item owner.
    - Some universities or companies might refuse to disclose any information about the person sought, but this can easily be overcome by asking the claimant (whose real life identity is unknown, but who is nonetheless contactable if they have made a claim on an item) to communicate with their chosen verification source and asking them to be cooperative with Zutha.
  + The user who authenticates a user as the owner of an item should record all the relevant details about how they did the authentication
    - This would involve recording by what means the owner was contacted, who was contacted to verify their identify (if any), what organisation the verifier was a part of, etc.
* I want to make my application distributed
  + I found a project called Shark Framework which seeks to create a standard P2P protocol for semantic information
  + It would be great to be able to eventually interoperate with other semantic services and be able to merge knowledge seamlessly using Topic Maps’ merging facilities
  + The main thing I see standing in the way of that is my still apparent need for IDs (the one that would be used for identifying an item in the URL)
  + Do I still need the base 32 human readable ID system now?
    - Durable links
      * If an item’s name is changed (at least the one that shows in the URL), I want the old link to still work
      * With a permanent ID number in the URL this is easy; just redirect the user to the URL with the same ID number but corrected name
      * If I used only the unconstrained topic name in the URL to identify the topic, then, if that name changes, what happens to links pointing to the old name
      * The old name may become more appropriate to a different item
    - I’m starting to think of some cases where it would be really ugly to try to maintain unique names across the whole graph
      * A Request item with name: “Request Item description change” might be created multiple times in widely varying contexts
      * Topic Maps specifies that all names in a given context must be unique
      * I could add qualifying information to items that represent repetitive concepts like this, but there will situations where an automated qualifying string for a name could get quite ugly
      * This Request item could be given qualifiers like “david77: Request Item description change [Javascript Tutorial [w3schools]]”
      * This example makes it apparent how ugly qualifying strings could potentially become (with square bracket qualifiers within qualifiers etc.)
      * Of course, this example is not airtight either because david77 might easily ask for another description change for the item “Javascript Tutorial [w3schools]” at a later date
      * You could add a date qualifier, of course, but this is beginning to get out of hand
    - It is clear to me now that I will need concrete IDs in my URLs and my original base 32 system seems ideal for this
  + Are there any problems with my base 32 ID system?
    - If the system is distributed, how do I make sure that the same ID is not given to multiple items?
    - There must be a way for a single node in the network (N1) to be able to pick an ID (ID1) for a new item, that won’t be picked by a different node in the network (N2) before N1 can tell N2 that it has used the id: ID1.
    - This problem could be avoid by making every node in the network have a different available set of IDs
    - Example: Node1’s IDs must begin with 00, Node2’s IDs must begin with 01… Node 800’s IDs must begin with 0X6 or something
    - Note that the number of initial characters that specify the creating network node can be increased later (once we reach 05 for example)
    - If the same concept is created twice as two different items on two different nodes, this is not a problem
      * Normal merging rules will apply and the two unique IDs become associated with a single Topic (or Item in my terminology)
    - What happens when information from 3rd parties is imported?
      * 3rd party topics are merged with existing ones using Published Subject Identifiers
      * When external Topics do not match any existing Topics, new topics are generated and new IDs are generated for them in the normal manner
* I can now forget about qualifying item names in the URL
  + The URL name will be defined separately and determined purely for human readability and SEO
* It might still be useful to add scope qualifiers to many names in the unconstrained scope
* There will still be those items for which it doesn’t really make sense to try to make unique names of
  + “Request Item description change” for example
  + What do I do about the Topic Maps requirement that items with the same name in the same scope should be merged?
  + I don’t want to abandon this convention, since it will be really useful to identify and merge duplicates
  + Perhaps I could disable merging for certain kinds of items, like comments

# Mon 27-Sep-10

* How do I satisfy the Topic Maps rule that topics with the same name in the same context must be merged, in the case of Comments and other items that can have the same name but be distinct items
  + I could prepend the item ID to the name in the global scope for items that should never be merged
    - So “Re. TMQL for HyperGraphDB” becomes something like “0X9CB: Re. TMQL for HyperGraphDB”
    - The URL name would be defined separately in a Variant as “Re.\_TMQL\_for\_HyperGraphDB”
    - I could potentially also define a Display Variant
* I think it would be worthwhile adding some minimal rich text features to the Content property
  + <code>, <emphasis>, <ul>, <ol>, <li>, and <link> would probably be sufficient
  + bold, italic, underline, highlighting, text size etc. should be unavailable as if these sorts of features are needed the content really belongs in a proper rich text document, which should be created with external tools
  + <link> will be a custom tag which will be be used instead of <a href=””>
  + <link> will only allow linking to other items in the site
  + If the link points to an item with a resource locator, the link will redirect to the actual external resource
  + Perhaps an option should be provided (in the form of an attribute in the <link> tag) to determine whether links to items representing addressable resources should be automatically redirected to the actual external site or not
  + The <link> tag should always render an html link to an internal Zutha.com location
    - Redirection will be used to send user agents on to an actual external location
    - This will be useful for preventing spammers from making use of links from popular Zutha.com pages to their own websites as a means of SEO
    - If links to external sites are never actually rendered on Zutha.com pages, then search engines will not be able to read the eventual target of the links
    - Actually perhaps search engines might follow the link, be redirected, and consider the redirected location the actual target of the link
    - The behaviour of search engines in this respect must be determined by the kind of redirect and I’m sure there will be one that will tell the search engine not to use the link to increase the external site’s rank
    - Actually, in some cases I will want to help increase the rank of another page
      * This should still be achievable by controlling the kind of redirect
    - Another use for this redirect mechanism is for recording which links people follow
* What information should I record?
  + Back when I was planning to use an XML database, I considered at one point, recording all View events as well as all Edit events
  + This would generate a massive amount of information
  + It would also bring up a whole can of worms in terms of privacy
  + I had decided to abandon the idea to try to distinguish Zutha positively in reputation for privacy
  + The main reason to record View events would be to construct a profile for each user of the kinds of topics they are interested in
    - Such an Interests profile could be used to make the site more effective by putting higher rank on Items that are aligned with the user’s interests
    - I wonder whether the public would respond differently to this invasion of privacy for Zutha than they would for say, Google, because Zutha is using the information purely to help them find information applicable to them, whereas Google is using the information to find *advertising* which is applicable to them.
  + This feature should be turned on manually
    - This way, users will feel that they are actually choosing a feature that is for their own benefit
    - I think the approach Google and Facebook and others have often taken of forcing invasive features on users is largely responsible for the backlash against it
    - People see the pretence when they are told how wonderful a feature will be for them, but not given the choice to accept the “wonderful feature” on its own merits
  + I should still record purely statistical information about user behaviour
    - The number of times a link to a particular external website has been clicked should be recorded
    - Perhaps the visit count for every item should be recorded as well
  + I should consult the user base of the site early on in the site’s evolution to ask public opinion on this issue
  + The Interests Profiling feature should only be available to logged in users
    - This also would make it more obvious that its purpose is purely for the benefit of users
  + The interests profile generated by user activity should be completely editable
    - Google has this feature, but I think there is a lot of room to make Zutha’s interests profile feature more customizable
  + The interests profiling feature should perhaps be enabled or disabled on user creation
* Every item should have a Variant for sorting
  + The sort Variant name should be generated automatically based on the user-defined item name
  + The sort Variant name, as with all names except item ID, should be modifiable by a user with high enough privileges
  + Different item types may have different rules for determining the sort Variant name
  + Example: Person items could have a rule that says the sort Variant name is generated by putting the surname first, then the rest of the name
    - This of course would require a separate first name and surname to be defined for Person items
* How do I determine the primary name to show for Items with multiple names
  + E.g. James Oliver Rigney, Jr. has pen names: Robert Jordan, Reagan O’Neal, Jackson O’Reilly and Chang Lung
    - Wikipedia uses Robert Jordan as the primary name
    - But Robert Jordan rightly fits under the scopes Author and “Wheel of Time”
* Use site traffic statistics to determine the order that fields are displayed in on item Detail pages
  + It shouldn’t be too difficult to measure which fields are being visited most often from a given context, and move those fields to the top
* UI structure
  + Zutha.com/07B3B/Flour
    - Detail page for the Flour item
    - Has the standard Item format of the Flour item at the top
      * Actually it should be extended to list “occurrence” literals below the worth bars
    - Main subpage navigation links are below main item (Details, Investment, Comments, Investments, Events)
    - Relationships are listed below navigation bar:
      * Instance Of
        + Types: product, grain
      * Sales Info
        + Suppliers: Farm 1, Farm 2
        + Retailers: Coles, Safeway, IGA
        + Prices, $1.56, $1.70, $1.30, $1.51
  + Zutha.com/07B3B/Flour/Sales\_Info
    - Main Item at the top, possibly still with occurrences listed below
    - Navigation bar
    - Subheading: “Sales Info”
      * This should have a drop down or hover menu for selecting a different Relationship type (like “Has Types”)
    - Table representing the specific instances of the chosen relationship type relevant to this item
      * Column titles: supplier, retailer, price
  + Zutha.com/07B3B/Flour/Sales\_Info/Retailers
    - Main item at top
    - Navigation bar
    - Subheading: “Sales Info : Retailers”
      * This should be composed of two menus
      * “Sales Info” should have associated menu for selecting a different Relationship Type page
      * “Retailers” should have an associated menu for selecting a different Role within the Sales Info Relationship
    - List of Items in standard format
      * Except that below the worth bars is the extra information relevant to this item’s relationship with the focus item
      * Coles
        + Suppliers: Farm 1, Farms 2
        + Prices: $1.56, $1.51

# Tue 28-Sep-10

* Editing/Item Creation (non-JavaScript)
  + Zutha.com/07B3B/Flour
    - Edit Button top right corner
    - Turns all literal fields (occurrences) into form fields
    - Also shows some new fields like alternate names and identifiers (though perhaps these should only appear for advanced users)
    - In this case literals would be: names, description, external links, subject indicators
    - Adds a “Save Changes” button to the bottom of the main item body
    - When a link to an Association-type or Role-type page is clicked, it should open in edit mode
  + Zutha.com/07B3B/Flour/Sales\_Info
    - Edit Button top right corner
    - Puts an “Add Association” button (perhaps just a plus sign image) to the left of the top of the table (and perhaps the bottom of the table as well)
      * Adds a row at the top of the table with blank input text fields in every column
      * A Submit button is created to the left of the new row (where the Edit button is for the other rows)
    - Puts “Edit” buttons to the left of every row
      * Turns every field in the row into input text fields prefilled with whatever existing values the row had contained
      * Edit button becomes a Submit button
    - Items are resolved from the text input
      * If an ID is entered (recognised by a string starting with ‘0’), the item should be easily resolvable (if it exists)
      * If a string is entered it should exactly match the (unique) name of the intended item in the current context
      * If the string does not match a single item, a small list of the closest matches should be shown for the user to select from
      * A link should be added to the bottom of this small list requesting that the user search for the item separately to find its exact name or its ID, and then come back to the Edit page and put it in
        + The link should open a search page in a new window
        + It would be useful if the resulting search page already contained the search string that had found multiple items
        + This could be done using SESSION variables or GET
        + I want to avoid allowing search strings in GET for consistency reasons, though

Normal searches will not put the search string in GET because JSF always uses POST for form submits

* + - * + I guess there’s no reason why I couldn’t still allow search strings to put in a GET request
  + Zutha.com/07B3B/Flour/Sales\_Info/Retailers
    - “Add Item” link in top right corner
    - Goes to a bookmarked form at the bottom of the page
    - This form has fields for each role in the Sales\_Info Association except the one matching the context item (Flour)
    - So in this case the form will have fields: Retailer (at the top), Supplier, Price
    - The user should enter the names of the items he wants to add to a new Sales\_Info Association (where the product role is implicitly determined to be Flour)
    - Below the input fields is a Submit button
    - If the names entered exactly match existing items in the appropriate context, those items are added to the Association
    - If a name doesn’t match, a small list of items with similar names are listed below that field
      * This small list should be in the form of a set of radio buttons, so that the correct one can be selected if desired
      * There should also be a radio button next to the input field with label “Try Again”
    - Under these suggestions should be a link to search for more closely matching items (link opens in new window)
    - Under this link should be added a “Create New” button
      * The reason this only appears after a first attempt has been made is to reduce item duplicates
    - This button should expand the associated role field into a full (or maybe a cut down) item creation form, with name, description, external ids etc.
    - So if “Tucker Bag” was entered into the Retailer input field, and it didn’t exist as an item, a set of close matches would be listed below the Retailer field (after submit button is pressed), followed by a link to “Zutha.com/Search?can\_play\_role=Retailer&q=Tucker+Bag”, followed by a “Create New Retailer” button
    - When the “Create New Retailer” button is pressed, the form should be extended to include basic fields for a new Retailer item
      * Name
      * Description
      * External IDs
    - Below this subsection of the form will remain the input fields for the other roles (any number of which may be expanded to a create new item sub-form in the same way)
    - There will still be a single submit button at the bottom of the whole form
  + Zutha.com/09NG32/Harry\_Potter/has\_comments/comments
    - For Comments, the Add Item form should always be for creating new items, not adding existing items to a new Association
    - Comments will be one of the kinds of items which are never merged
    - The Add Item button should link to the form at the bottom of the page as usual
    - From page load, the Add Item form should have the input field for the comment role expanded to a create new item sub-form
    - In this case, there is only one role: comment
    - The sub-form will contain fields:
      * Name
      * Description
      * Attitude (praise/hate/adoration/contempt/suggestion etc.)
      * Maybe some others
* Require users to log in with a more secure method once they activate real money features
  + Perhaps only when they want to do actions with real money
  + Money will be extracted from the site by logging into the user account and asking to send the money to a registered destination account (paypal, direct funds transfer etc.)
  + Adding a new registered funds destination should require high level authentication
  + If users can start off with just a Gmail account, how do they initially get high level authentication to start using financial services
  + When financial services are first activated, high level security isn’t as crucial because there isn’t yet any real money at stake
  + A Paypal account can be associated with a user account just by email validation
  + Once Paypal is added to the account, authentication is needed from Paypal in order to add another financial account (say Google Checkout)
* When a user sells an investment of real money, the money earned should remain stored in the user’s account
  + The user can separately ask to send money in their account to an authenticated destination account
* There needs to be a way to determine which properties will be shown in the edit mode for an item
  + A Person Item might be allowed to play any number of roles in all sorts of Association types
  + It would be impractical to list every allowed association that can be added to a Person item, in the edit mode of every Person item
  + If a Book item is created and has its Authors property filled in with an author A1, then the “Works Written” property should appear in the edit mode for A1
    - This seems pretty obvious
  + The question is why should the “Authors” property be visible to Book before that book has first been entered into an Authorship Association?
    - Some properties/association types should be define in the schema to be visible by default
    - The Book type should define that the Authors property is visible by default – that is, from the moment a Book instance is created
    - The Person type, should define that the “Works Written” property is hidden by default.
      * It will be made visible once the first Book item is put in an Authorship Association with it
      * Once the Works Written property is visible, it should be possible to add additional “Works Written” through the author item
  + For every Association type, at least one of its Roles must always define that Association type to be visible by default
  + There should be a tool in the edit item interface to Activate Role
    - In an item of type Person, the Activate Role interface should take the form of a set (maybe a drop down menu) of all the Roles that are playable by an item of type Person, which have not yet been activated
    - Example Roles: author, leader, team member, activist, athlete, musician, etc.
* It would be useful to allow organisations to have exclusive control over publishing their information on Zutha.com
  + An API should be available for organisations to automatically update their data in Zutha
  + For example, a supermarket chain could be given exclusive rights to define Sales\_Info Associations where their brand plays the Retailer role in the Association
  + This would allow them to publish their prices for all their products
  + Such exclusive rights to control their information on Zutha should be granted only on the condition that they update it regularly and correctly
  + If an organisation abuses their exclusive rights by failing to keep their data correct and up to date, the exclusive rights should be withdrawn
* The Search Everything button should always use redirect so that the search string appears in the URL
  + This will allow people to link to search result pages
  + I had thought it would be sufficient to only allow linking to explicit filters, but I think it would be useful to be able to link to string based searches as well
  + Also, there will be some situations where it will be desirable to link to search result pages as part of the main application UI
    - Example: when editing an Association and the string entered for a particular role matches multiple items, the user should be given a link to search in a different tab for the intended item
  + Search Everything should not contain the GET data for the current filter (the filter should remain stored in SESSION but inactive while a Search Everything action is underway)
  + This means that there still isn’t a convenient way to get a link to a page that has explicit concept filters as well as a string query
  + Perhaps then, I should make the normal “Search” button redirect so that it uses GET as well
  + Redirection would be used for non-JavaScript users, and JavaScript based GET would be used for JavaScript users
  + Instant search would still of course not affect the URL
* Naming Scheme
  + ALL items have an id of the form 0BX4R
  + All names in a given scope must be unique, as per Topic Maps specification
  + For items that represent objective concepts, scope qualifiers should be added to their names in the unconstrained scope (if necessary) so that the name in the unconstrained scope is unique
  + Examples
    - Eclipse (astronomy)
      * “Eclipse” in “astronomy” scope
    - Eclipse (software)
    - Eclipse (horse)
    - Eclipse (novel)
    - “Eclipse (K.A. Bedford, novel)” in unconstrained scope
      * “Eclipse (K.A. Bedford)” in “novel” scope
      * “Eclipse” in “K. A. Bedford” and ”novel” scope
      * “Eclipse” in “K. A. Bedford” scope (possibly)
  + For items which don’t represent external concepts (like comments):
    - Name in the unconstrained scope should have the form: “0UV5T2: I love this book! (bigsteve8)”
      * This should have a “display” variant: “I love this book!”
      * The display variant will appear wherever the item appears in full (with description, owner, worth info included)
      * The username in brackets at the end is redundant for the purposes of uniqueness, but it will be useful in many cases to help identify a comment’s identity to humans.
  + An item’s full name will always be used where only the name appears, including
    - In the table in Association-type subpages like
      * “Zutha.com/0HB4/Harry\_Potter/has\_comments”
      * If an item in this page has a name for the scope that is currently active, that name will be used
      * Example: “Eclipse (horse)” would be displayed as just “Eclipse” in the page: “zutha.com/07HW/Horse/has\_instances”
    - In the item name part of the URL:
      * “Zutha.com/0JER26/0JER26:\_I\_love\_this\_book!\_(bigsteve8)”
      * This will always use the name from the unconstrained scope
    - In the page title (the one that appears at the top of the browser window, or as the name of a tab)
      * “Zutha.com – 0JER26: I love this book! (bigsteve8)”
      * This will always use the name from the unconstrained scope
  + If the item does not have a display name, the full name will be used
* I’ve been capitalizing words unsystematically
  + Concepts which are not proper nouns should have names which start with a lower case letter
  + Examples: “book”, “novel”, “software”
  + Actually I think it would look ugly to have un-capitalized headings in the actual page
  + Perhaps I could add a “title” variant which would have capitalized words
  + The un-capitalized version of names should be used in the URL and in queries, particularly role names and relationship names
  + Examples
    - “Instance of” should appear in the url as “Zutha.com/09BH/Harry\_Potter/instance\_of/types
    - “Type” should appear in search queries in lowercase: “Zutha.com/search?type=Book”
    - Should Book be capitalized
    - I’m thinking perhaps I just want role and association names to be lowercase (except perhaps when the actual Association Type item is the current item for example)
* In the table in the Association-Type subpage, there needs to be a way to sort the table rows in various ways
  + There should be a dropdown menu below each column title (a role in the association)
  + The dropdown menu will have list of all the “sort by” options (as well as <none> to not use that column in sorting)
    - <None>
    - Name
    - Name DESC
    - Net Worth (Zuth)
    - Net Worth ($)
    - <Custom>\*
  + There needs to be a way to specify the order of precedence of each column in sorting the rows
    - This could be done by just making the last selection have the most precedence
    - I could add yet another control to determine column sort precedence – perhaps a simple number input box for each column
  + In the non-javascript interface, there will need to be a “Sort Rows” button

# Wed 29-Sep-10

* It could be useful to set up my own set of Published Subject Identifiers (PSI)
  + Instead of matching fields in GET requests to the names of items, I could use the last part of my own PSIs to identify commonly used topics
  + Examples
    - The item “Type” could have a PSI of “psi.zutha.com/type”
    - The role type “author” could have PSI: “psi.zutha.com/author”
  + Both of these examples would have additional external subject identifiers to identify them even more clearly (particularly for merging purposes)
  + I can now use the last part of these PSIs in GET requests like so:
    - “Zutha.com/search?author=08JET” where 08JET is the ID of an Author item
      * Here “author” is interpreted as psi.zutha.com/author”
      * This is a Role Type, so the search query will match all items in an Association which includes 08JET in an author role
      * This is a problem since this would match 08JET’s co-authors as well
    - “Zutha.com/search?type=comment”
      * This should be more clear cut since the “type” role should only appear once in an instance-of Association
      * “comment” is a topic type so it should be interpreted as would be intuitive
  + What about
    - “Zutha.com/0B7ER/Dejanice/murdered-by/murderers”
  + Associations will have multiple PSIs for their different scopes:
    - The murder Association would have
      * “psi.zutha.com/murdered-by” for the victim scope
      * “psi.zutha.com/murdered” for the murderer scope
      * “psi.zutha.com/used-for-murder” for the “cause of death” scope
    - Except I just realised identifiers cannot have scope
  + It will be much faster to identify items by their subject identifier than have to look up their name every time
    - Actually why should it be?
    - If both names and subject identifiers are indexed, why would subject identifiers by any faster to look up?
  + Also, permitting names in GET requests could get messy for very long names e.g
    - “Zutha.com/search?mentioned-in=09J4G:\_All\_The\_Movies\_I\_Really\_Hate!\_(BigBadBen88)&producer=Lorenzo\_di\_Bonaventura”
    - This might look ugly, and it would be better to use IDs for the right hand side parameters in this case, but why should I ban the use of names in the GET string when the names are unique
  + This method will solve the dilemma I had had about how to have a consistent ID system used by every item, as well as human readable unique identifiers for commonly used items
    - Except I do have human readable unique identifiers already since every name in the unconstrained scope must be unique
  + Each page under psi.zutha.com should be dynamically created from the item with that PSI
    - This will be a way to document my PSIs
  + I really do need to have scope in my identification of roles
    - In order to query for “/search?author=Raymond\_E.\_Feist” and not get Janny Wurts (a co-author) in the results, “author” needs to be identified as being the name of that role only from the “Written Work” scope
      * From the “author” scope, the “author” role needs to be called “co-author”
    - This precludes my use of subject identifiers in GET requests and such
* It needs to be possible in GET filter parameters to filter by a role only when it exists in a certain relationship
  + “/search?written-by/author=Raymond\_E.\_Feist&in-genre/genre=Fantasy”
  + “/search?written-works/co-author=Janny\_Wurts|Joel\_Rosenberg
    - The | character indicates OR
  + “/09N4/Book/has-instance/instances?written-by/author=Robert\_Jordan
  + Using this <Association-Type>/<Role-Type> syntax should be compulsory
* Advanced filter syntax
  + OR
    - “/search?written-works/co-author=Janny\_Wurts|Joel\_Rosenberg
      * Find an author who has written works with Janny Wurts or Joel Rosenberg
      * This is a shortcut for:
        + “/search?|written-works/co-author=Janny\_Wurts&|written-works/co-author=Joel\_Rosenberg”
        + Or “/search?logic=OR&written-works/co-author=Janny\_Wurts&written-works/co-author=Joel\_Rosenberg”
    - “/search?|discussed-in/comment=09N3G&|in-genre/genre=Topic\_Maps&|about/subject=Topic\_Maps”
      * Find an item which is discussed in the comment 09N3G OR has the genre Topic Maps, OR is about the subject of Topic Maps
      * Same as: “/search?logic=OR&discussed-in/comment=09N3G&in-genre/genre=Topic\_Maps&about/subject=Topic\_Maps”
      * &logic=AND is default and implicit if logic parameter is not given
  + Combination of OR and AND
    - “/search?discussed-in/comment=09N3G&|in-genre/genre=Topic\_Maps&|about/subject=Topic\_Maps”
      * Find an item which is discussed in the comment 09N3G and is either about the subject of Topic Maps or in the genre of Topic Maps
    - If &logic=OR parameter is given, the | characters become meaningless

# Thu 30-Sep-10

* There needs to be an association type for capturing all types of items that are about the focus item
  + There must be a separate association type for just comments so fluid internal discussions about an item are possible
    - Also, the Add Item form at the bottom of the comments subpage needs to be really simple and in the form people are familiar with
    - If comments were only available through the Discussion page, then the Add Item form would need to be the normal advanced one with a drop down menu to select the item type and an initial option to add an existing item.
  + There should be an additional tab in the navigation bar called “Discussion” for this relationship type
  + The relationship for capturing all items about a topic should be called “Discussed In” (from the perspective of the focus item)
    - The other role in the relationship should be “Document”
  + The relationship for capturing specifically comment items about the focus item should be called “Has Comments”
    - “Has Comments” should be a subtype of the “Discussed In” Association type
  + Document should be s Role type as well as a supertype for Comment, Web Page, etc.
* Format of Web Page type items?
  + One subject locator pointing to the web page
  + Name is a descriptive name of the page (taken from the actual page title if it is a good one)
  + Name must begin with the ID in the unconstrained scope since there is nothing to make the name unique
  + When linking to a Web Page item, there should be a GET argument available (like external=true) that causes the link to be immediately redirected to the external page
* For some pages, the Add Item form will need to have a drop down menu to select the type of the item to add because multiple item types may be allowed in that page
* Example graph traversals
  + How do you get Berkeley DB XML PHP API to compile with PHP5?
    - Find “Berkeley DB XML”
    - Add it to filter (so any item related to it in any way is found)
    - Search within results for “PHP API”
    - Should find something like “Berkeley DB XML PHP API”
    - Search Everything for PHP5
    - Add PHP5 to filter
    - This filter should be specific enough to have very relevant discussions about issues with the Berkeley DB XML PHP API and PHP5
  + How do you install PHP5 on Centos5.3?
    - Add PHP5 and Centos5.3 to the filter separately using Search Everything
    - This would probably be a specific enough constraint that the desired information could be found by browsing
    - It might be useful to Group By “/about/item”
    - The Group By page should contain such items (acting as groups) as “Compile”, “Install”, etc.
    - More specific results could also be attained if necessary by explicitly determining the association type to PHP5 and Centos5.3
    - “/about/item=PHP5” and “/about/item=CentosOS 5.3”
    - Otherwise there might be results like a web framework which is written in PHP5 and compatible with CentOS 5.3
  + Find the number of killings by stabbing for every murderer
    - Find the “Stabbing” Cause of Death item
    - Go to page “/used\_for\_murder/victim”
    - Group By “/murdered\_by/murderer”
    - This should show every murderer who has killed by stabbing as well as how many items are in that group (the number of victims the murderer has killed by stabbing)

# Sat 2-Oct-10

* I could generalize the concept of finding all items invested in by investors in the current item to work with all relationship types
  + When you select “people that like this item also like…” it will take you to a search page with a filter: “has\_investments/investors=Paul\_Graham/has\_investments/investors”
  + This same syntax and functionality can be applied to other association types
  + You could find all books written by any of the coauthors of a particular book
    - “written\_by/author=Daughter\_of\_the\_Empire/written\_by/authors”
* Some collections of items defined by complex relationships should fit under a single category item for ease of access
  + Example: all the companies/startups whose founders participated in Y-Combinator
  + This set of items could be manually put in a category called something like “startups founded by Y-Combinator participants”
  + Such an item isn’t really a primary concept, though, and items will only enter the set when someone is actively seeking items to go in that rather specialised category
  + A better way to define a category item: “startups founded by Y-Combinator Participants” would be to make it a Search item with a pre-defined query stored in it
  + The query would specify: Find all items of type Company which have at least one founder who participated in Y-Combinator
  + The query syntax I’ve invented for GET requests could be extended to define this query as:
    - Instance-of/type=Company&founded-by/founder=(participated-in/activity=Y-Combinator)
  + Or in a form that could actually go in a GET request:
    - Instance-of/type=Company&founded-by/founder=Y-Combinator/participated-in-by/participents
  + I just thought perhaps this could be shortened by removing the type=company restraint because it is implied by the founded-by association, but perhaps that is wrong
    - No it does work: The whole point of requiring the association-type and role-type to be included in concert is so that then a specific role-type to return can be inferred by the context-specific association-type name.
    - So “founded-by”, which might have roles “founder” and “group” implies that items playing the role of “group” should be returned. If items playing the “founder” role should be returned a different name would be used for this same association that indicates this, namely “founded”.
  + So a query for companies whose founders participated in Y-Combinator could be written:
    - Founded-by/founder=Y-Combinator/participated-in-by/participents
    - This would return all groups founded by any of this group of people, not just companies

# Sun 3-Oct-10

* The interface for investing in items should be easily accessible on every item in every list, but in order to save space and avoid clutter, it should be made visible by clicking a button
  + In javascript mode an investment interface should immediately appear below the targeted item
  + In non-javascript mode, a post request will be sent which will return the same page with the investment interface below the target item
  + Once an investment has been submitted, the investment interface will disappear
  + Actually this interface could be made visible just using CSS :hover – that would be a lot nicer
* The investment interface should have a tick box next to each of the four input boxes (one for each kind of investment pool) that indicates the money or zuth specified should be a donation instead of an investment
* For some items, only donations should be allowed
  + Event items should fall under this category, since the purpose of investing in an Event item would be to praise the Event doer for that action so all the money or zuth should go to them
  + For Bets, investments are donations, but the investors in the winning Bet will receive a prize funded by the “donations” from other contestants
    - In this case, the donation model is appropriate because the money spent should remain in the respective Bet items as a record of what how people betted
* For every item that allows Investing, donating should also be allowed
  + This will allow people to donate in the ordinary sense and have their donation give publicity to the receptor as well
  + Donations can also be used as a form of payment for actual products
    - To facilitate this, it should be possible to enter a code with a donation that will allow a retailer to check when a particular consumer has paid for their product, and then they can allow the consumer to download the product (or receive it in some other way)
    - A donation model might actually encourage people to sometimes offer more money than is asked for a product if they really like it.
  + Donations can be offered in payment for voluntary services like answering a question
  + A donation could be offered as a reward for someone answering a particular question
    - This kind of promise would probably have to be enforced purely by an honour-based system, but one in which reputation (which will be very important in Zutha) is at stake

# Sat 16-Oct-10

* How should a user’s investment value be calculated when item worth is being filtered by a set of users?
  + I think the best way would be to calculate an item’s filtered worth by simply adding up the current market value of all users’ investments in that item
  + A disadvantage:
    - A user X invests say 100 zuth in item A and 200 zuth in item B.
    - Someone does a worth filter on user X so that the worth of all items in the site is now calculated using only investments by user X.
    - It is possible that item A will be worth more than item B even though user X initially invested more in item B.
    - This is because the value of user X’s investments may have changed based on the behaviour of the investment market in the respective items.
  + I can’t think of a better way of doing this though
  + The “investment height” of all investments by users in the filter set could be added up and used to calculate the filtered worth of the item
    - The worth of the item using this method is likely to be much less than the amount actually invested by those users
    - If the users invested when the item was already worth a huge amount, then the filtered worth could show up to be extremely small
    - I think this method is worse than the first one
  + I don’t think it is possible to keep track of the actual amount spent by a user to acquire an investment
    - The main dilemma here is if a user buys an investment for say 100 zuth; it then increases in value and the user sells 100 zuth worth of the now more valuable investment. How much has the user supposedly spent to acquire the investment now? 0 zuth? But the user still has an investment worth a positive amount.
    - What if the user buys an investment for 100 zuth, it increases to be worth 1000 zuth and then the user sells half of it? Should the user be considered to have spent -400 zuth in this item?

# Sun 17-Oct-10

* It is possible for an item’s filtered worth to be greater than its total worth
  + The sum of current market worths of a set of investments is much greater than the worth contributed by them to the item
  + A better way to measure an investment’s contribution to an item worth could be to re-calculate its worth at the item worth it was purchased at
    - This would mean recording in every Investment the total worth of the item at the time of purchase
    - It would also mean that a user would need to accumulate multiple Investments in a single item if they invested in that item multiple times
    - Selling investments could get messy with this system as it would matter which investment was sold (or partly sold)
      * Say a user bought investment A at 100 zuth and it is now worth 1000. The user now buys investment B at 1000 zuth.
      * This user’s contribution to the filtered worth of the item would be 1100
      * If the user sold investment A for 1000, his contribution would now be 1000
      * If the user sold investment B for 1000, his contribution would now be 100
    - I don’t really want people to have to worry about which investment they are selling from
      * A user should simply choose an amount to sell
      * The user’s lowest investment in the “vase” should be sold first
      * This would mean investment A would be sold first and the contribution of the user to the filtered item worth is maximised, which is probably desirable
* Some domains of knowledge that are worth thinking about in terms of making the UI convenient
  + Defence of the Ancients – a Warcraft III custom map
  + The Wheel of Time

# Sat 30-Oct-10

* An item can have multiple owners
  + Ownership can be divided up uneavenly
  + Income is divided among owners proportionate to the percentage of the item each user owns
  + Open source software projects can divide up ownership of the software among the most contributing programmers
    - Each contributor will then earn money from Zutha according to the proportion they own
  + There will be some challenges in formalizing how proportions of ownership are determined
    - There will be many cases (open source software is likely to be a case in point) where there is not really any clear-cut authority to determine how ownership should be divided up

# Wed 10-Nov-10

* Allow third parties to use Zutha purely as a service to run their rating system
  + I thought of this idea while using the Peerwise tool for Engineering Systems Design where students create and answer each other’s questions
  + There are countless examples of sites that have sets of items that need rating, and my experience has been that for the most part, rating results are very poor
  + One of the main causes of bad rating results is allowing an item to be ranked highly when it has been scored very highly by a very small number of voters
  + My idea is to let 3rd party sites tap into Zutha and let users use their Zutha accounts to invest in items on their site
  + The items on the 3rd party sites would be created in appropriate categories in the main Zutha website
    - Items generated in Zutha for use by 3rd parties may be optionally hidden to the public
    - Such hidden items would only be visible to Zutha users who are also users of the 3rd party site
  + This system will only be practical once Zutha is a very well-known service and lots of people are actively using it and have their own Zuth
  + Since many users of the 3rd party service will not already have a Zutha account, they should be able to offer the option to transparently create a new Zutha account for the user in the background.
    - This should be done using OpenID technology
    - The user will be able to access their Zutha account by logging in with their 3rd party account details – that 3rd party will then authenticate them and identify them for Zutha.com
    - Once they log into Zutha.com using their 3rd party OpenID, they may also authenticate that user account with additional OpenIDs
    - Once additional OpenIDs are linked to a user account, a person can log into that user account using any of the associated OpenIDs
  + This system would be an excellent way to encourage more users to use Zutha itself
  + The service should thus be offered for free
  + It should be possible to make Zuth and use it to vote entirely within the environment of the 3rd party site
    - This shouldn’t require any extra work
    - When users of the 3rd party service create items on that service that subsequently receive investment, they will earn Zuth
    - When the 3rd party first starts using Zutha for its rating system, there will be a need for an initial injection of zuth into their system that must come from the main Zutha site.
    - The managers of the 3rd party service could harvest some Zuth for themselves on the main Zutha site and hand it out as they choose to their early users.
  + 3rd parties that use this service should be encouraged to store at least some informative content about each of their items in the Zutha item
    - It would be potentially tempting for 3rd parties to simply store the ID as the sole name in the Zutha item and do all the content processing at their end
    - I want to discourage this because I want the items they create on Zutha.com to be findable and understandable by any web user
    - If the 3rd party does not want to, and is never likely to want to make their content public, then I guess there is no reason why they shouldn’t simply store their item IDs in Zutha.
    - Even in this case, though, it would be handy for Zutha users who are authenticated with that 3rd party to be able to understand and navigate that 3rd party’s items via the Zutha interface
  + The 3rd party should be able to choose whether real money investment is allowed for their items
    - If real money investment is allowed, the mechanics should work as though the 3rd party items are any other items on Zutha (which they are).
    - The same proportion of investments should go to the item owner and the same proportion to Zutha
    - I see no reason why the owner of the 3rd party service should get a cut of investment revenue
  + The incentive to use Zutha as the rating system should be purely because it produces better quality rankings, which will make X 3rd party service a better service
* It seems still too difficult for new users to initially acquire Zuth so they can start investing
  + This difficulty is likely to be a major put-off for a lot of users
  + On the one hand, I sort of want mainly users who are really enthusiastic and capable of being constructive contributors
  + The barriers of entry that exist in the current model are likely to be effective at selecting the most desirable contributors
  + Nonetheless, I do want lots of people to participate
  + There are other dynamics at play that should produce desirable behaviour and ranking results despite undesirable users
  + One possible approach to make it easier for beginners to get started investing with Zuth would be some kind of loan system
    - I’m wary of providing automatic loans to new users as I think this is very likely to create opportunities for taking advantage of the system undesirably
    - I could make it easier for users to give loans to each other though
    - New users could easily acquire a loan or an initial sum of Zuth from their friends, but not all new users will have friends that are already Zutha users
    - If I made it safe, somehow, to provide loans to users you don’t really trust, then this may make it easier for new users to get loans off existing users
    - One disadvantage I foresee of this strategy is that it is likely to generate a lot of nagging requests for money such as annoy people so much in World of Warcraft
    - Actually I’m thinking it may not be possible to make loaning “safe” anyway, and it is also likely to open the door “cheating” by acquiring lots of loans for many different new users that you never intend to repay, and sending all this money to a new user who won’t bear the brunt of the unrepaid loans.
    - So a system of loans of real Zuth to new users it out of the question
    - There is some hope (though I sense not much) that some kind of loan of nonreal Zuth to new users could work
      * By nonreal, I mean that it doesn’t add to the pool of Zuth; it cannot be passed onto other users and its value will vanish one the user who invests with it either makes a return from it or loses it.
      * However even if I can work out a way to make this work, it will still leave the door open to “cheating”.
      * A Power User could still produce arbitrary amounts of new users and invest risklessly in arbitrary numbers of item
    - I’m now convinced that the concept of loans to new users is unworkable
  + I’m warming again to the idea that the only way to initially earn Zuth is to post some useful content
  + the main issue will be if users contribute stuff but no one finds it and thus no one invests in it so the contributor doesn’t get his initial payment of Zuth to entice him to keep going
  + If I can ensure that contributions by new users are quickly found and invested in (in proportion to their value), then that may be enough to attract enough of the kinds of users I want
    - This should be doable simply by rewarding high level users for searching out contributions by new users and rewarding those contributors by investing small amounts in those items

# Thu 11-Nov-10

* If a level X user demotes a user, then the demoted user can only be promoted again by a user of level X or above
* Provide a search tool for finding users of a particular level who don’t currently have a lot of active requests
* Provide a request system were users can send general requests to the whole body of users at a particular level
  + A user may look in his level’s group inbox for work to do
  + This form of making requests should be encouraged

# Fri 12-Nov-10

* Provide the equivalent of the Facebook Like button
  + The 4 bars of different pools of worth should be shown in a miniature form
  + There should be a single input box for entering an amount to invest and 4 different buttons to choose the type of investment
  + Provided the investment size is small enough, a user should be able to be identified using a cookie from one of his associated OpenID accounts
  + Large investments, selling investments, or editing anything should require more authentication
* Provide visualizations for item worth history
  + Use a widget from Google or something to display the worth over time data as a graph
* Provide visualizations of the current investment stack
  + Show relative sizes and positions of the investments in an item
  + Hovering over a block that represents an investment should give more details about it like value at time of purchase, current value, and the owner of the investment
  + It should be possible to zoom in and out and scroll up and down the timeline from the earliest investor to the last
  + There should be links from a list of the currently logged in user’s investments, to their place in the stack of investments

# Sat 13-Nov-10

* It should be possible to find all items in a relationship with at least one of another set of items
  + This could be facilitated by creating an Associations page for a groups of items
  + From a search results page it should be possible to another page which shows all associations participated in by at least one of the items in the result set
  + This group Associations page would have the same formatting as the Main or Associations tab of any item
  + Next to each association should be shown the number of items in the set that are part of an Association of that type
  + Clicking on roles under an Association would go to a set of items that play that role in that Association type with any one of the items in the initial result set
  + This feature would be useful for doing complex queries such as
    - Find all movies starring actors born between 1985 and 1998
    - Find all universities that have been attended by a Nobel laureate
    - Find all entertainment entities (books, movies etc.) with characters who were born in Ireland
  + This process of finding Associations from a result set should be repeatable such that even more complex queries would be possible such as:
    - Find all directors of movies that starred actors born between 1985 and 1998
    - Find all people who have been CEO of a company that has sold goods or services in a country with GDP less than 500 billion US
* There should be a feature to find the union of two sets
  + Currently there is a concept of adding an item to the filter
    - What is really being done here is a set of items is stored in order to participate in an Intersection set operation with another yet to be specified set
    - If an item such as Australia is “added to the filter” this really represents the set of all items participating in any Association with the Australia item
  + I want to divide the “add to filter” operation into a separate Union and Intersection operation and probably also a Subtraction operation
  + Examples:
    - Find all software that is either open source or made by Google
      * Find the Open Source item
      * Follow association/role: is-license-of/product
      * Click the Union button
      * Find the Google item
      * Follow association/role: creator-of/product
      * Click the Intersection button
      * Find the Software item
      * Follow the Type-Instance association to the set of all instances of Software
  + Union is equivalent to OR; Intersection is equivalent to AND; Subtraction is equivalent to NOT
    - The names of the buttons should use both notations i.e. “OR (U)”
  + It could also be useful to provide brackets so that expressions of the form: (A OR B) AND ( C OR D) are possible. E.g.
    - (works created in Australia UNION works created in New Zealand) INTERSECTION (instances of Book UNION instances of Film)
    - That is: books and films created in Australia or New Zealand
* The user interface is becoming ever more advanced and flexible and I think this endangering the usability for non-advanced users
  + If someone just wants to find out what is the most popular car on the site at the moment, they have to
    - Search for Car (or more specifically Automobile) and select the appropriate item
    - Search through the set of Associations to find the Type-Instance Association and click on the instance role to get a set of all instances of Automobile
  + I think it would be good if you could search for Cars and one of the search results would take you straight to the set of all cars
  + This could be facilitated by named searches
  + Someone can create a saved search called Cars (or Automobiles or probably both), which takes you straight to the set of items playing the instance-of role in the type-instance association with the Automobile type item
  + I think a Saved Search should be represented as an item like most other things in the site, but this leaves the question of how to present it so that opening the Cars item immediately shows the set of all cars and not yet another item that has to be navigated (because this wouldn’t achieve anything)
  + I just remembered that in the specific case of finding instances, there will be a tab called Instances on the Automobile item page
  + The concept of Saved Searches will still be applicable to many situations though, and perhaps still here so that the set of all Cars can be reached in one click from search results
  + Perhaps a Saved Search could act like a link in most cases (the way a URL item would) and would only be opened directly by advanced users
  + This would mean you could search for something like “3rd world countries” and clicking on the “3rd world countries” Saved Search item in the results would take you directly to: zutha.com/search?type=Country&classification=Third\_World

# Sat 11-Dec-10

* After reading Daemon and Freedom TM I feel sure that the information sharing platform I create with Zutha should eventually be made decentralized and distributed.
  + I want to eventually set Zutha up so that it is self-sustaining and independent of any individual person’s control or guidance. This would mean setting up Zutha’s finances to be run automatically to pay for expenses like hosting.
  + The automatic system that runs finances should be modifiable by the community, subject of course to rules similar to those that govern all modifications to the site by the community.
  + All Zutha operations should eventually be outsourced and it should eventually be entirely guided by the community of users.
  + It should be distributed and robust in the way of a living organism.
  + It should be able to adapt to new kinds of problems and attacks via the collective efforts of the community
* I want to start a collaborative project to create some of the best aspects of the Darknet as described by Daniel Suarez. He has proposed such an alternate internet in real life speeches and I think there is a lot of promise in some of his ideas.
* How to make Zutha distributed and decentralized
  + Research is being done at the moment to create distributed memory systems that work by sending packets of new data to a majority of nodes, and querying a majority of nodes whenever new data is requested
  + One challenge I foresee, is that if an item is modified and the changes are sent out to a majority of nodes, that doesn’t mean all nodes that store that item will have the latest changes.
    - Since the most recent modifications are present across a majority of nodes, the most recent version of an item can be found by picking the most recent version from a majority of nodes
  + It would be important to make sure a user is logged on only once; otherwise a user could (for example) spend his zutha twice by spending it on separated nodes at the same time. If this happened, it would not be clear how to rectify the discrepancy once the nodes communicated again.
    - The fact of a user being logged on would need to be stored in a majority of nodes as with any other piece of data.
    - When a user attempts to log on, a majority of nodes would need to be queried to make sure that user is not logged on already.
* How to manage permissions and level in a decentralized system
  + The current permissions system relies on there being a highest level user to be the ultimate resolver of disputes – like a High Court
  + A user can currently only be promoted by a higher level user
  + In a decentralized system the highest level users would have to be promoted by the community
  + The concept of discrete levels is starting to seem inappropriate for a decentralized system
  + I think it could work to use the concept of zuth for permissions as well
  + Instead of a discrete set of levels, users can be invested in to vary their influence on a continuous spectrum
  + Investing in a user should be exactly the same as investing in any other item
    - You should be able to make profit by picking a popular user early by selling some of your early investment when the user is worth more
    - You should be able to invest in the negative pool as well
  + I’ll call (at least for now) the difference between the positive and negative worth of a user, that user’s power
    - I use worth to describe the amount invested in an item (including a user)
    - I think I have used worth before to describe the total amount of zutha a user could have in cash if he sold all his investments
    - I’ll have to think of another name for net worth, or perhaps I’ll think of a different word to describe the amount of zuth invested in an item
  + I think all the rules that I developed for permissions with the discrete level system will still work with a continuous power system
    - A user with power X can specify the minimum power, Y, required to modify a particular piece of data, D, where Y<=X
    - Only a user with more than Y power can modify the power threshold on the piece of data, D
* How do users get paid?
  + Previously I had planned to specify a fixed salary of zuth for each level
  + Now I can make a user’s salary proportional to his worth
  + It is important that salary be linearly proportional to worth because otherwise there will be an incentive to concentrate worth in a single user in order to make more zutha from the salary
  + I think this system is much more elegant than an arbitrarily determined salary based on a set of discrete levels
  + I still need to determine the linear factor that determines how much salary is paid per unit of worth
  + I think this factor should be variable so that an optimum value can be found over time. The optimum value might also change over time.
  + A real money salary should use the same system. The linear factor would be determined based on how much money Zutha is earning and what proportion of that the community wants to devote to salaries.
* The concept of querying a majority of a many-node distributed system for every activity seems onerous, particularly for doing complex queries
  + Running a computationally intensive query on a majority of nodes seems very wasteful when almost all of the parallel queries will return the same result
  + The fact that all operations are performed in parallel makes the process of doing actions on a majority of nodes far more practical than it seems intuitively, but it still seems wasteful to *query* a majority of nodes for every piece of information needed.
  + Instead of sending every change to over 50% of nodes and then querying over 50% of nodes to make sure the latest information is found, changes could be sent to (for example) over 90% of nodes and queries sent to over 10% of nodes.
  + The redundancy factor (difference between percentage of nodes written to and percentage of nodes read from) can be increased to make the system more robust, particularly for handling rapid entry and exit of nodes from the network.
* Eventually I want it to be possible for Zutha to run in distributed fashion on a swarm of 3rd party servers and even mobile devices
  + The practice of querying many nodes for the same information will make it possible to identify nodes with incorrect code (either because of a deliberate attempt to hack the system or by accident).
  + It will need to be possible to send updates out to the swarm
    - Somehow, the updating system must be protected from hacking
    - Updates need to be allowed using a mechanism of collective agreement
    - The system needs to start with rules that allow changes only when a critical amount of zuth is invested in them.
    - The amount of zuth required to enable a code change should be proportional to the total amount of zuth in the system so that it remains high enough that a large number of people need to pool their zuth in order to make it happen.
    - I’m concerned that it will be difficult to make the threshold low enough that changes will actually happen, but high enough that a malicious but powerful agent cannot make malicious changes.
    - This is where being able to identify actual people would make a democratic process much easier.
    - Unfortunately, I don’t think my current system will be immune to powerful agents, like corporations, gaining too much control over the system.
  + Some devices will be just clients and will only have the code for the user interface and the knowledge of how to make queries to the distributed system. Clients should also be able to relay messages between neighbours, facilitating a truly distributed network that doesn’t require every client to connect to a central network tower as with current mobile phone networks.
  + Devices that are powerful enough may become full or partial hosts of the actual data.
    - Devices that do this will be policed by their conformance with the results of their peers.
    - Devices that return results that are inconsistent with their peers will be banned from the network
    - There will need to be further security than this because a powerful organisation could potentially host a huge number of hacked nodes such that at least some connections to the network may pick a majority of the hacked nodes and thus consider the legitimate nodes to be the hacked ones.
    - Such a hostile takeover of the network would resemble a cancer and may spread quickly if the hacked nodes continue to be treated as legitimate
    - One way I can think of to at least make such a hostile takeover more difficult, is by preventing new nodes from having as much influence
    - Nodes should follow a similar reputation based system as everything else in Zutha, perhaps even the exact same mechanism
    - Nodes that are known to be reliable will have their reputation boosted and should have more sway in the distributed network
    - Every request, either to change data or query it, must connect to a minimum number of high reputation nodes.
    - High reputation nodes will need to be able to handle more traffic because every request has a statistically higher chance of connecting to any arbitrary high reputation node than any arbitrary lower reputation node. This effect will be magnified by the fact that there are less high reputation nodes.
    - I’m sure there are better ways to make use of the concept of node reputation to maintain the integrity of the system. It shouldn’t be necessary to require high reputation nodes to handle vastly more traffic in order for them to have more sway in determining the state of the system.
  + There should be at least one fully functional parallel Development network which is used to test new code
    - This is much like the concept of beta testing servers in MMORPGs
* Does it make sense to give users subject-specific power
  + It used to seem too complex and too messy to me to have subject-specific level
  + If user power is given by the investment system then I think it might be more practical to allow investment in a user for a particular role, rather than in a universal way
  + This way, users would be given credibility specifically for the subjects they are skilled in. It would be like the level system in Daemon and Freedom TM where people are e.g. level 17 Journalists or level 13 Biologists etc.
  + There are a few problems that might be solved by adopting subject-specific power
    - Certain critical decision may be made by only allowing users with a high reputation to invest in them
    - I think investment, rather than voting, should still be used in this scenario, because it captures information about how much confidence and importance a participating voter places in his decision.
    - A good example would be deciding on changes to the codebase that should be allowed into production.
      * There should probably be a role specifically for not just programmer, but programmers who specifically work on the Zutha codebase
      * For very technical issues like security flaws and code bugs, only Zutha programmers should be allowed to participate in decisions
      * For issues of functionality, possibly everyone should be allowed to vote
* There are skills that fit into a hierarchy and reputation should pass up the chain accordingly
  + A Zutha Programmer is still a programmer and reputation he gains for being a Zutha Programmer should also contribute to his Programming reputation
  + Everyone may still have a universal reputation which is determined by the sum of zuth (positive minus negative) invested in all their reputation items
  + For new types of items that have not been assigned a specific domain yet, universal reputation should be used by default
  + Investments in actions should also carry up to the domain-specific reputation they apply to
  + Calculating the universal reputation is looking like it will be very computationally expensive task
  + Instead of re-calculating reputation every time it is queries, though, it could be updates every time a value that affects it is updated.
    - When someone invests in an action, say a code modification in the Zutha dev code, the reputation of the acting user in Zutha Programming, Programming and Universal is recalculated.
* How should rogue hosts be policed
  + The first thought I had was that every request should be sent to at least a few very high reputation nodes
  + This would be impractical in many situations, and would drastically hinder response times
  + I think clients should still be required to connect directly to very high reputation nodes regularly, just not with every request
  + Every request can be used for identifying rogue nodes
    - If a client makes a request to 20 nodes and 2 of them return anomalous results, the client should forward the anomaly to a majority of nodes in the network (as with all modification requests)
    - Other nodes in the network should then mark the identified anomalous nodes as suspicious
    - The suspicious nodes should automatically receive negative investment for being wrong
    - When clients query the network for the set of available nodes, those two anomalous nodes will now have a lower reputation and will be given less credibility when future anomalies are found
  + There should also be policing that goes on directly between hosts (nodes)
    - Every node has a list of other nodes it knows about
    - Nodes should regularly send tests to each other
    - If a large number of anomalous results occur, perhaps a large body of nodes returning the same different result, the extent of the test should be expanded to more nodes, incorporating more high reputation nodes.
    - Disputes should be resolved by finding which collection of nodes represent the majority view, weighted by node reputation
  + It would be possible for an organisation to set up a large quantity of nodes that operate correctly initially, with the intention of raising those nodes’ reputation and eventually modifying all their behaviour simultaneously in an attempt to take over the network.
    - The organisation could use their extensive resources to find mountains of zuth to invest in the reputation of their own network of nodes
    - It would be important that members of the community make an effort to make the community as a whole aware of such attempts to take over the system when they are discovered
    - It should be possible to discover that a large number of high reputation nodes are controlled by a single organisation and it should also be possible to discover that these nodes are being positively invested in heavily without sufficient merit.
    - If enough of the community is made aware of such attempts to take control, they can collectively work together to invest negatively in the offending nodes so that they cannot grow too powerful.
    - The integrity of the system relies on the community as a whole being far more powerful than any individual agent (like a large corporation).
      * I think this is easily achievable, and increasingly so, as the relative power of a single organisation appears to be diminishing in the world.
      * It should also be very difficult for a large corporation to convert their monetary power into power in Zutha
      * The structures in place in Zutha should facilitate more democratization than exists in the outside world because power in Zutha is granted by merit not money and large corporations do not have a significant amount of value to offer compared to the whole world population.
* I want it to be possible for isolated communities to be connected to Zutha via satellite and be able to function effectively despite the satellite delay
  + For example I want to be able to beam the Zutha network down into places like China and North Korea
  + I don’t think there would be any way of making edits fast when the network is connected by satellite, by reads could still be made fast
  + Edits would always require messages to be sent to most nodes in the global network
  + Some edit requests may fail because someone has made a change from elsewhere on the network which prevents your change
    - This would happen if your client had not been made aware of the remote change when it allowed you to request your change
  + This does not need to necessarily slow the client i.e. the client should not have to wait for a reply from every node in the network before it allows its user to continue
    - Sending a request should not ever halt operations on the client
    - If a request fails, this information should be sent back to the client so that it can notify the user of the failure
    - A request may succeed on the first few nodes it is sent to, but eventually encounter a conflict later in its propagation across the network
  + Read requests should be allowed to be satisfied by just the nodes nearby in the network
  + The same inter-node policing system would continue to take place over a satellite (or other slow) connection
  + Some read requests would still need to be sent to distant nodes to verify that the local network is authentic
* There will be a danger that clients may be compromised en mass so that they only connect to an inauthentic set of hosts
  + Perhaps the policing system could be extended to verify the authenticity of clients as well as other hosts
  + But if a client has already been compromised to only connect to malicious hosts, there would be no authentic hosts to test the compromised clients
  + It doesn’t seem likely that there is a way to prevent clients from being hacked and duped into thinking they are still connected to Zutha when they are in fact connecting to a separate imposter network
    - The imposter network would be able to receive updates from the real network
    - Would it be able to send any requests to the main network that would not otherwise be possible without the collection of compromised users?
    - Regimes that want to control what their citizens see may bother to compromise their citizens’ clients even if it wouldn’t give them any more control over the real Zutha, but only allow them to control what their citizens see.
    - If the device is specialised to only connect to the Zutha network, then it will be protected by the distributed nature of the network, because it will not accept commands that come from a minority of hosts.
    - Many devices are likely to also be connected to other systems like the internet, though and will not be protected from being hacked from those avenues.
    - Protection from being hacked through the internet will have to remain the responsibility of the owner of the device and will likely remain an ongoing battle for as long as the internet exists.
  + The issue that remains then is how to prevent an imposter network of hosts and clients from compromising the integrity of the main network
    - There should not be enough information sent from clients by a user for malicious hosts to be able to pretend to be that user
    - A user must have unique private information that is used to encrypt their commands to Zutha
    - Encrypted commands can be unencrypted with the user’s public key, but it should be impossible to create new commands by that user using that information; the user’s private key would be needed to construct new commands
      * This is kind of the opposite of public key cryptology used in SSL and such
      * Here all commands sent are publically readable, but only someone with the right private key can create commands from a given user
    - The time of the command would be encrypted along with the command itself so that malicious agents could not repeat commands from powerful users at will, or change the time they are run
* How should edit conflicts be resolved?
  + One of the challenges is going to be verifying when a request was sent
    - The client that initially creates a command asserts the time the command was given by the user
    - It should be possible to prevent 3rd parties from subsequently modifying this timestamp
    - However, what’s to stop a client from initially asserting a command creation timestamp in the past?
    - If conflicts were resolved by giving the earliest given command precedence, then it would be possible to override someone else’s command by sending a command for all intents and purposes “in the past”
    - It would not be possible to send things more than a few seconds or so in the past because hosts could verify that they receive the command a reasonable amount of time after they are claimed to have been sent
    - The first host to receive the client’s message has more information about the validity of the timestamp they receive
    - A host should be able to compare the claimed time taken by a message from its source with the average time taken by messages from that source. If the claimed time is wildly outside the normal delay times, the host should reject the message and ask the client to send it again
    - A compromised (or deliberately misbehaving) host could claim a correct timestamp for when it relayed the message but claim that the message has passed through a convoluted path and was sent a long time ago, when actually it was sent directly to the host a very short time ago.
    - One undesirable application of this technique would be to watch for sudden jumps in the worth of an item, and then invest in it *before* the jump in worth.
    - Exploits like this would be very dangerous to the integrity of the site because they would allow bots to collect huge amounts of zuth without contributing anything useful – much like bot-controlled high volume trading in the stock market.
* I am concerned that it is possible to invest in one’s own reputation, allowing a bot to ramp up a user’s reputation and income arbitrarily
  + The reason this was impossible with the discrete level permissions system was that users could only raise the level of users of lower level than themselves
  + I don’t think this principle can really work with the decentralized system because who invests in the highest reputation user?
  + A related flaw is that a rich user could invest heavily in another user’s reputation in, say, the Zuth Programming role
  + I could require users to have a minimum amount of reputation in a field in order to invest in the reputation of others in that field
  + I think this makes a lot of sense, but there is still a major flaw here
  + Two users could invest in each other’s reputation in a particular field indefinitely, increasing both of their reputations arbitrarily (subject to their wealth)
  + All of these exploits could be counteracted if the wider community acted to invest negatively in such cheating users
  + It might be practical to rely on the stewardship of the community in this regard, because there is an incentive to invest negatively in users that are cheating; it is an activity that will attract further negative investment and thus generate profit for the early negative investors.
  + Let’s say some agent (comprised of one or many people) builds up two powerful users by having them invest in each other’s reputation (power). More than two users may participate in this exploit.
    - There would likely be members of the community hunting for just such exploits as these in order to invest negatively in the cheating users and then advertise their cheating to the public in order to make a profit from catching them early
    - The bad will that cheating is likely to create in many people will probably drive the negative investment to build on itself as more people find out.
    - In many cases, it is possible that the negative investment will surpass the positive investment the cheaters had accumulated, thus putting the cheating users’ reputation in the negative.
    - Being in the negative in terms of overall reputation should cause salary to also become proportionally negative, becoming effectively a regular fine.
    - If a user does not have enough cash to pay the regular fine, an appropriate amount of investments should be automatically sold at the time the fine is due (same time salaries are paid), in order to pay the fine.
      * I’m not sure what would be the best way to determine which investments should be sold
      * It could be random; most recent first; oldest first; or something else I haven’t thought of yet
      * The user can chose which investments to sell himself by selling them before the due date of the fine
      * Actually more flexibility could be granted to users by allowing them to have negative cash for a limited period of time
      * Users owing money would be notified so they have time to sell the investments they chose in order to pay their debts
      * It’s interesting how similar to real life finance this is looking
    - If the user runs out of cash and assets to pay the regular fine, they should go into debt, which will accumulate as long as they have negative reputation
    - Once the user base feels the cheaters have been sufficiently punished, they will likely pull out their negative investments in order to make their profit and free up the zuth to use for other investments
    - Thus it is unlikely for a cheating user to be left to accumulate large amounts of debt
    - Even if a large number of people felt significant ill will towards a user, there would be no point putting them into huge amounts of debt because the person owning the user would just abandon it and start a new one on a blank slate.
    - It would be different, of course if a way could be found to identify real people
    - I really like this solution – I think it will be a very effective and adaptive system for preventing cheating
    - What’s wondrous about it is that I didn’t just think of some clever plan to prevent cheating, I simply followed my new reputation system to its logical extension (i.e. negative reputation and negative salaries) and found that it would solve many problems elegantly.
    - It will even allow the user base to prevent anyone from becoming too powerful because they can vote to take power and wealth away from people.
    - There is still bullying to deal with of course
* Bullying
  + It would be possible for a powerful user to do a lot of harm to other users by investing negatively their reputation
  + The main defence against bullying will I think be retribution on the bully by the wider community
  + Again, there is an incentive to find bullies and punish them because one can gain profit by it, particularly by being the first to act, which is the hardest part
  + If the bully is punished enough, he will start running out of zuth to spend on bullying
  + It is likely, I think, that someone caught bullying will quickly stop and try to convince the community that he has reformed so they will stop taking his wealth and power away
  + Someone who has resources they are willing to spend on bullying someone may hold out for as long as possible, but the community has the power to counteract their wrongful punishment of the victim while the regular fines eat away at the bully’s power.
* For items that have not been claimed by anyone yet, the share of investments that would have gone to the owner should go to the item creator
  + This will encourage new (and poor) users to find good items to add to the site and provide another avenue for them to start earning zuth
  + For such item, the creator should be shown wherever the item appears, but for all owned items, only the owner should be shown.
* The same rules of investment permission should apply to both positive and negative investment
  + General (or universal) reputation should not have any restrictions on who can invest in it
    - Activities like cheating, bullying, punishing bullying and cheating, giving constructive feedback, resolving disputes – these should all be praised/punished by investing in the relevant player’s general reputation
  + Similarly, items in the general scope/domain/subject may be invested in by anyone
  + domain-specific reputation should only be able to be invested in by other users with high enough reputation in that domain
  + only users within a certain range of (domain-specific) reputation to another user should be able to invest in that other user’s reputation
    - I don’t like this
    - It makes it too easy for users to become too powerful in a particular domain to be stopped
  + For any domain specific item (including a user’s reputation in that domain) users should only be able to invest in proportion to their reputation in that domain.
    - The proportion could even be as simple as 1:1 – if you have a reputation of 100 zuth in Programming, you can only invest up to 100 zuth in any one Programming related item or in another user’s Programming reputation.
    - This means that the only way to initially get reputation in a domain is to be invested in by someone who already has reputation in that domain
    - So how does the first user get reputation in a domain?
* Whenever a new domain is enabled, it should be made into a major Zutha event of which as many people as possible should be made aware
  + There should be a long period given to choosing the “founders” of the domain
  + Anyone is allowed to invest in their founder of choice for the new domain
  + The domain should be activated after a certain period of time and perhaps also conditional on a critical mass of investment in the proposed founders
  + A certain number of the most popular founder candidates will be selected as the founders
    - The number of founders may vary depending on the conditions of the election process
    - E.g. if the 4th and 5th most popular founders are far more popular than the 6th, 5 founders might be chosen
    - If there is a relatively smooth reduction in popularity from 1st to 20th, some predetermined number of founders will be chosen, say 6.
    - All investments in unselected candidates should be automatically sold at the moment of election
    - This may encourage voters to move their investments into the most promising candidates because having an investment in an unselected candidate will not be profitable
  + Once the founders are chosen theirs is the only domain-specific reputation that can be invested in by the “public” – people without reputation in that domain
    - I’m beginning to feel this is risky, because if most of the reputation of the founders is voted away by the public without a new election being called it could jeopardise the ability of that domain to function
    - What should happen when a user with his maximum allowable investment in a domain-specific item, loses reputation in that domain?
      * Should his existing, now illegal investments be automatically partially sold?
      * This would mean that all reputation in a domain would be anchored to the reputation of the founders
      * If all the founders’ reputation went to zero (or less) then all reputation in that domain would dry up
      * Actually not necessarily because users could be invested in each other’s reputation – this is what allow the problem discussed above where two users arbitrarily increase each other’s reputation
  + It should be possible for someone to create an item that calls for a new election for a particular domain
    - If and when that item passes a critical threshold of worth, a new election for founders of that domain will be held
    - Subsequent elections will differ somewhat from the initial one
    - All existing reputation should remain, but all users will be once again candidates for founders
    - Once a new set of founders is chosen, investments in reputation for this domain by users having no reputation in that domain, will be automatically sold
  + I’m not sure I like this system
  + I also just had the thought that users with reputation in a broader domain should still have some sway in the more specific domain, just not as much
* Reputation in a more general domain should have a smaller coefficient of investment allowance.
  + E.g. if you have 1000 zuth in your general role (not including zuth in sub-roles), 100 zuth in your Programming role and 50 zuth in your Zutha Programming role, you might be able to invest, say, 70 zuth in a given item in the Zutha Programming domain – that is 50\*100% + 100\*10% + 1000\*1%
  + This demonstrates a rule of reducing the influence of a user by a factor of 10 for every degree of abstraction away from the domain in question. Such a rule might be appropriate.
  + With this system of diminishing influence for more general familiarity with a subject, the concept of founders and elections would be unnecessary.
  + There is still the problem of two users investing in each other to lift them mutually to arbitrarily higher reputation in a particular domain
  + It would be important for the wider public to be able to discipline such cheaters
  + If a user has negative reputation in a general domain, the natural extension of the system above would be for a proportion of it to carry into more specific domains
    - E.g. a user has -1000 zuth in general, 100 zuth in Programming and 50 zuth in Zutha Programming. You are able to invest a maximum of 50 zuth in an item in the Zutha Programming domain – that is 50\*100% + 100\*10% -1000\*1%
    - This doesn’t seem potent enough, and it isn’t elegant either
* Domain-specific Reputation should be earned exclusively via actions
  + This implies that you cannot invest directly into someone’s reputation in Programming for example; you have to invest in a piece of code they wrote or a comment they made about some piece of code, or a book they wrote about programming etc.
  + This concept is looking promising, I think
  + It seems appropriate that users should earn reputation for domain-specific actions and creations (like writing a book about Programming that is well regarded)
  + It is better that boosting a user’s reputation be justified by a specific event
    - It will be easier for users to identify unjustified attribution of reputation by looking at an item or action and considering whether its current worth is justified – than by deciding whether a user they don’t know deserves their current reputation
  + Anyone should be allowed to invest in all items that attribute reputation to their owner
  + Decisions that are domain-specific should be made by users with reputation in that domain
    - The maximum amount of zuth a user should be able to invest in a single decision should be some linear factor times their reputation in that domain
    - The linear factor may even be 1
  + Certain actions might require a minimum reputation in a specific domain
  + All items should provide reputation to their owners when invested in
    - Every item should be categorized under one domain (more would be possible I think but would make things complex) which determines the categorisation of reputation received through that item
    - E.g. investment in a book should add to the author’s reputation as an author
      * An author is a user (created automatically if the author does not yet exist) and can be claimed by a real person who subsequently has control of that user
    - E.g.2 investment in a security software product should add to the developer’s reputation as a security software developer
      * The developer (or owner) of a security software product may be a company
      * Companies should be user just like authors and may be similarly claimed by the company itself
      * The company would then be responsible for determining responsibility for controlling the user
    - Security Software Developer is a subtype of Software Developer and thus reputation as a Security Software Developer should add to reputation as a Software Developer
    - The domain under which reputation is categorized should be determined by the role the owner plays in connection with the item
      * A Security Software Product has an association: Security Software Development which has a role Security Software Developer
      * This role should be specified as determining the owner of the Security Software Product
      * A very specific type like Security Software Developer is useful for specific attribution of reputation, but it may be a pain to create such a detailed type hierarchy
        + I think it would make sense to make it easy to upgrade the detail of the type hierarchy such that item that were once of type Software Product with a genre association to Security, may be modified en mass to be items of type Security Software Product once that more specific type exists.
  + It should be possible to create items that represent actions or behaviour by a particular user and wire that item to the user’s reputation
    - An example of where this might be applicable is the case of a user bullying another user
    - Someone else could create an item representing “user A bullying user B” and make user A the owner of the item
    - This item then becomes the focal point of community opinion about that behaviour
  + It should still be possible to invest directly in a user and this simply adds to total reputation
* I’m really beginning to feel that what I’m developing here (with a lot of credit going to Daniel Suarez) is the beginnings of a desirable successor to our current representative democracy system of government.
* How should edit conflicts be resolved? (continued…)
  + When an edit request is sent, the request data is hashed and the hash used to select a set of hosts from the list of currently known hosts. These hosts should thus be chosen effectively at random.
  + The selected hosts record the moment they receive the request packet and communicate with each other to agree on an appropriate timestamp to label the event with
    - I think this part would be very expensive because for all the chosen hosts to verify the authenticity of the findings of all the other hosts would require a hell of a lot of packets sent between this set of hosts
  + The hosts that were selected to determine the timestamp then send the request with the added timestamp out to the rest of the network
    - The selected hosts should digitally sign the packet that is sent out so that other hosts know which hosts have verified the timestamp
  + The request data should include a unique id, perhaps comprised of an asserted timestamp and the unique id of the client.
    - This unique id would be used to prevent hostile agents from repeating or significantly delaying the execution of a command sent by a user they don’t control
    - It would also allow hosts to avoid executing a command twice when they receive it multiple times from different sources
  + I don’t like the amount of communication that would need to happen between hosts with this system, and it just seems messy
  + The hash of the request should be used by the client to determine a set of hosts to ask for timestamps
  + The hosts should digitally sign their responses and these signed assertions of the current time should be included in the second request sent by the client
  + The timestamp for the command is selected as the maximum of the returned signed timestamps (where some responses may have timed out)
  + Hosts will accept the command provided that it includes signed timestamps from a sufficient number of the hosts determined by the hash of the first request.
  + What’s to stop someone sending out the first part of a request for a whole lot of different investments, but only sending the second part of the request (confirming the command) for investments that would do well; cancelling the rest?
  + I think I still need a better strategy
* How will item IDs be uniquely determined?

# Sun 12-Dec-10

* How should edit conflicts be resolved? (continued…)
  + I think it will be best if clients only send one request per command and the cloud takes care of the rest
  + I like the concept of randomly choosing (based on the hash of the request) a set of hosts to take control of the request
  + Somehow, this set of hosts should agree on a single controller host to manage the rest of the process
    - The highest reputation host should notify the others that it has taken control
    - There are various eventualities to handle such as the highest rep host no longer being available, but I don’t think there’s any point trying to think up solutions to every eventuality at this stage (wry smile)
  + The controller host should timestamp the request with the moment it received the request
  + It should then send out the request with the added timestamp to be propagated across the network
  + What happens if a host has executed an investment A in an item X and another investment request B comes along with an earlier timestamp than A?
    - It should be possible to recalculate the “height” filled by each investment based on the new order
  + What happens when a field is modified and a new earlier request arrives to modify the same field?
    - The earlier request should be executed and the later one cancelled because the later one was sent without correct knowledge of the existing value of the field
    - The user that requested the later modification should be notified that his modification attempt failed
* What if someone got through a malicious update that gave them control of the whole Zutha cloud?
  + Whatever they did to achieve this would be publically visible in records of any host so community hackers would be able to figure out a patch to remove the vulnerability
  + However, the malicious hacker may have modified the behaviour of the cloud so that it no longer accepts updates from the community
  + Regular full backups should be made, some of which get turned into fully functional dev environments
  + The process of turning backups into dev environments would need to be controllable from the production environment as that is where all the permissions are worked out
  + However, if the production environment was compromised, it would be desirable to have a separate emergency mechanism for booting a new production environment from a backup
  + Unfortunately, potentially anything about the production environment might be compromised, so it cannot be relied upon to make community decisions about moving to a fresh environment from a backup
  + I think the worst case scenario would be that members of the community would have to communicate as best they can and start a new parallel Zutha environment manually from the latest uncompromised backup – probably a smaller one (in terms of number of nodes) than the existing compromised one.
    - This would have to be done by people with direct access to the host hardware
  + They would then start trying to convince people to move to it and expand it, showing them that the old environment is compromised
  + Unfortunately, I don’t think there would be a way for the community to take back control of Zutha’s finances and give it to their new environment
    - Actually I’m not sure yet how secret information like Zutha’s various bank account details would be managed in the open, distributed environment
  + Obviously it would be preferable to prevent cloud from being hacked in the first place, and I think this will be practical if enough of the best hackers in the world are constantly searching the code for vulnerabilities
* How will item IDs be uniquely determined?
  + The only way I can think of to quickly and reliably generate unique IDs (without them being huge randomly generated GUIDs) is for each host to have its own private store of IDs.
  + I don’t think it will be practical to allow any host to generate IDs
  + I think there should be an elite set of trusted hosts that all have very high reputation and can be relied upon to remain up and running barring a major disaster
  + Perhaps the trusted hosts should be used for determining timestamps as well
  + A request from a client is hashed and the hash used to pick one of the elite hosts as the controller of this request
    - Since the list of elite hosts would change very infrequently and elite hosts can be relied upon to be running if they are on the elite hosts list, this process should be reliable
  + The controller host timestamps the request when it receives it and gives it a unique id – something like 1K\_6YH3V2 where 1K is the id of the elite controller host
  + The controller host then sends out the request to be propagated across the cloud
* How will secret information that only Zutha should know, be managed?
  + I think this is going to have to be figured out be smarter minds than mine
  + I think it will be necessary to outsource a constantly changing encryption mechanism
  + Various secret pieces of information will need to be passed back and forth between the Zutha cloud and outsourced security services which provide constantly varying encryption
  + I still don’t know how the cloud will be able to know something without any other agents (like people) knowing, when anyone can own and watch the hardware of a host, while knowing the full source code of the Zutha engine
  + Perhaps there will be some way to store bits of information on different hosts with some complex mathematical relationship allowing them to join their pieces together without any individual host being able to know what is formed by the whole
  + In this way, the cloud could “know” something (like the information required to access a bank account) without any individual host knowing it.
  + A 3rd party security entity should be able to share secure information with the cloud, knowing that only the cloud as a whole can know what it shares with it; an individual host does not have enough information to pretend to be the whole cloud.
  + There would need to be redundancy with this system as with everything else
    - Multiple hosts would store the same information
  + It would also be important for the arrangement of the distributed secret information to constantly change so that it is extremely difficult to acquire control over a sufficient set of hosts to discover the secret information
  + There should also be redundancy in the 3rd party agencies trusted to secure Zutha’s identity
    - That is, multiple independent 3rd parties should share the responsibility, such that if a few stop being trustworthy or fail completely, the Zutha cloud can shift responsibility to the other security agencies.
  + The more highly trusted elite hosts should host most, if not all of the distributed secret information
* How to keep secret user information secure?
  + If I outsource user authentication it might not be necessary
  + If I did need to store information about users that I wouldn’t want to be publically available – even the encrypted version of it – then I could distribute bits of the information across the cloud as would be necessary for information known only to Zutha
  + A user’s password could be encrypted into multiple dependent parts, with each part sent to a different node
  + When the user enters their password again, the same multi-part encrypted data would be generated and sent to each of the relevant hosts for verification
  + If all hosts return verification that their part was valid, the user’s password is accepted

# Mon 13-Dec-10

* Formula to determine salary
  + I had initially thought it would be important to make the salary directly proportional to power/reputation
  + The reason I thought this was important it that I was thinking still in terms of the old power model where being more powerful was increasingly more rewarding as you get even more powerful
  + Now, however, I want to adopt the opposite model: I want to encourage more power to exist among the masses than among individuals
  + For this, I think it will be appropriate to cause salary to have diminishing returns on reputation
  + The function should like something like:
    - Salary/week = A\*rep^k
    - Where A,k < 1
    - I was thinking something along the lines of: Salary/week = 0.25\*rep^(0.5)
  + The coefficients A and k should be able to be modified based on (weighted) popular decision by the community
  + One of the benefits of diminishing returns to reputation will be that it will make it much harder to boost one’s own reputation, because your salary does not scale with your reputation so it will become a slower and slower process
  + Actually I just realised this is false: your salary is still increasing as your reputation increases and since salary is a rate, you can continue to pour it into reputation – at a growing rate as your salary increases.
  + I think reputation itself should be the thing to have diminishing returns (to investments in owned items)
  + Actually this would be important because it is likely that, being in “extremistan” as Nassim Taleb would say, Zutha will result in a small number of users who receive a disproportionate amount of praise.
  + It would not be desirable for these few users to dominate decision making within their domains, though it isn’t a problem if some people have far more influence than most other individuals
  + If reputation is given by, say, sqrt(worth of owned items), this would temper the relative power of high reputation individuals
    - If a user has (worth of owned items) equal to 10,000, they will have reputation of 100, and may only invest up to 100 zuth in any one decision
    - A user with (worth of owned items) equal to 2500 will have reputation of 50
  + Salary per week may now be proportional to reputation again, perhaps even 1\*reputation given how low those reputation numbers are looking
* Pages
  + Details: occurrences and associations
  + Statistics: investment statistics
  + Association: table of Association instances
    - /view/01X5HYU/Daughter\_of\_the\_Empire/Authorship
    - => /association?id=01X5HYU&association=Authorship
  + Role: set of items that play a role in an association with a particular item. Also shows data about other roleplayers in the association
    - /view/01X5HYU/Daughter\_of\_the\_Empire/Authorship/authors
    - => /role?id=01X5HYU&role=Authorship/authors
  + Search: set of items satisfying a criteria
    - /search?logic=OR&discussed-in/comment=09N3G&in-genre/genre=Topic\_Maps&about/subject=Topic\_Maps
* How to make Zutha resilient to being compromised by a successful malicious update being pushed out to all hosts?
  + One way might be to fragment the network such that any one update would not necessarily be passed to all hosts
  + One way this could work is if different parts of the network were controlled by a different set of users
  + This could be done perhaps through a mechanism such as the one that gives different groups of user power over different domains
  + If updates had to be accepted independently by several different “factions”, as it were, on the network, then this would most likely lead to a divergence of “laws” between factions
  + If someone didn’t like an update or “law” made to the fragment of the network they are currently on, they may “move” to another network fragment or “faction”
    - This may be done explicitly or by simply focusing ones attention and energy in a different faction
  + If I’m going to allow fragmentation of rules and functionality anyway, there’s really no need to call it all the one entity: Zutha
  + I might as well design Zutha to be one instance of a family of entities that share information with a standard interaction protocol
  + How would a user on one network be treated by another network in such a system?
  + This is a similar scenario to the way different countries interact in the current world
  + I can envisage a very complex, real world like scenario, where factions or independent networks have their own faction-specific currency, with exchange rates between them as we have now between countries
  + I’m not sure how exchange rates would be determined – it would be more complex than with money, I think, because the currency in the Zutha-like system encompasses reputation and influence as well
  + Faction A would need to decide somehow how highly they value a person on network B with X reputation in currency B
  + Every Faction might have its own rules for determining exchange rates with other factions, but there would be external forces pushing them to align their choice of exchange rates with outside factors
    - One such external force would be the potential for arbitrage, where moving currency between multiple factions might allow profit generation without effort if the exchange rates have been set poorly in one or more factions
  + A user should be able to transfer his currency (and thus his reputation) from one faction to another, and act in that faction like any other user
    - Acutally it is possible that some factions may enact rules that change the rights of users originating from another faction
    - It might be necessary to prove oneself again in a new faction before certain privileges are granted
  + I think this even further distributed system has a lot of potential because it will allow for different bodies of people with different belief systems and cultures to govern themselves in different ways
  + With the single Zutha network, various minorities may be disadvantaged by the decisions made by the (weighted) majority
  + With the multi-network system, these minorities can branch off and modify those rules that did not suit them
  + It also looks promising that the multi-network system solves the security flaw of allowing updates to be pushed to all nodes
    - If some malicious hacker finds a vulnerability in the code of one network and is able to push out a malicious update to all nodes in that network, they may (in a worst case scenario) eventually obtain complete control over that network
    - But since different decision-making processes go into allowing updates on different networks, and each network will have independent security keys and often different security strategies, the malicious hacker will only be able to control a single network
    - Once it is clear that their network is compromised, users of the compromised network will be able to leave it for other uncompromised networks
    - It is likely that many networks will collaborate on updates and adopt the same updates collectively, but the safety factor is in the fact that a separate approval process goes into accepting the update on each network
  + The attempted takeover of a network would be in many respects similar to an attempted coup in a real country
    - Other networks may change their rules slightly in order to cater for refugees from the “war-beset” country
    - If enough refugees leave the compromised network, and if other networks ban trade with it, then its currency will lose all value and it will lose all its power
    - Owners of nodes in the compromised network will switch over to being hosts for other networks
    - Eventually all users will leave the compromised network, leaving whatever attempted dictators remain to rule over a soon-to-be-non-existent entity
    - One concern would be how to distinguish between refugees with legitimate currency from the compromised network who you would like to allow to keep at least a large part of their wealth and reputation when they migrate, and criminals participating in the takeover who may have illegitimately granted themselves funds from the compromised network and seek to escape to another network before the compromised one comes crashing down around them.
    - I see no reason why networks couldn’t keep historical records of data in other networks
    - Networks accepting refugees may allow them to keep their wealth from a backup of the now compromised network that is known to be from before the moment of compromise

# Tue 14-Dec-10

* It may not always be necessary for a network’s population to migrate to another network when their home network gets hacked
  + If a malicious update is passed, it is likely that, at least for a while, users on that network will still have normal communication
  + Actually other networks should be able to manage communication between users, since all share in the information sharing process
  + So when the community (or the friendly hackers in the community) realise they have been hacked, they need to publicise it and make sure hosts know they need to reverse the malicious update and fix the vulnerability
  + Instead of using the internal programmatic mechanism for pushing updates, now, the community is using natural human cooperation to achieve the same effect as the automated update
  + Individual hosts can always modify anything about the behaviour of their host, but if they do it alone, other hosts and users on the network will stop talking to them
  + If a heap of hosts coordinate a modification because they all agree that it needs doing – such as after being hacked – this is effectively like creating a new parallel network
  + As more hosts find out about the compromise and shift their resources to the rebel network, there should be a seamless of users automatically to the rebel network once it own the majority of hosts
  + Actually, when I say seamless, I just mean that even unwitting users will eventually find themselves connecting to only the rebel host without them having to do anything; there are likely to be a lot of bumps along the way
    - For a while, there will be a similar number of hosts on each opposing network, and connections to a random distribution of hosts will return roughly equal amounts of results from each network
    - Sometimes, perhaps most of the time, the results from both networks will be the same, but for the times when they differ, clients will get confused
    - An additional confusion will be that each network will store a different list of available hosts – that is, only hosts on their network
    - Clients should be programmed to recognise such disputes (similar size sets of hosts claiming the same network ID) and perhaps ask the user which network they would like to use
    - Many non-technical users may be stumped by such a request, especially since the only way to distinguish between the warring networks may be by their respective list of hosts
    - The community may publicize the choice users should make, but this may be unreliable since a sophisticated attacker may be able to emulate (using bots) a user community supporting their compromised network
    - I think perhaps the best way to resolve the dispute would be to program clients to refuse to connect to a network in dispute
    - Clients should instead ask users to temporarily connect to a different network (one outside the dispute), where they may have lesser privileges, but at least would be able to communicate and have access to all the networks’ information base
    - Having clients automatically disconnect from networks in dispute would help the honest side and hinder the compromiser because hosts will eventually migrate to the uncompromised side so it will eventually win, but knowing the behaviour of clients, a hacker is less likely to bother taking over a network because he will know that it will simply disable the network for a time – until what they’ve hacked disappears.
  + So ultimately I’ve realised the other independent networks are essential after all since they are needed while an attacked network is in dispute
  + However, in perhaps most cases of networks being attacked, users will eventually be able to return to that network once one side has won the battle (by having the majority of hosts support them)
  + If there is a legitimate dispute between two factions within a network – likely because an update was passed that many users disagreed with – then this will likely cause a permanent split of the network
  + To make a separation permanent, one faction of the dispute simply needs to change their network id
  + Once there are two separately identifiable networks, clients must make their own choice to move to the new one
* It is likely that in some parts of the world only a subset of networks will be available
  + Actually, the status quo should be that hosts act as relays for any network, even if they only host the code for one
  + If there were a place where only some networks were available because hosts there refused to refer clients through to the hosts hosting the network they are after, clients would be forced to operate in whatever network is available there
  + If the available networks do not relay clients to their preferred network they are also unlikely to offer trade of currency from the unavailable network to their own, so users might be without currency in a foreign network until they can get access to their own again
* Clients would need to be updated as well, but this should be done manually by users by manually downloading an update or when they purchase new hardware
  + Actually it would be convenient for update to be pushed automatically, particularly for users who would not otherwise bother to update
  + The fact that clients can manually update their clients again if an automatic (or manual) update was compromised makes clients collectively very resilient against attacks
* Some groups, such as corporations or governments, may want to host a closed source and encrypted Zutha network
  + The status quo for all open Zutha networks should be that closed networks should not be interacted with
  + That is, users and currency on closed networks should not be recognised on open networks
  + This doesn’t necessarily mean closed networks are being criticised, but it is just a safety precaution
  + I want to encourage companies and governments to do exactly this – make closed Zutha networks to improve their internal decision making
  + It really isn’t much of a disadvantage to closed networks if their internal users and currency are not allowed to leave for open networks – in fact it’s sort of an advantage if they are mistreating their users
  + The main thing these closed networks would want from the open networks is the information, and this is freely available, so they get everything they are likely to want
* How to verify the identity of a distributed entity?
  + This may be necessary not just for having Zutha control bank accounts but so that networks can identify each other
  + Based on what I know cryptography is capable of, my intuation says it is likely possible to set up a scheme where secret information is distributed across a set of nodes such that no one node (or small set of nodes) has enough information to discover the secret.
  + The nodes that collectively store the secret should change over time, subject to some universally readable, but non-deterministic signal, such as some approximation of the cosmic background radiation
  + With this system in place, it should be possible for some 3rd party, such as a bank, to verify the identity of the distributed entity by sending it a signed message (to verify the 3rd party’s identity) and receiving in return – after the message has been passed among many nodes without any knowing the full path taken – a message that only that distributed entity could have created.
  + This is similar to public key encryption insofar as some entity X can encode a message with his private key and a 3rd party can decrypt the message with X’s public key and know that X’s private key was used to encrypt it.
* To make networks more secure from hacking by automatically pushed updates, the process of updating a host could be a manual process, or at least one where each host is responsible for accepting an update (whether they do so automatically or not)
  + This would lead to a situation in which multiple “versions” of the Zutha software exist at once
  + It might be possible to set up a framework which makes the migration process from one version to another a seamless one for users
  + Clients would know which version of the network they are connected to and which versions they support
  + Most of the time, the newer version will have no trouble reading the data generated by the older version and so users who remain on the old system for a while will still be able to migrate to the new version whenever they want without losing any assets like wealth or reputation
  + Similarly, the database of information would continue to be shared between network versions
  + It might be that a version update entails changing the data model somewhat, but in that case conversion algorithms (hopefully two-way) may be created to enable interoperability between the versions
  + Some updates may change the rules about who is allowed to modify what (for example) and this may lead to a change on the old system that should not be permitted on the new system
  + Perhaps there would be necessity for divergence between the data in different network versions
  + If there is such divergence then it becomes a far more difficult matter to decide what data is allowed to migrate across networks
    - A user may have earned reputation that he would not have been able to on the new network version
    - Should he be allowed to keep his old reputation?
    - Should he be allowed to migrate at all?
  + Given the potential challenges involved in interoperability between network versions, I think it would be prudent to avoid a network splitting into multiple version unintentionally
  + Once the user body has agreed upon an update it should then be up to the hosts to declare that they will implement the update at an agreed upon time (by all)
  + If a host or too small group of hosts do not conform to the (weighted) popular decision of the user body, then they will find themselves on their own and ostracized by the majority of hosts
  + I think this is a much safer architecture since it does not require the internals of hosts’ machines to be open to commands from the network
    - Instead, every host is independently responsible for his own security
    - Hosts should probably even be encouraged to diverge in the way they configure their environments and implement their security measures as this will make hacking all at once practically impossible, yet all at once (or at least the vast majority) must be hacked in order to infiltrate the network as a whole
    - Any individual hosts that get hacked will simply be ostracized by the rest of the network (as soon as they actually return a minority response, that is)
  + I guess it would be possible for an attacker to slowly work his way through compromising a majority of hosts, continuing to have his hosts behave as normal until he has control of the majority, and then changing their behaviour all at once.
  + The hacker would have to get control of the actual machines that run every host, and then modify the software they run all at once
  + Provided the managers of hosts tend to be very security savvy, which I think is realistic to hope (and plan) for, I think it would be difficult in the extreme for one agent to gain control of a majority of independent, very well secured hosts.
* Querying
  + I think all queries should be a single string, and should be passed through GET requests as a string to a single parameter, like query=
  + The query language should scale smoothly from the syntax people are currently used to (namely Google and ilk) to much more complex functionality
  + Thus a search string containing a space separated list of words should simply find items with all of those words somewhere in the title or description of the item.
    - Items with more keywords in the title should be prioritized
    - Items with all keywords in the title in the order given should be prioritized most of all
  + AND, OR, and “-“ keywords and brackets should be available to make more complex logic
    - E.g. to search for jobs relating to apples: query=apple AND (job OR occupation) –computer –“Steve Jobs”
  + An expression with an = sign in it is treated with different logic
    - expressions on both sides of the = sign must be made up of identifiers that uniquely match single items
      * I’ve changed my mind on this, I don’t think this will be desirable
    - the left side of the = sign can be either a Role or an Association/Role pair
    - the right side of the = sign is simply an expression which returns a set of items
      * this could be a unique identifier like: Harry\_Potter
      * it could be a keyword-based search string like: harry potter
        + this may return multiple results
    - the result of an expression of the form A/R=E is the set of all items participating in an association of type A where one of the items returned by expression E plays the role R in that same association A
    - the result of an expression of the form R=E is the set of all items participating in an association in which at least one of the items returned by E plays the role R
    - R must uniquely identify a Role item
    - Any alphanumeric string containing an underscore is treated as a unique identifier
      * Not sure yet how single words should be turned into unique identifiers
    - A single Role type may participate in many Association types, so using the R=E form of the = expression and omitting an explicit Association is like using a wild card \*/R=E. Any Association of which Role R is a member may be used here
    - Examples
      * author=rowling
      * Authorship/author=J.K.\_Rowling
      * CEO=(educated-at=Harvard\_University)
        + Find all entities (probably companies) in an association in which the role of CEO is played by an entity (probably a person) who was educated at Harvard University
  + Group By
    - Only one Group By clause should be permitted in a query
    - Group by clause should take the form: groupby:A/R or groupby:R
      * Where A is an Association identifier and R is a Role identifier
    - Examples
      * Type=Book groupby:author
      * Type=Person groupby:murder/cause-of-death
    - That last example shows a need for determining the role played members of the result set
      * When I say “Type=Person groupby:murder/cause-of-death”, do I want the grouped results to include both murderers and victims or just one or the other
      * If I want only victims in the results, I want to not only specify that all Person items in the result play the role Victim in some arbitrary association of type Murder, I want them to play the role Victim in the same Murder association which yields the cause-of-death they are grouped in
      * Perhaps something like this: groupby:victim\Murder/cause-of-death
        + This would group victims of murders under their cause of death
        + One way this result could be constructed would be by first finding all items playing the cause-of-death role in a Murder association which also includes a victim role. For each cause-of-death X, find all items satisfying the query (not specified) AND victim\Murder/cause-of-death=X. If the number of results for a particular cause-of-death is zero, don’t show that group
  + Participation in a Role or Association
    - What if I want to find all items that participate in a Murder association, but don’t care which role they play in it?
    - I could use wildcards: query=Murder/\*=\*
      * This means find all items that participate in a Murder Association where any of the Roles is played by any item
      * I think this syntax should be allowed, but I also want there to be a less ugly solution
      * Then again, this sort of query is likely to be pretty rare and obscure, so I guess it isn’t so bad if the syntax to do a weird query is weird
    - If I want to find all items playing the Author role in any association: query=author\\*/\*=\*
      * This is really starting to look horrible
    - If I want to find all items participating in an Association which includes the Author role: query=author=\*
      * Ah… that’s much nicer
    - I wonder whether the \*\\*/\* syntax is really contributing anything
      * query=author/\*/\*=\* looks somewhat nicer I think
      * the backslash doesn’t really help and I think more obscure characters should be avoided
      * the semantics are still consistent
        + author=\* is equivalent to \*/\*/author=\*
        + Authorship/author=\* is equivalent to \*/Authorship/author=\*
* Complex queries are likely to be quite conducive to parallel processing
  + Many query are comprised of multiple independent queries which are combined using set operations
  + Foreseeably, a query could be divided up into sub queries and dispatched to many different hosts all around the world
  + Once the sub queries are complete, the results can be combined, with different parts of that operation potentially being done on different machines as well, before finally being returned to the user
  + The decentralized system I’m envisaging would also involve every operation being duplicated in parallel for the various benefits that such massive redundancy would provide
  + The duplicate parallel processing could have another benefit: results could be calculated and sent to the client in a different order from different processors (a processor here being a particular set of hosts working on a query without redundancy)
  + If you did a query like query=groupby:type, a lot of results can be returned without the full set of results being yet known
    - Processor A might start returning an incomplete set of items which participate in an association which includes a role called Type
    - Processor B may be at the same time sending the client a different incomplete set of items satisfying the same condition. Some of the results from the two processors will overlap right from the start, but the overall rate at which results are received by the client will be, for a while, almost double the rate achievable with only one processor returning results
  + The client would keep track of which hosts sent it which items and hopefully all the sets will be equal by the time all the hosts have finished sending it results
  + The client does the work of merging the results for display to the user as they come in from multiple processors at once in real time
  + Though this system requires a lot of processing by the client, I don’t think it will be unrealistic to expect this much of the average client of the next decade
* Do I need the visual/browsing based system for doing complex queries that I partially developed earlier?
  + The query language is definitely sufficient to do an enormous range of queries – I cannot, off the top of my head, think of any criteria of selection that cannot be implemented with the query language I just developed above
  + I do think it is likely that many people will feel more comfortable with clicking buttons than learning a query language, no matter how intuitive and convenient it is
  + If the visual and textual querying system are integrated well, the visual querying mechanism could provide an excellent way to learn the more powerful textual querying language
  + Yes I think I should definitely continue designing a visual querying system
  + I don’t think it would be possible to make the visual querying system as powerful as the query language, but it should be possible to make a visual system powerful enough to cater for almost all common use cases
* Visual querying system
  + Group By and Union buttons may be all that are required
  + Actually an Intersection command might be needed for finding items related to a specific item through any Associations and Roles
  + Group By
    - On any page of results, either a search results page or a Role subpage of an item, there should be a Group By button available
    - Clicking Group By will open an embedded dialogue box (by javascript or Post) with a filter textbox and a list of Role/Association/Role triples that are applicable in the current context
    - Typing in the filter box should filter the list to entries containing all the keywords
      * This should operate similar to a combo box
    - The user must select a single entry in the combo box as each entry is a link that uses GET to request the page with the chosen group by clause appended to the existing query
    - How should wildcards be dealt with?
    - Should every possible combination of literals and wildcards that would yield results be listed in the Group By dialogue?
    - Actually explicit wildcards would not be necessary; the target Role must be specified and thus wildcards in the R1 and A positions of a triple of the form R1/A/R2 could be simply left out
    - The obscure case of R1/\*/R2 can I think be safely left out of the visual interface; users must use the query language to do such a query
    - Example groupby lists
      * Current query=type=publication
        + Author (possibly left out since it’s the same as Authorship/author in this case)
        + Authorship/author
        + Publication/Authorship/author
        + Book/Authorship/author
        + Publisher
        + Publishing/publisher
        + Publication/Publishing/publisher
        + Book/Publishing/publisher
        + Publication-date
        + Publishing/publication-date (publication-date is an occurance, but should be accessible the same way as a role)
        + …
    - Seeing that ugly list, I think it would be far cleaner to only show the single part target Role
    - Examples
      * Current query=type=publication
        + Author
        + Publisher
        + Publication-date
        + Genre
        + …
      * Current query=type=Book OR type=Person
        + Author
        + Publisher
        + Publication-date
        + Genre
        + Age
        + Sex
        + Occupation
        + Spouse
        + Parent (supertype of father and mother)
        + Father
        + Mother
        + Child (supertype of son and daughter)
        + Son
        + Daughter
        + Institution (here including the association type Educated-At/Institution would improve clarity)
        + …
    - I think perhaps the Association type should be shown, but not the first role (that of the results being grouped on)
    - Example: query=type=Book OR type=Person
      * Written-by/author
      * Published-by/publisher
      * Published-on/publication-date
      * Categorized-as/genre
      * Works-in/occupation
      * Spouse-of/spouse
      * Child-of/parent
      * Educated-at/Institution
    - Here I have used the context-specific Association names
      * Given how much clarity this adds, I think it will be worth going to the trouble to determine association names based on context
      * I just remembered Topic Maps do this much context mapping easily
      * It was giving Roles context-specific names that I recall deciding was too hard to bother with
      * I only ever encountered a single circumstance where context-specific role names would be useful, though: cause-of-death/method-of-murder for a murder-association
      * Cause-of-death is easily a sufficient description regardless of whether the context is the victim, the murderer or the cause-of-death itself
    - Every entry in the Group By list will be a link with the url predetermined to append the selected groupby clause to the existing query
    - The grouped results page
      * All the groups with more than zero results in them should be shown as a normal item list
      * The main item title should link to a page showing the members of that group
      * What query will show the members of a selected group?
        + Let the query that had yielded the results shown when the Group By button was clicked be q1
        + Let the selected group by relationship be of the form: A/R – Associaton/Role
        + Let the selected group item (the one whose link was clicked) be G
        + The query that will show the members of the group G is: q1 AND A/R=G
      * There should be a smaller non-text link on each Group item that links directly to that group item’s details page
      * Each Group item should show the number of items that it contains in the same style as properties are shown on item details pages
  + Union
    - Every page containing a set of items should have a Union button
    - When Union is clicked, a marker should be added to the visual query builder representing the current query
    - At the end of the visual query builder list of markers should be placed a Union symbol:
    - The user should be returned to a blank Search page with the above changes made to the visual query builder
    - After searching and browsing until another results set is shown which the user would like to add to one currently remembered and represented by the marker in the visual query builder, the user should press Execute
    - Pressing Execute should combine all the separate queries stored in the visual query builder and compile them into a single query string which should be executed as a GET request
    - The automatically generated query string should be shown in the search textbox – this will help users to learn the query syntax
  + Intersection
    - This should work similar to Union except that after Execute is pressed, the intersection of the two queries (one before pressing Intersection button and one after) should be found
  + Subtract
    - Similar to Union and Intersection except that the Subtract set operation should be performed (results before pressing Subtract minus results from query entered after pressing Subtract)
  + Operator Precedence
    - A user has a set of results shown, A, which might have been made by using multiple visual query operators, but crucially, he has pressed Execute so all of queries he has performed thus far have been compiled into a single query which results in the set of items A
    - He then selects a visual operator, say Intersection, and then finds another set of results B (using searching/browsing)
    - Regardless of any normal rules of precedence (such as Intersection has higher precedence than Union – I think), this Intersection operation should perform the operation
    - If necessary, brackets should automatically be placed around the visual query made thus far (before Union was pressed)
    - I have so many bells and whistles already that I’m thinking I might as well add open and close bracket buttons as well
  + Brackets
    - Pressing open bracket adds ( to the end of the visual query and takes you back to a blank search page as with other operators
    - Until this bracket is closed with a matching bracket, pressing Execute only shows results for the visual query entered after this unfinished clause
    - The part of a visual query that is being used to generate the current list of results should be highlighted or underlined or something
    - Pressing close bracket ) should execute the full expression just closed by that added bracket
    - Another press of Execute may be required to add the finished bracketed clause with visual query blocks before that bracketed clause
    - Multiple presses of Execute in a row may be required to fully execute a deeply nested set of bracketed clauses
  + How to query on a Role being played by any of a set of items?
    - In the query language this is done with the syntax: query=written-by/author=(born-in=Australia)
    - Basically I need a dialogue similar to Group By except that instead of grouping by target roles where the current results play the source role, I want to find all source roles where the target role is played by any of the items in the current set
    - Consider a page with the results of the query “born-in=Australia” displayed
      * I can group by such things as works-in/profession or has-nationality/nationality
      * These candidate association/role pairs are found by looking for associations in which members of the current result set play a role
      * But this is exactly what I’m doing to find written-by/author=(born-in=Australia)!
      * I just realised this query can be performed using a Group By operation
        + Query=born-in=Australia groupby:wrote-works/works
        + This will display a set of Groups – one group for every work written by any of the authors born in Australia
        + Most of these groups will likely have just one member: the Australian author of that item
        + The only exception would be co-authorships where multiple Australian authors may be authors of the one work and would therefore all be members of that work’s group
      * The problem with this method is that it the Group By page is designed to be intermediary
      * The (main) links on items do not go to that item’s page, but to another Search page – in the case of the works by Australian authors query above, likely a Search page with only one item on it: the author of the selected work
    - I think it would be convenient to have a counterpart to the Group By function that has essentially the same functionality, except returns a normal page of results instead of a set of Groups
      * The set of association/role pairs should be named from the context of the role played by one of the items that will be in the new result set
      * Also, instead of the role shown in the association/role pair being the one that will be played by members of the new result set, it should be a role that is played by members of the current result set
      * Examples from query=born-in=Australia
        + Written-by/author

Counterpart to wrote-works/works in Group By

* + - * + Employs\_(profession)/worker

Counterpart to works-in/profession in Group By

Employs\_(profession) must be distinguished from association: employs as it applies to employers

* + - * + Employs/employee

Counterpart to employed-by/employer in Group By

* + - * + Is-nationality-of/person

Counterpart to has-nationality/nationality in Group By

* + - * This feature does actually add some subtle new functionality
        + This “Reverse Lookup” function makes the role of result items a wild card, but makes explicit the role that must be matched by one of the items in the current set
        + The Group By function makes explicit the role of result items, but uses an implicit wildcard for the role that must be mached by one of the items in the current set
    - I think “Reverse Lookup” would be an appropriate name for this function, but a shorter and less ambiguous name would be ideal

# Wed 15-Dec-10

* Query Language
  + I haven’t catered for querying more than two roles at once in a single association
  + Examples of syntax I like better
    - Type=Book work:written-by(author=David Friedman)
      * Results matching this query must play the role “work” in an association of type written-by (named in the context of role “work”) where David Friedman plays the role “author”
    - Type=Book groupby:written-by(author)
      * There’s sort of some ambiguity here where groupby appears in the lexical position normally occupied by a role identifier
      * In some ways, though, this syntax makes it clear that the role played by items in the results is not specified
    - Type=Publication groupby:author
      * Here it is known that author should be treated as a role of any association because of the absence of a bracketed clause at the end
    - Type=dog victim:murdered-by(murderer=(type=person gender=male),cause-of-death=pitchfork)
      * Find dogs that were murdered by men with pitchforks
      * If I use a comma to separate role conditions in an association like this, it may be necessary to ban commas from item identifiers
* Item Identifiers
  + Any quoted string should be considered a literal item identifier
    - Example: “J.K. Rowling”
  + The url form of an item name should double as its human readable identifier
    - Example: J.K.\_Rowling
  + If string literals are used for identifiers, then it won’t be possible to search for the presence of string literals in title or description fields of items
    - Since this is how users are used to using string literals, I think it might be better to leave them like that
  + I could use square bracket delimiters for explicitly declaring a string an identifier
    - Example: [J.K. Rowling]
    - I like this better – it looks more like a programmatic unit
    - I think some programming languages (VBA?) actually use this syntax for specifying identifiers that contain spaces
  + There will be some characters that it would be desirable to allow in item names, but not in item identifiers
    - Commas would be confusing to have in item identifiers if commas are part of the query language syntax
    - Commas would be desirable to name items like “62 Pascoe Avenue, Kilsyth”
    - Names with commas should be converted to url/identifier form by replacing spaces with underscores and removing commas (and likely some other special characters)
    - “62 Pascoe Avenue, Kilsyth” would become 62\_Pascoe\_Avenue\_Kilsyth
    - The square bracket delimiters would permit using the full names inside, including commas and other special characters
  + Examples
    - Resident:lives-at(address=[62 Pascoe Avenue, Kilsyth],start-date>=2002)
* How to secure users’ real world privacy?
  + The distributed system would be ideally suited for automatic and random assignment of multiple remailers so that the physical source of messages would be practically unobtainable
  + The client would choose several random paths through multiple hosts (here acting as remailers) and uses the standard cryptographic remailing procedure to obscure the path taken by the message and (for now) its contents
  + At the end of each path, the receiving of the message (there will be multiple via different remailer paths) decrypts the message and then distributes it across the distributed network
  + In this way, users retain their online identity via their public/private key identity (or possibly other means), but nothing about their real world identity can be known
  + I would like to eventually make this level of privacy the default in order to create an environment in which users feel safe sharing any information whatsoever. There will still be reputational disincentives to say things, though (for users with high reputation)
  + Another feature of universal absolute privacy would be that the community as a whole would feel far braver about taking on powerful entities as everyone would be confident their participation in the conflict through Zutha would be untraceable to their real world, more vulnerable identity
  + There will on the other hand be benefits to users connecting their virtual identities to their real world identities
  + In the case of authors and public figures in particular, it would be desirable for them to have an accessible presence in the Zutha virtual community
  + Also, if a user account is openly linked to an OpenID, no amount of security precautions on Zutha’s part can prevent interested parties from finding out everything there is to know about the owner of the OpenID account, which might be a lot, and with a lot of resources, could be considerably more
    - For example, a Google account acting as an OpenID could have lots of personal information in it including contacts, emails and explicit personal contact details
    - Even if the owner of the Google account has associated no information with it and does not even use it except to identify himself as that Google username, it might be possible for resourceful parties to crack Google historical databases and find out information like the IP address from which the account was created
  + I think therefore, that, in order to avoid misleading people about their level of privacy, Zutha should only provide the additional privacy features its distributed system can provide after explaining to them what other precautions they need to take to ensure their security
    - Some precautions taken on the part of the user can be checked by Zutha, and should be checked before the account can receive a status as highly secure/private
  + I think an explicit privacy meter for each user would be a better way to cater for the conflicting benefits of privacy and connections to real world identities

# Thu 16-Dec-10

* While reading about the Grey Goo Scenario in Future Imperfect by David Friedman, it occurred to me that some of the decentralized concepts I’ve been thinking about for Zutha could help for developing safe (from unintentional over-replication) nanotechnology.
  + The principles that I brought to bear in this context are massive redundancy and the collaboration of many independent “nodes”
  + What makes these principles powerful in the context of the decentralized Zutha platform that I envision is that no small group of nodes is capable of changing the behaviour of the whole group
    - This means both that it is extremely for the whole to malfunction because malfunctions of a minority of nodes are quickly eliminated by the malfunctioning nodes being evicted from the group until they reform
    - It also makes it very difficult for some agent to intentionally take control of the system since consensus rules; no changes that can’t be agreed to by practically every node will be accepted
    - In the case of a system that is completely run by simple algorithms, without any human decision making involved, the consensus must be airtight at the time the distributed system is then created. After the distributed system is established, its decentralized, consensus-requiring nature can prevent attacks or mutations on a small number of nodes from affecting the whole.
  + This principle could be brought to bear in designing nano-replicators by making every replicator into an independent regulator of its neighbours
    - Each nanobot would be programmed to watch other nano-bots and keep track of its behaviour
    - A collective distributed database keeping track of the behaviour of every nanobot should be possible, though it is likely that each nanobot’s individual memory capacity would be relatively small, so the distributed memory system would have to spread information quite thinly.
    - If a nanobot misbehaves, say by diverting from its assigned task, by not self-destructing when it should, by reproducing itself too many times etc. the collective “cloud” of nanobots will “correct” it – perhaps by destroying it.
    - Such a system could be very robust because of the principles outlined above
    - As long as no decision of the collective mind ever relies too few individual agents, the logic programmed into the system when it was first created will persist
  + How would the software in a nanobot cloud be enabled to be safely updated without undoing the benefits just described?
    - I wrestled with this issue with Zutha for a while, and the solution I came up with was that again, there should be no point at which a decision is concentrated at one point
    - For Zutha, it was possible to make decision making about updates distributed by having each human controlled host make an independent decision about updating his host’s software
    - The problem with a nanobot cloud, if it is controlled by a single human entity, such as an organisation, is that that organisation would want to make a decision internally about how to change the behaviour of the nanobot cloud, and then inject that decision into all the nanobots.
    - But even if the decision making process of the organisation is robust and safe from mistakes (it might have been made in a distributed way, perhaps by a system like Zutha) the very fact that they are able to inject their collective decision into all the nanobots at once is a major vulnerability
    - If there is a system in place to do something to every nanobot, then it doesn’t conform to the specification of a collaboration of independent agents
    - The only way I can see that updates to the collective nanobot cloud could be possible safely, would be to have the control of the nanobot cloud divided up among many different controlling agents
    - The controlling agents could be individual people, organisations, Zutha networks or some other intelligent, decision making entity that can collaborate independently with the others to manage the nanobot cloud
    - In order for an update to the whole nanobot cloud to succeed, every independent controller would have to agree to that update
    - It may not always be necessary for every “faction” of nanobots to be updated in the same way, but some kinds of updates, if enacted on only a minority of the nanobots, would trigger behaviour in that minority that is not considered acceptable by the majority. The majority would then act to correct the unacceptable behaviour in the way programmed – perhaps by destroying all the nanobots in the misbehaving (according to the majority) minority group.
    - As an owner of a nanobot “faction”, there would be no way to get around the fact that some kinds of updates would be impossible without the agreement of the majority of other factions. It is precisely this property which protects the collective nanobot cloud from accidental or deliberate destructive behaviour (like turning the biosphere into grey goo) from a minority.
    - If it was discovered that there was a weakness in the defensive software of the majority that enabled a minority to do something undesirable to the majority – such as taking control of other nanobots and thus potentially the whole cloud – it shouldn’t be too difficult to get the majority to agree to a security update which closes the vulnerability
    - It should not be too hard to use cryptographic technologies to enable control of a particular group of nanobots to be given to only to a particular controlling agent (who has the private key or perhaps some more advanced equivalent)

# Fri 17-Dec-10

* URL structure
  + /Item/01XHR4/Lift\_Web\_Framework
    - Redirects to details page
  + /Item/01XHR4/Lift\_Web\_Framework/subject-of
    - Redirects to Association page
  + /Item/01XHR4/Lift\_Web\_Framework/subject-of/comment
    - Redirects to Role page
  + /Details?id=01XHR4&name=Lift\_Web\_Framework
    - Shows item properties and associations
  + /Statistics?id=01XHR4&name=Lift\_Web\_Framework
    - Shows investment statistics
  + /Association?id=01XHR4&name=Lift\_Web\_Framework&assoc=00X5BV3
  + /Role?id=01XHR4&name=Lift\_Web\_Framework&assoc=00X5BV3&role=00X7V2QN
  + /Search?q=type=[Dog] victim:murder(murderer=(type=[person] gender=[male]),cause-of-death=pitchfork)

# Sun 19-Dec-10

* I want to eventually host arbitrary files on the Zutha distributed network
  + Once the distributed system is in place, hosting files could be done very reliably and safely – safely because it would be very difficult to host malicious files when doing it legitimately will allow the file to be quickly identified as such by the community and removed, and modifying an existing legitimate file would be very difficult because you would have to modify the file on many separate servers all around the world at once.
  + How will files be managed on Zutha?
    - Files should be items like any other, with their own ID and the filename as their unique name
    - A file will have two associated address, one for viewing and modifying the file’s metadata like filename, author, perhaps geodata etc. and one for directly accessing the file itself
      * /file/0V8XBN40G2/Secret\_Leaked\_Document\_(Pentagon\_Papers\_2,2012-11-08).pdf
        + This url links directly to the pdf file
      * /item/0V8XBN40G2/Secret\_Leaked\_Document\_(Pentagon\_Papers\_2,2012-11-08).pdf
        + This shows the normal Zutha item in normal format with associations and occurrences
    - A file may have multiple revisions, which can be accessed directly by putting the revision number in the url like
      * /file/0V8XBN40G2/r3/Secret\_Leaked\_Document\_(Pentagon\_Papers\_2,2012-11-08).pdf
    - Only the latest peer-reviewed (and accepted) revision becomes the active one, so the active revision – the one accessible without specifying the revision in the URL – may not be the latest one
  + In the short run, I want to only host the avatar images using this setup
  + Later, I will start allowing html documents to be hosted so people can put creative works up directly on Zutha as html files associated with a Document item
  + Eventually (if I’m still making the decisions alone by this time) I want to start enabling large files like videos and software
  + It is very likely that once this robust distributed system is set up to host large files, people will want to start using it like Rapidshare to host pirated data and perhaps secret (leaked) documents
  + I’d like to support whistle-blowers and this setup would be perfect for deflecting attack from any small group of individuals who would thus be vulnerable, however, if it was safe to be a whistle-blower, it would also be safe to upload sensitive information that would be purely dangerous in public hands, like detailed manuals for building weapons of mass destruction or documents revealing the identities of spies and leading to their deaths.
  + Unfortunately I think it is increasingly going to be impossible to prevent such documents getting into public hands if there are people who want to get them into public hands. Thus, I believe the benefits of enabling fuller global transparency will outweigh the costs. In fact I think this would be true even without the excuse that the sensitive data would get out anyway.
  + Individual hosts would need to somehow be protected from prosecution for hosting material others don’t want them to host i.e. leaked documents; copyrighted material etc.
    - If it is possible to harass just a few hosts enough that it frightens many others away from hosting that material openly, then the benefits of the decentralized system will be undermined by human psychology.
    - I think the solution here is again encryption
    - Documents should be fragmented into many encrypted pieces and distributed across many hosts in a redundant way
    - No host should be able to know which parts of what files they are hosting
    - If this is achievable, it appears to solve the problem of individual hosts being targeted, because there would be no grounds on which to target any particular one and no way for any host to selectively change their behaviour to acquiesce to pressure.
    - A powerful entity could chose hosts at random and pressure them to shut down, but this is only likely to strengthen the resolve of others to build new hosts and keep existing ones running
    - The crucial vulnerability of hosts that know and can modify what they are hosting is that they can bow to pressure in a small enough way that it just feels like prudence. Also, the pressure to remove specific sensitive material could feel justified, but the pressure to close down when there is no evidence you are doing anything wrong (or at least not hosting a particular piece of sensitive material) would feel very unjustified and will likely incite resistance.
  + Another benefit of distributing large files redundantly and in pieces is that different hosts can host different amounts of data depending on their capacity
  + All semantic data (or at any given time almost all) will need to be stored locally on every host for efficiency reasons though
    - Since it is text data, this should not be too great a barrier
    - If there were billions of nodes, this could mean Terabytes I guess, though likely much less with compression
    - Terabytes will soon be a very insignificant amount of storage space though I think
    - For each host to store all file data though (like video and eventually probably high resolution 3D worlds) will soon require far more than petabytes
* When the decentralized version of Zutha is first being rolled out, most people will still likely want to access it through the conventional web
  + The concept of a client that I developed for the decentralized system could possibly be done purely in a browser with JavaScript
  + PC and mobile software could be released as well, but many people will just want to visit the website from anywhere as they are used to
  + The only purpose of the conventional website then, would be too download the JavaScript client to your browser
  + There could be many mirror sites that host the JavaScript client
  + Unfortunately, the nature of the web is that you connect to one host at a time, so if a mirror site for the client was compromised, users who pick that mirror will be vulnerable.
  + If an online list was kept of the safe client mirrors then the responsibility could be left with each individual to ensure they aren’t using a compromised client
  + It isn’t currently a common occurrence for large reliable sites to be compromised and stay so for long, so I don’t see why it should be any more common in the future – unless Zutha is specifically targeted from entities with a lot of resources, which, if Daniel Suarez’s Daemon is to be taken at all seriously, is perhaps not that unlikely.
  + Compromising a client mirror should only make it possible to deceive users who connect to it; unless the mirror downloads key loggers and such to the user’s device, it should still be impossible to steal identities with a compromised client.
  + To steal a user’s identity, or at least forge some requests to Zutha from them, it would be necessary to infiltrate all of the ID mechanisms they are using
    - Actually perhaps not quite all, since if just one ID mechanism became unavailable to the real user and they needed all to authenticate with Zutha, they would be rather stuck.
    - Probably just a minimum number (depending on the user in question) of ID mechanisms would have to be infiltrated in order to take control of a user’s account
  + Some users may be more vulnerable than others, in fact some might have, say, a single Google OpenID as their identification mechanism, but such users are unlikely to be powerful users, since more powerful users will have been encouraged – if not forced – to adopt greater security measures
  + Having infiltrated a user’s client should not help in infiltrating their various 3rd party ID mechanisms because any traffic associated with ID that flows through the client would already be encrypted.
* Is there any reason to divide and encrypt normal item data in the same way as files?
  + I don’t think so
  + The open system for item text data would already have long been in place by the time Zutha starts hosting large files so it would be difficult and unnecessary to move to an encrypted system
  + The distributed secret storing mechanism will likely already be in place by the time storing of large files is enabled. Encrypted file storage should use a similar mechanism, with the main difference being that anyone may request a distributed encrypted file, whereas only specific institutions (like banks) will be allowed to access the distributed secrets.
* Resolving name conflicts
  + If someone tries to create a new item with the same name as another item, all items with a variation on that name should be shown so that the user can determine whether the item he is trying to create already exists, or whether he needs to specify yet another variation on that name
  + Let’s say there already exists an item called David Jones, and it refers to the Australian chain of department stores.
  + Someone tries to create another item called David Jones, this time referring to the Guernsey politician
  + Since “David Jones” has not yet been established as a name for multiple items, the user will be asked to either merge his new item with the existing one, or specify that they are different items
  + If the user specifies that they are different items, he must rename each conflicting item appropriately
    - David Jones the department store chain can be uniquely identified by it being of type Department Store Chain (or perhaps by a super type of this like Franchise).
      * It could thus be renamed “David Jones (Department Store Chain)”
      * It could also be renamed “David Jones Limited” which is what Wikipedia does
      * It should probably be given both names, with one as the primary one
      * For now, for the purposes of a later example, we’ll assume it gets called “David Jones (Department Store Chain)”
    - David Jones the politician could be renamed initially “David Jones (politician)”
  + A new disambiguation item will be created automatically and given name “David Jones”
  + The two, now renamed, David Jones items will be linked to the disambiguation David Jones item
  + If someone tries to create a new David Jones department store chain item (having not looked for it first), and they immediately call it David Jones Limited, how will it be identified with “David Jones (Department Store Chain)”?
    - Perhaps on all item creations, a confirmation page should show items with similar names and ask the user to confirm that they are not recreating an existing item
    - Also, with JavaScript enabled, a list of similar existing item names should appear when the user types in their new name, as an additional way to try to avoid duplicates
  + If there is a disambiguation item whose name appears in full inside the name of a new item, it could work to automatically link the new item to that disambiguation item
  + When a new disambiguation item is created with name X, a search should automatically be done to find items containing the string X
  + If someone now tries to create a new item called David Jones, referring to the Welsh politician, they will be asked to determine whether they mean one of the items in the disambiguation list, or a new item
    - If they rename their item “David Jones (Welsh politician)” then it will be linked as another ambiguous item to the David Jones disambiguation item, and the “David Jones (politician)” item will remain as is
    - If they try to rename their item “David Jones (politician)”, they will be asked to rename both “David Jones (politician)” items and a new disambiguation item called “David Jones (politician)” should be created
      * The “David Jones (politician)” disambiguation item should be linked to the “David Jones” disambiguation item
      * I think any items identified as ambiguities associated with “David Jones (politician)” should also be linked in the same with to “David Jones”.
      * If more levels of disambiguation items were created, such as a child disambiguation item of “David Jones (politician)” like “David Jones (Welsh politician)” if there were multiple Welsh politicians named David Jones, all ambiguities of child disambiguation items should also be associated directly with all its ancestor disambiguation items.
    - The second option would be preferable
  + It would be ideal if when someone began by trying to create an item called “David Jones (Welsh politician)”, the system identified another item with a similar name “David Jones (politician)” and asked the user if he wanted to create a “David Jones (politician)” disambiguation item
  + It would be easy for name collisions to be identified when for example, an item called David Jones Limited is created and there already exists one called David Jones
    - Since “David Jones” is a substring of “David Jones Limited”, this collision can easily be identified and the user should be asked to make a disambiguation item for David Jones
  + More complex name collisions should also be identifiable robustly, if the naming system is standardised well enough
  + The part of a name outside brackets should have an exact string overlap to be considered a collision
    - Perhaps this overlap should also be required to be at the start of both strings
    - You wouldn’t really want the system complaining about collisions between “Stallion” and “The Black Stallion” for example
    - There may be times when prefixes might be different for items that should be merged or linked to the same disambiguation item
      * Example: “The Herald Sun” and “Herald Sun”
    - There might also be times when two items have the same prefix but are unambiguously different
      * For example, you probably wouldn’t want “Light Switch” and “Light Cone” to be both associated with a disambiguation item called “Light”
    - I think more complex rules would be desirable for identifying collision, such as a set of words that are likely candidates for prefixes and suffixes that don’t substantially change the meaning by their presence
      * “Limited” would probably go in such a list for suffixes
      * “The” would probably go in the prefix list
  + If the base parts of a name (the beginning part before the bracketed part) collide, how does the bracketed section affect identifying a collision?
    - If one name has no bracketed qualifier and the other does, this is a collision.
    - Example: “David Jones Limited” and “David Jones (Department Store Chain)”
      * The base parts “David Jones Limited” and “David Jones” collide because they differ only by a suffix containing only words in the list of suffixes that commonly don’t much affect the meaning of a name
      * Since one has a bracketed qualifier and the other doesn’t, these two names will collide
      * The common ground between the two names, which we’ll call the “collision string” is “David Jones”
      * If these two names referred two different concepts, the “David Jones Limited” item would need a qualifier and both would be associated with the “David Jones” disambiguation item
    - If one name’s bracketed qualifier contains a subset of the words in the other, this is a collision
    - Example: “David Jones (politician)” and “David Jones (Welsh politician)”
    - Words like “the” and “a” should be kept out of bracketed qualifiers, but in case some users put such words in, they should be excluded from identifying collisions
    - Example: “David Jones (a politician)” should still collide with “David Jones (Welsh politician)” even though one’s bracketed qualifier does not have a subset of the words in the other. With generic words like “a” excluded from consideration, {politician} is still a subset of {Welsh, politician}
  + Regardless of how good I design the automated collision detection system, there are bound to be exceptions; both false positives and false negatives
    - False negatives can be left to be manually resolved
    - How do I deal with false positives though?
    - If the system regards two names as colliding it will keep reporting the collision whenever it searches for collisions
    - I don’t think it would be desirable to try to make exceptions such that some items would be considered collisions but are explicitly excluded from being considered collisions even though their names still match the rules for a collision. This would just be inelegant.
    - I must therefore ensure that the collision rules do not result in unresolvable false positives
    - I think they are already pretty good
    - If you get a false positive on base names, you can easily add different bracketed qualifiers and remove the collision
    - It is possible for two such items to not share a disambiguation item, provided no disambiguation item collides with both
    - A general rule that disambiguation items link to all items they collide with will work quite well I think
  + It will be desirable for qualification strings to be (at least optionally) automatically picked
    - This will make it a smoother, more appealing process to create new items
    - It will also improve consistency
  + Items have different names in different context/scope
    - An item’s name in the global scope should collide with its name in all more specific scopes because the global scope name should have a superset of all bracketed qualifiers
    - The standard rule should be that a single base name is the default for all scopes and the more specific the scope, the less qualifiers
    - It should be possible for names in more specific scopes to be generated automatically by subtracting qualifiers
    - This means qualifiers will need to be associated with the scope in which they do not need to appear
    - It would be easier if qualifiers were always automatically generated based on users’ selection of qualifier scopes
    - Example: “David Jones” is associated with scopes Wales and Politician
      * the simplest way to generate automatic qualifiers from these scope items would be as “David Jones (Wales, Politician)”
      * I’m not sure whether commas should be included automatically
      * It would be desirable for users to be able to customize what is generated automatically while still retaining links between the text qualifiers and the associated scope items
      * So a user could rename the scope Wales to Welsh and omit the comma
    - Having qualifiers associated directly with scope items would make detecting collisions for the bracketed qualifier part of a name easy: one name must have a subset of scopes of the other for there to be a collision.
    - Users should be able to rearrange the order that qualifiers appear in, by simply rearranging the order of the list of qualifier scopes (where scopes are simply other items)
    - Sometimes, automatic renaming of Scopes for when they appear as qualifiers, might be possible
      * The item Wales could have the name Welsh in the scope of Nationality
      * A user selecting scopes for “David Jones” the Welsh politician will select from items he is associated with
      * When picking items as scopes, they will be grouped according the Association/Role they are linked to the current item through
      * Wales will be associated to “David Jones” through Citizenship/Nationality (or something like that)
      * This means the name that the item Wales takes will be based on the scope Nationality because that is the role it is playing in the current context
      * Therefore the default qualification name associated with the scope Wales, should be Welsh
    - When specifying qualifiers, users should be able to rename any scope and specify different delimiters between each qualifier/scope
    - The interface could be like this

|  |  |  |
| --- | --- | --- |
| **Association/Role (dropdown)** | **Qualifier Name (text field)** | **Delimiter (text field)** |
| Citizenship/Nationality: Welsh | Welsh |  |
| Employment/Occupation: Politician | Politician |  |

Preview: “David Jones (Welsh Politician)”

* + - If nothing is specified for Delimiter, it should be a space
    - Another example: John Smith (Medal of Honor, 1880)

|  |  |  |
| --- | --- | --- |
| **Association/Role (dropdown)** | **Qualifier Name (text field)** | **Delimiter (text field)** |
| Awarded/Medal: Medal of Honor | Medal of Honor | , |
| Awarded/Date: 1880-10-18 | 1880 |  |

Preview: John Smith (Medal of Honor, 1880)

* + - The form for specifying name qualifiers should appear on the main item details edit page
    - If a desired qualifier scope has not yet been associated with the item, that can be done first and then that new associated item can be added to the qualifier scopes
    - A better way for picking the qualifier scopes would be to have checkboxes next to every item in the associated items section
    - Some Association types, like Comments, will not have every instance visible because it would be impractical for items that are associated with hundreds or thousands of comments to have all of those comment items listed with checkboxes next to them in the edit details page
    - Most likely all of the associated items from which qualifier scopes would be picked will be visible however
    - Association types which have more than about 10 instances should have the list paginated (with independent paginations for each Association type)
    - When associated items are added to the qualifier scopes list by checking them, a new entry should be added to the Qualifier Scopes table
    - I think the checkbox system should be the only way to specify qualifier scopes as it would be too difficult (and now unnecessary) to design a drop down menu that would capture the complexity of multiple instances of multiple role types per association type and multiple instances of multiple association types
* Query Language
  + I don’t like the compulsory square bracket syntax (i.e. “developer=[Google]”) for specifying identifiers
  + When placing conditions on roles (as in this case containing the developer Role to be Google), the most common thing to do would be to specify a specific item
  + However, when the query words appear on their own (i.e. don’t come after an equals sign), they should behave as they would in Google, as keywords to matched with any words in items’ title or description
  + I want the query language to be as general as possible, though, and features like Reverse Lookup should be available on keyword-based results
  + Someone might want to find all items which have an associated comment containing the word “fabulous” in their title or description
  + The syntax for this should be: comment=(fabulous)
  + Thus to do the Reverse Lookup operation on a keyword (or indeed any) search, just surround the existing query with brackets () – call this (…) - and place it in a query like so:
    - role=(…)
    - role:association(role=(…))
  + to match an identifier containing spaces (i.e. a non-URL item name), just surround it with standard quotation marks
    - so comment=”fabulous” would attempt to find items associated with a comment item titled “fabulous” which will not exist
    - author=”J.K. Rowling” would find books written by J.K. Rowling
  + to match a single word identifier on its own, or to match an item name exactly, use name=”Item Title”
    - name would be a reserved word and would not be permitted as a Role type name
    - example: name=”J.K. Rowling” would match only the item called “J.K. Rowling” exactly
  + name and description should both be reserved words and should be usable anywhere in queries
    - name=(tiger) would find any items with the word “tiger” in their name
    - description=(“Chris Barnett”) would find any items with the exact phrase “Chris Barnett” in their description field
  + null keyword
    - I had initially planned to make this a keyword which can be used to specify that a Role is not present in an Association
    - Now I think it would be cleaner if there were an actual item named “null” which is used to fill any Roles in an association that are unoccupied
    - Some association types should automatically be instantiated for new items of certain kinds, and all roles except the one that the new item plays should be played initially by null
    - This is particularly applicable where there should only be a single instance of an association for a particular item
    - Example: a Book item should have exactly one association called Authorship (or written-by)
      * When a new Book item is created, a new Authorship association should automatically be created with the “work” role played by the new Book item and the author role played by null
      * When an author is subsequently specified, it replaces null in that role
      * Additional author roles can also be specified of course
    - The null item would have to be made the subtype of every type
      * This might be difficult to do in any of the pre-existing topic map constrain languages
    - How could the null keyword be used in queries to match item in an association where one of the literal values (like a date) associated with the association, is missing?
      * Here null should be treated like a keyword
      * If null is being matched to a literal field, it becomes a keyword which states: where this occurrence/literal field is not present in the association or is present, but empty.
  + External ids should be allowed in queries
    - Example: http://en.wikipedia.org/wiki/J.\_K.\_Rowling
    - These would mainly be used for queries done through an API, but I want the API query language to be the same as the user query language
    - To prevent syntax confusion, I think external ids, which will be URIs, will need to be delimited by a new symbol
    - Example: author=[http://en.wikipedia.org/wiki/J.\_K.\_Rowling]
    - I think the square brackets are more appropriate here than for normal internal name identifiers
  + The | character should be allowed as an alternative for OR
  + The & character should be unnecessary and would make URLs ugly
    - Any whitespace is implicitly AND, and the AND keyword can be used explicitly as well
  + How should a query like skin-color=(black|brown) be treated?
    - Should black and brown be considered unique identifiers?
    - The definition I made above was that in a bracketed clause with floating words, which this is, those words should be treated as search keywords, not identifiers.
    - The syntax for specifying Black and Brown as identifiers could be: skin-color=Black|Brown
    - It would never be necessary to use AND in a situation like this, so a simple OR clause delimited by | instead of OR, work quite well
    - I think | should be reserved for exactly this purpose
    - The query: Black|Brown should return exactly those two items
    - The query: Black OR Brown should return items with either of those words in their title or description
    - This behaviour also makes things more consistent because skin-color=(Black|Brown) will behave the same as skin-color=Black|Brown
    - Actually there is a situation where it makes sense to place an AND condition of more than one item on a role: if that role can appear multiple time in an association
    - Example: find Books co-authored by Raymond E. Feist and Janny Wurts
    - This query would ideally look something like: type=Book work:written-by(author=”Raymond E. Feist”+”Janny Wurts”)
    - + is a better alternative to & for use as AND because it will appear in URLs
    - So + and | are the AND and OR operators for identifiers
    - The above query could be written more verbosely as: type=Book work:written-by(author=”Raymond E. Feist”,author=”Janny Wurts”)
    - I’m not sure how the OR syntax could be expanded equivalently though
      * Without | It might be necessary to write: skin-color=Black OR skin-color=Brown
      * If the whole role:association(role=…) syntax was needed for each expression in the OR clause things would get really messy
      * Hence the | syntax
* Syntax Stuff
  + Associations should have a capitalized name in the global scope
    - Authorship
    - Sales Info
      * url form: Sales\_Info
  + role-specific association names should be lowercase with words delimited by dashes (-)
    - written-by
    - manufacturer-sales-info
  + Role names should be capitalized with normal spaces (translated into underscores for URLs) just like any other item
  + It should be possible in the query language to omit the first role from a role:association(role=…) clause
    - Authorship(author=(name~=Tolkein)) would return any items in an Authorship association where one of the authors has Tolkein in their name
    - This example appears to have introduced the need for a fuzzy matching syntax, for which I have introduced ~= meaning “contains”
    - To make parsing this syntax unambiguous, any bracketed part of an item name must be preceded by a space (underscore in URL)
    - Also, no item names may end in an underscore or space
  + If I require that every Association has a unique role-specific name for all of its roles, then specifying the target role explicitly in the role:association(role=…) clause becomes unnecessary
  + The query: written-by(author=”J.K. Rowling”) implies that the results play the role “work” in an Authorship association (whose name is written-by from the work role) for which J.K. Rowling plays one of the author roles
  + I’m not sure whether or not I should keep the role:association(role=…) syntax or not
  + Groupby should still use the groupby:association(role) syntax I think
    - Now it’s possible to specify which role the results must play in the grouping
    - The query: groupby:written-by(author) will group items that play the role “work” in an Authorship association, by the author role of that association
    - The query groupby:Authorship(author) would also show co-authors under groups for authors that have collaborated
  + I think I should put order-by functionality in the query language, instead of as a dropdown menu
    - This will limit its use to more advanced users (though I could still keep the dropdown Order-By menu), which would be a good thing because I want results to almost always be shown in order of worth
    - Then again, aren’t I just trying to create the most useful service I can; if people want to order by something else, they should be able to
    - So I’ll keep the Order-By dropdown control
    - All I’m changing is that instead of appearing as a separate GET parameter, Order-By will appear as a clause in the query string
* Ordering results by relevance
  + The only time I can think of that it would make sense to use keyword based relevancy ranking would be if there are keywords hanging alone at the top level of the query
  + Example: type=Book lift framework
    - This should do a conventional Google-like query for “lift framework” and filter the results for items that are of type Book
    - The results should be ordered by relevance by whatever engine I use to do full text querying (Lucene or Google or Bing or something)
* What happens when items are merged?
  + Should events associated with both items be merged into the new one?
  + If not, then which item’s events should be kept?
  + Perhaps, as long as every event retains information about which of the merged items it was associated with (which is possible since there are two different IDs), all the events could be put under the new item.
  + It should be possible to construct a history diagram with forks in it for representing merges
  + The merged item will be associated with both the old IDs, but one will have to be made the primary one
  + If you want to find out what the two items looked like before the merge, you would have to examine the history of the item
* What happens when an item is deleted?
  + The data should be retained and remain associate with the ID
  + When (most) users try to access the URL of the deleted item, they should receive a 404 Not Found error or perhaps a “This item has been deleted” notice
  + Only high level users should be able to view the actual item and possibly be able to undelete it

# Mon 20-Dec-10

* ID scheme
  + Z looks a lot like 2, especially when handwritten
  + If I use up X to separate the Host ID from the rest of the ID, I don’t have 32 characters anymore
  + It also annoys me to have to waste 2 characters that are the same for every ID: the initial 0 and the X separating the two parts of the ID
  + The first digit is a 0 so that IDs can be reliably identified in queries by being the only words starting with numbers
  + This leaves open using other numbers than 0
  + The first digit could determine the number of characters used by the Host ID
  + I then don’t need the separating X, but can use X in the normal characters set again
  + Examples, where the host ID is surrounded by square brackets:
    - 00
    - 0H784DG
    - 1[0]8B3
    - 1[G]60V
    - 2[X1]N3B3
    - 3[00B]NV59E
    - 4[08HE]BG3
  + This leaves me up to 9 base 32 digits for storing the ID of each host
  + I don’t think I’ll run out of host IDs since this gives me the capacity for about 3.5\*10^13 of them
  + The character set is now: 0123456789ABCDEFGHJKLMNPQRTUVWXY
  + That’s everything except: I, O, S, Z
* What should be done with the money Zutha earns?
  + I had once planned to use a proportion of Zutha’s earnings to divide up among users who accumulated large amounts of Zuth, as a reward.
  + At that stage, I believe I was thinking in terms of encouraging users to make more Zuth being the end goal – with the implicit assumption that if the main drive among users was to make more Zuth, then they would find the most effective means possible to do that and hopefully those means would entail making the most positive contributions to the site.
  + I now realise that by making accumulating lots of Zuth effectively the end goal, there is a risk that it will attract people whose only motive is greed (or at least morally blind self-interest) and who will use any means they can find to satisfy that greed by accumulating Zuth and the money that (potentially) comes with it.
  + I fear that the kind of people I want to dominate the Zutha “market” (for want of a better term) – people who are energised by making the world better for themselves and others – will be eventually (if not rapidly) overshadowed by people willing to cheat and be ruthless to accumulate power and wealth.
  + One way I think this risk can be reduced is by targeting the incentives towards the people who I want to dominate the “market”
  + Instead of zuth being an avenue to acquiring money, it should be purely the currency of influence within Zutha
  + While I had always intended zuth’s primary purpose to be influence within Zutha, I had previously not realised how much of an incentive that could be on its own
  + Having influence within Zutha will mean more and more as the distributed, decentralized society as a whole grows in power
  + Early on, having a lot of zuth may only buy you more influence in determining the relative value of an item or deciding the outcome of some dispute such as what an item should be named or what its description should be.
  + Later, zuth will offer influence over the allocation of Zutha’s real world money
    - Until I’m able to launch Zutha as a fully independent distributed entity, I will probably just manage the allocation of funds myself, but base my decisions on rules I develop for resource allocation by investment of zuth by users.
    - The resource allocation system should include restrictions on the kinds of things towards which Zutha may allocate resources. Such rules would constitute a sort of “constitution” which would require a far stronger voice from the community than normal to change.
    - At the simplest level users would invest zuth in various charitable organisations or community projects that they believe in, and Zutha would allocate its funds towards those causes in proportion to the relative worth assigned to each by the community.
  + Eventually I see the platform of Zutha having the capacity to take on more and more of the roles currently played by a government, so having more zuth may increasingly be akin to real political power
    - To steal an example from Daniel Suarez, a distributed entity like Zutha could potentially have control of a private, robotic law enforcement force, whose rules of behaviour could be determined by the Zutha voting mechanism.
    - In the more immediate future, Zutha might be managing a large scale software development project. Zutha users could be responsible for choosing the active decision makers of the project in real time. A group of people (or perhaps just users if everything was done through cyberspace) would be selected by zutha investment to start off as leaders of the project. If the community as a whole later felt, in compelling enough degree, that the leaders (or a leader) were unsuitable, they may vote to change the leadership.
* One of my objectives early on was to make it possible for a teenager to earn real money from his bedroom by contributing his time and intelligence to the Zutha community
  + Abandoning the payment of real money to users purely for accumulating zuth will reduce the capacity of an unskilled teenager to make money by “playing the stock-market” of Zutha, or by grinding out lots of small contributions that are only each likely to ever acquire small investments of zuth.
  + However, I’ve realised that not only would allowing a teenager to make money in that way also enable the Zutha equivalent of high volume traders and Chinese gold farmers to make money in that way, but I don’t believe that was, or is, really my objective.
  + It is still possible for a teenager in his bedroom, as with any other intelligent entity, to contribute some creation of utility to the community and earn money from it by taking a cut of real money investments into the associated item.
  + For example, a teenager may create awesome music using synthesiser software on his computer, or make cool software, or make popular videos or create any conceivable work which others like enough to stake their money on its value by investing in it.
* What happens to the distributed secret storing mechanism (responsible for entrusting financial resources to the distributed entity) when a Zutha network splits in two (or more) pieces?
  + If a money-owning Zutha network splits in two very suddenly, I don’t think there’s a way around either a dispute over control over financial assets or mutual loss of control over financial assets.
  + If enough hosts decide to share their data with both networks it might be possible that the two separate networks both have access to the secret and would be able to give conflicting requests to financial institutions regarding their assets.
    - This would be the most undesirable scenario I think
  + If, as is more likely, the split occurs by similar size groups of hosts disagreeing about implementing some code change, one group will start up a new network with a new ID and take with them whatever secret data they held
    - The secret data should be useless on the new network, but it is nonetheless missing from the old network
    - If all the hosts that had held a particular piece of a distributed secret leave the network, then the network loses access to that secret
  + I think it is actually important that it is possible for a network to be deprived of its funds if enough hosts leave it
    - The whole point of giving each host independent choice over which changes to the code it accepts is to provide a security mechanism against infiltration and hostile takeover of the system.
    - If an update gets through the community filtering system which enough people believe will jeopardise the integrity of the network, then that (large enough) dissenting group should have the power to deprive the existing network (the one conforming to the undesirable change) of its financial (and possibly other kinds of) assets and thus the majority its power.
  + If a network pushes through a change which enough hosts disagree with to leave the network, then they have a strong incentive to revise their decision and try to woo the dissenters back, particularly if enough hosts have left to deprive them of (some of) their distributed secrets and thus (some of) their assets

# Tue 21-Dec-10

* What is the incentive for adding new items to the site that you wouldn’t have a right to own (like a Book you didn’t write)?
  + I had planned at one stage to have a percentage of real money invested in items go to the creator
  + Later I decided that a more deserving item owner than the creator (such as the author of a Book item), should get to co-opt ownership from the creator
  + There’s an additional issue though: what happens when two separate items get created that are supposed to represent the same thing
  + I think it will be possible to avoid this happening too much with clever name collision detection techniques, but it is likely to remain possible for some to deliberately create a duplicate of an item in an attempt to earn revenue from it
  + Due to the combination of this risk and the fact that it just doesn’t seem elegant to have ownership given to the creator by default and passed on if a more deserving owner is found, I don’t think the item creator should get a cut of item earnings unless he is also the deserving owner of the item (as in the case of writing a comment)
  + Reward should be given for users creating new items and writing comments about them which explain why the item is good and worth using/watching/investing in.
  + There will be an incentive for users to invest in young (poor in zuth) users that they identify as deserving of praise because if that user later becomes much more respected, and invested in, the early investors in him will be rewarded.
  + There will also be an incentive to invest in early comments of praise for an item you expect to become very popular in the future
    - Since the most profitable way to invest is investing in nearly empty items, there will likely be a tendency for goodwill towards a particular item to be spread out in the form of investment in associated items like comments praising that main item
    - Later investors will do this because the main item will already be filled with a lot of zuth and be deemed a less profitable investment, whereas good comment items about it will potentially provide greater profitability because they don’t yet have as much zuth in them
    - Early investors should invest in the auxiliary items to a main one also, in anticipation of later investors doing the same
  + So new and young users can get into the game by finding items not yet added to the site which they believe are deserving of popularity, adding them to the site, and writing good comments/reviews about them
    - It would be important to write a good review as an item adder, in order to attract notice to yourself – otherwise, people may love the main item, but forget to consider praising the user responsible for them finding it on Zutha
* How to merge two items that both already have investments in them
  + If no one had yet sold any investments, the investments in the merged item could be constructed by simply repeating the combined investments of the two items in the order of their timestamps
  + Investments in the primary item (the one who’s ID becomes the primary ID of the merged item) would only have to have one field changed: the “height” value that determines how of the conceptual “vase” that investment would have filled up at the time it was invested.
  + Actually this is all that would need to be changed for investments in the secondary item as well, because in the process of merging, the Investment items would automatically be associated with the new, merged, item
  + The tricky part about merging will be when the merged items have already had investments sold on them
  + Let’s specify some variables
    - h = the conceptual “height” added to the vase by an investment
    - h0 = the height of zuth in the vase at the time of investment
    - H = the height of zuth in the vase at any given time
    - V = the conceptual “volume” of the vase filled by an investment (amount of initial investment in zuth)
    - W = the current worth of an investment in zuth (the volume removed from the vase if sold now)
  + The number that should be used to calculate the other two again, once merging has occurred, is V.
  + When a user sells part of an investment, say amount X, he diminishes W by X, leaving W1.
  + A new value of h - call it h1 - can be calculated by how much height needs to be taken from the top of the vase to provide X volume
  + A new value of V, V1, can be calculated by finding the volume added between height h0 and h0+h1
  + There seems to be an accounting error in this process, because X will be larger than V-V1 (provided the sale occurred when the item was more valuable than at the time of purchase), so who is the extra zuth taken from?
  + Well there will now be some investments made whose h0 value is less than H. What is the implication of this?
    - Well their W value will be less than their V value
    - V will still be the same the way it is currently defined though (as the amount of zuth initially invested)
  + I somehow need to capture the fact that a little bit of zuth is taken from every existing Investment to account for X
  + I don’t think it will be practical to recalculate V for every investment item every time an investment is sold
  + It should be possible to calculate a value of V that adds up to the total value of the item, based just on the h values of all the investments
  + So when an investment is sold (of size X), a new value of h should be calculated for that investment item, H should be reduced by the same amount as h is reduced, and that’s it.
  + When a merge occurs, the V values of all the investments in both items should be calculated and these amounts should be used to recalculate the h values in the merged item
  + Calculating the V values would simply be a matter of virtually selling every investment from the top (most recent) down and keeping track of the sell value of each; this would be its V value
  + The two sets of now independent investment items (one from each item to be merged) should be combined into a single timeline and virtually invested back into the merged item, earliest first
  + The impact of a sale on other investments is effectively that all investments above (after) the sold one have their h0 values increased
  + When an investment is sold, all more recent investments could have their h0 values increased by the same fixed amount
    - Again, though, I think this would be unnecessary, unless there was some reason users wanted to know the real time value of V
    - They might want to know the initial price they paid for an investment, but this is fixed and based on the initial value of h0, which I think should therefore be fixed
    - The height at which an item would be sold if all items were sold from the top down is a variable which we might now start calling hL for “height of liquidation”, where liquidation is the event of an item having all its investments sold instantly (which would be done in top down order).
    - If there were ever a reason to “liquidate” an item in this sense, there might be value in providing users with the “liquidation value” of their investments
    - One context in which liquidation might be applicable is when an item is deleted
    - Another one could be when a decision has been made via investment; instead of everyone scrambling to sell their investments once the decision item is superfluous, the system automatically “liquidates” the item so everyone gets their fair share back.
* The logical extension of a rule that limits the maximum amount you can invest in some items to the reputation you have in the relevant domain, is to apply this rule to all investment
  + I think this is actually a very powerful idea
  + This would mean that your influence would be fundamentally dependant on your reputation in all contexts, whereas before it was possible for a wealthy user to significantly sway the balance for the worth of an item
  + Reputation scales as the square root of the total worth of items you own in that domain
  + I think I’ll ban investing in items you own even though you could still do it indirectly through another user you control
  + One of the main benefits of this rule will be to force influence to be further distributed over more people
  + I say over more people, not just over more users, because I think it will be very difficult for one person, or even one entity like a company, to control a very large numbers of independent users with enough collective reputation and worth to substantially influence the worth of a given item.
  + It would be possible for a very wealthy user to divide his resources and artificially create an army of low reputation, low wealth users.
    - This would be done by investing in items owned by users you control
    - By dividing his wealth in this way, the powerful user could gain more influence over any given item than he otherwise would have
    - He would also be able to have that army of user invest back in items he owns and so gain in reputation (and salary) himself
    - Such behaviour would be undesirable so it is important that there be a mechanism to prevent it
  + The problem the powerful user has is to make his investments in all of those low level users’ items convincing
  + If the investments are not convincing in the sense that it is not obvious that the items deserve the investment, then there is an opportunity for cheater hunters to invest negatively in those items and benefit from being the one to catch the cheater after he entices other users to “down vote” the cheating user as well. It shouldn’t be hard to find people with a desire to punish cheaters
  + For the investments to be convincing, the entity owning the powerful user must be able to produce large numbers of unique items of value
    - I think this will be exceedingly difficult because one person can only have value to offer in so many areas
    - It is easier to have a great deal to offer in a specific area such that you receive immense praise for that narrow area of contribution; it is much harder to have value to offer in a very large number of areas
    - I think you would need to make all the contributions relatively distinct from each other since it would probably be relatively easy to identify the connection between many users writing in a similar style about a similar subject and possibly all receiving investment from the same powerful user
    - To put the difficult more simply: it would just be difficult to avoid repeating yourself since there’s only so many subjects one person can talk about with authority (which I define as having value to add to a subject)
  + The difficulties faced by the cheater would be compounded if he wanted to gain control over an item in a specific domain
    - He would need to produce large numbers of users with high reputation in that domain
    - This would be far more difficult than producing a single user with very high reputation in a domain because reputation builds on itself
    - It is not possible to use your expertise in an area multiple times for different users because once you’ve said something of value, people won’t give another user credit for repeating it
  + Thus the most efficient way to acquire a large amount of reputation would be to concentrate your resources in a single user so that you can allow your reputation to build on itself
  + Normally this effect of reputation building on itself leads to a very small number of people being granted disproportionate authority in their fields
  + I’m going to temper this effect by making reputation scale as the square root of total attention received (in the form of investment in your items)
* Every action in the site should have an associated field specifying the reputation required to do it
  + E.g. every field of an item has an associated field specifying the reputation required to edit it
  + All of these reputation requirements need to be domain specific
  + This could mean that there will need to be two permission values associated with every editable field (and other action); one to specify the domain of reputation required and the other to specify the amount
  + This hopefully won’t be necessary, though. It would be better for the domain to be inferred from the domain of the item (in the case of fields of an item)
  + Certain fields of an item should probably always be global domain, though
    - Perhaps specifying which domains an item is in should be editable by anyone with enough reputation in the global domain
  + I don’t think there actually are any actions that don’t fall under the category of editing an item’s fields
  + Creating an association between items requires the necessary privileges on all the associated items, all of which are specified within those items as permissions on editing its fields
* User identification stuff
  + I think every method of ID a user uses should be public information
    - This means that, for example, the username of a user’s associated PayPal account will be visible to everyone
    - I think it is worth preparing people for such information exposure early, since once Zutha is distributed, all underlying information will be public
    - I don’t think it will be practical to maintain secret information about every user as a distributed secret
    - Anyway, it is likely that there will be a gap between making Zutha an open distributed system and handing over financial control (from me) to the distributed cloud – mainly because it will take a long time to develop and test the robustness of such a distributed secret system (not to mention the reliability of the system in behaving correctly and hopefully wisely in using the financial control that distributed secret grants it).
  + For maximum security, a user may attach many 3rd party identification services to his account
    - This could include the kinds of OpenID accounts existing at the moment like Gmail and Yahoo etc.
    - There are also likely to be (or perhaps already are) services that use advanced technology for identification, such as biometric analysis and evolving encryption key devices.
    - Whatever the means of identifying yourself to the 3rd party identification service, that service then needs to pass on the identification to Zutha securely
    - One way this could be done would be for the 3rd party to simply to confirm to Zutha the current public key being used by the user
    - An example
      * User X has a publically visible id, X1, with 3rd party identification service Y.
      * User X’s client generates in real time a public and private key pair (P,S)
      * X uses whatever means Y requires for identification to create a secure link with Y
      * X shares his public key P with Y
      * Y creates a secure link with Zutha using Y’s well known public key
      * Y tells Zutha that X1 is currently using public key P
      * Zutha looks up 3rd party username X1 and discovers it is associated with Zutha user X
      * Zutha can now receive messages that it knows came from X
      * The messages from X encrypted with X’s public key P are distributed openly across the Zutha network and everyone can decrypt the messages with X’s known public key, but no one can construct new messages that pretend to be created by X because they do not have X’s private key S
    - For all I know, this is how OpenID currently works
* Worth bars should be hidden if they are empty
  + This will make pages of items of mostly zero worth more compact
  + A good example would be an Events page, where almost all items would have zero worth in all 4 measures, yet the presence of the 4 empty bars would roughly double the height of each item on the page.
  + I’m intending to reduce the height taken up by the worth bars from the current layout though.
* When an item is associated with a place X as being “inside-of” that place, it should be at the same time given an inside-of association with all items X is inside-of
  + Example: the “1 Shelley Street Building” would have an “inside-of” relationship with “Shelley Street, Sydney”. At the moment this association is created, another association should be created between “1 Shelley Street Building” and “Sydney” making “1 Shelley Street Building” considered inside-of Sydney. The same should happen with NSW, Australia, Southern Hemisphere, Australasia, Earth etc.
  + This work is not absolutely necessary since I think the Topic Map system is capable of defining recursive associations such that Sydney being inside NSW and NSW being inside Australia implies that Sydney is inside of Australia, without a 3rd association being necessary directly between Sydney and Australia
  + I want to create that 3rd , strictly speaking redundant association anyway, though, for efficiency reasons
  + This is in keeping with the philosophy of doing as much work as possible at write time
  + By doing some extra work at write time and taking up some extra storage space, the time to query all items that are part of a large place (like a country) for example, will be much faster
  + Even if write time is increased dramatically by all the work I am assigning to be done then, that work only has to be done once, yet the benefits to read time are gained on every read and there will be far more reads than writes – certainly in this case, and probably in most other situations where read and write time can be traded off against each other

# Wed 22-Dec-10

* How to automatically reward hosts for their contribution of computing power
  + Hosts should be rewarded on their conformance to the consensus of the cloud, the amount of traffic they handle, and the speed with which they handle traffic
  + Whenever a client makes a request, the content of the request, combined with the current time, should be hashed and the hash used to determine which hosts it should request data from
  + All the hosts it sends the request to will then also know which other hosts were sent the request
  + It will be up to the client to report inconsistent data
  + If a minority of hosts return anomalous responses, the client should send the anomalous responses to other hosts (they don’t necessarily need to be the same ones who dealt with the initial request – in fact this complaint request is an action request and should ultimately be distributed to all hosts)
  + The anomalous response (packed with the request) will be digitally signed by the host that sent it – at least, hashes of both request and response will be digitally signed – so an anomalous response contains within it proof of a host giving an incorrect response
  + When a client distributes this evidence of a host misbehaving, all other hosts should update their database to slightly reduce the misbehaving host’s reputation
  + There should be specified rules about how much certain mistakes should cost a host in reputation
  + The loss of reputation will be achieved by Zutha investing negative zuth in the host’s item
  + Provided a client does not complain about a host’s response, all other hosts that were sent the client’s request will assume that that host responded correctly and assign it its due reward
  + The reward for each single correct response would be exceedingly small, but would add up slowly over time
  + The set of hosts that are sent the user’s request then report to all other hosts which of them responded correctly and all other hosts update their databases to add the small reward to each correctly responding host
  + I haven’t yet thought of a good way to keep track of how fast hosts respond
    - Since the time delay in receiving a response is probably more likely going to be due to network delay which is not under the control of any one host, it is perhaps not fair to rate hosts on their response times
    - I also can’t see a way to prevent clients lying about when they receive responses
    - I won’t try to rate hosts on response times then
  + Hosts should be rated on uptime
    - If a user sends a request to a host to create an item, and the request times out, the user should immediately report this to other hosts
    - As in other contexts, it is important that the hosts the client reports to here not be chosen by the client, but be determined cryptographically by the message and the current time (to low resolution such as minutes or maybe seconds)
    - The hosts that receive a report that another host didn’t respond, should check their own ability to receive responses from it
    - If none of the hosts can get a response from the not-responding host, the host should be considered “down” and should be punished for it
    - a host should be able to report that it is going down for a time and not be punished directly, though it would sacrifice some reputation by doing so
    - when a host reports that it is shutting down, the rest of the network will take it off the list of active hosts, and soon after clients should stop asking it for requests
* How to determine the host that is the creator of an item
  + I’ve wanted to avoid ever having just one node responsible for an action, but I realised that it doesn’t undermine the robustness of the distributed system by having clients pick a single active host to be responsible for creating their item
  + If the host doesn’t respond to the request, the client reports it inactive so that it will be taken off the list of hosts to choose from, then picks a new host to create the item
  + Discovering that a host is down like this would be an inconvenience, but would likely only cost the end user a few seconds – nothing compared to the cost of a major company server going down with most present IT systems
  + So a client should use a predetermined algorithm that uses the current time and a hash of the message to be sent, to pick a host from among the currently active hosts to be the creator of the item
  + The chosen host will then be responsible for giving the new item an ID and a timestamp
* A self-organizing network
  + There will be a lot of occasions where somehow every single host needs to receive new information
  + It would be wasteful for every host currently with the information to arbitrarily send it to other hosts, since many hosts would receive the same information many times from other hosts that didn’t know they already have it
  + I think a useful approach here would be a dynamically evolving probabilistic system whereby host A actively estimates in real time which other hosts are most likely not to already have a piece of information held by A
  + Whenever A receives information that needs to be distributed to the whole cloud (which is pretty well every message received except view requests) it should send out the information to a random set of other hosts, but a random set determined by a non-uniform distribution
    - When hosts reply to A’s message they will tell A whether they had the information already
    - If host B already had the information A sent to it, A should decrease its weighting in its probability distribution such that next time A has a message to send it is less likely to send it to B again
    - Conversely, if host B did not have the information, A should increase B’s weighting in its probability distribution
  + In this way, the system as a whole will dynamically create an approximation of a minimum spanning tree across the whole graph. It won’t actually ever be a minimum spanning tree, but the resultant amount of traffic will be far less to cover all hosts than would be the case if every host simply sent messages to other hosts at random
  + I still haven’t dealt with how to ensure that every host does in fact receive the message
    - I don’t think it is necessary for every message to reach every single currently active host – after all, there will be new hosts joining regularly which don’t have the latest information so there is never going to be a situation where all hosts have all information
    - It would be sufficient for the vast majority of hosts to receive each new message
    - The probabilistic approach is capable of achieving this goal reliably I think
  + What will stop a new message from continuing to be distributed indefinitely?
    - After a while the message will reach saturation point
    - Take a host who receives a message when most other hosts have already received it
      * He will send out messages to his own set of randomly selected hosts
      * Probably all of those hosts, though, will have already received the message and will not send the message on any further
      * The propagation of the message thus ends here
    - It could make sense for the probabilistic host picking algorithm to choose less hosts the longer it has been since the message was created
      * I’m not sure this would help all that much though, since the main way the propagation should be halted is by saturation, which is precisely the condition under which I want the propagation to stop
      * If there happens to be very large network delay on an occasion, too few hosts may receive a message because the number of messages sent out by each host too quickly dropped below a critical amount necessary for saturation to be reached before the message dies
* How to deal with the fact that some hosts won’t have the latest data
  + Actually most hosts are likely to be missing a non-trivial amount of data at any given time
  + It would be important to distinguish between a host returning an incorrect response because it doesn’t have the latest data, and returning an incorrect response despite having the latest data
  + This can be done I think, because every host will know how recently each piece of information was updated, so it can report how up to date its relevant data is in its response
  + Let’s examine a scenario in which user X asks to modify the name of item Y
    - Some time ago, the reputation required to modify item Y was less than user X’s current reputation
    - Item A did not receive (or has not yet received) the update to the reputation requirement on modifying item Y’s name
    - All other items user X queries has the latest information
    - Item A’s response to X’s request should include information about how up to date all the relevant information is in A’s database
    - The relevant information in this case would include the current name of the item to be modified and the reputation required to modify that name, and the editing user’s current reputation
    - A could communicate how up to date it is by sending the IDs of the latest event items associated with each of the critical pieces of data
    - X will receive an anomalous response from A and will be able to discover that A is not up to date because event items are sent from the other hosts which are more recent than the ones sent by A
    - X’s client should now notify the network (which will include A) that A is not up to date, and A can respond be requesting the data it is missing
    - The problem is how to ensure X’s client does this
    - X might want his request to succeed even though he no longer has sufficient privileges, and may modify his client so that A is not notified of not being up to date and will therefore accept X’s change request
  + A host should manage the whole of a user’s edit request
    - I’ve been assuming X’s client is responsible for notifying other hosts of anomalies
    - I don’t think it is a good policy to expect anything from clients since they are not bound by any reputation (and the fear of its loss) as hosts are
    - Actually I’ve already thought of the structure that avoids putting any responsibility on the client: making a host responsible for carrying out a user’s request
  + Let’s re-examine the scenario with a random host managing the request
    - User X sends a request R to the host specified by the hash of the message and the current time. Call this host M for Master Host
    - M is, as far as X is concerned, a random host, so it is impractical for X to arrange to have M cheat in X’s favour
    - M will check with multiple randomly selected other hosts to make sure its relevant data is up to date
    - Once M is relatively confident of being up to date, it processes the request, performs it in its own database, and then sends the request out to random hosts to be propagated across the network
    - If it is later is discovered that a change G was made before request R that wasn’t known by any of the hosts M contacted, what should happen?
      * This could happen if the host responsible for managing event G was very isolated in terms of network delay and was slower to propagate its change across the network than M was with R
    - If there was some way to reliably establish that change G is genuine, then the request enacted by M should be recalled
    - Other hosts that received change G before request R will not accept R so it is essential for the inconsistency to be rectified
    - There should be a limit to how much delay is allowed on the part of a host before its changes are not accepted – at least with the timestamp asserted by the slow host
    - Should a user be punished for unluckily getting a slow host and having his change requests delayed substantially?
    - I don’t think there’s any way around that – users that happen to be assigned slow hosts (either in terms of processing speed or network isolation) are just unlucky
      * Since slow hosts should quickly be eliminated from the system and there will be a great deal of hosts, it will an extremely unlucky event indeed
    - It is better for the slow host to negatively impact a single user than everyone on the network, though
      * It would impact many users if, as in the situation we are trying to resolve, a delayed change G is going to force a change R to be rolled back.
      * It could potentially cause many changes to be rolled back
    - The impact of the slow host should be kept to just a few unfortunate users by being taken off the list of active hosts as soon as possible once its performance is identified as being unacceptably low
    - If a host receives change G substantially after its creation timestamp it may challenge the slow host A to identify a minimum number of hosts that received its message G within a maximum time after the message creation timestamp
      * When a host sends a message it should be required to check that a minimum number of hosts receive it within a maximum period of time
      * If it hasn’t received enough confirmations back by the time this maximum time is reached, it should recall the message
    - If A fails to prove that the required number of hosts received its message within the maximum time, then it should be severely punished, if not immediately blacklisted for disobeying that crucial rule
    - If hosts receiving A’s message directly get it too long after it was created, they should confer with the other initial receivers of the message to make sure enough of them received the message quickly, before sending out their share
      * Initial receivers of a new edit request will be aware of this fact, and aware of the other initial receivers, because they can perform the host allocation algorithm on the message using the creation timestamp as time input, to calculate the same set of hosts calculated by A as the initial hosts to propagate the message to
    - To ensure A contacted all the hosts it was supposed to, the initial receivers of its message should randomly contact the other initial members (as determined by the algorithm on the hash of the message) to make sure they received it also
      * Perhaps every initial host should contact every other and verify all the requirements
      * This seems wasteful, but it will be done in parallel so each initial host will only have to respond to additional requests equal to the number of initial hosts, which will be insignificant compared to the volume of requests a web host should be expected to handle
      * They will not have to do this extra work for every user request since they will only be selected as initial hosts for a very small proportion of requests
    - As soon as it is discovered that A broke the rules and lead to a message being propagated too long after it was created, the message should be recalled and A severely punished or blacklisted for misconduct
    - The initiator of the change request G could at this point be notified of its failure
      * but it hasn’t really failed
      * The only thing that should be rectified about G is its timestamp
      * It is important that the timestamp not be too much earlier than when the majority of hosts receive the message
    - The algorithm for determining the master host of a request could have a function to be repeated to determine a new host
      * The problem is that this function needs to be dependent on time (or better a time dependent variable not known in advance) to prevent manipulation of the chosen host.
      * Without a central host, how can an exact time be picked to run the master host allocation algorithm again to determine a new master host
      * Once the initial master host is taken out of the picture, there is no way I can think of to return control of the request to a single host so that it can re-determine its timestamp (and probably new id)
    - I think it will be necessary at this point for control to be returned to the client who created the request, since it is the only remaining entity that can be singled out in the life cycle of this message
      * The client will probably just notify the user that the first attempt at the request failed, and that it is trying again
      * More details should be available for an interested user
      * An advanced user may want to rent his wrath at the delay at the host responsible
  + It makes a lot of sense to offload work and responsibility from the client for write requests
    - A client will want to be able to keep operating while write requests are being processed by the network
    - A user may have sent a request a while back, and sometime later, while he is doing something else, will receive a notification that it failed
    - This is better than the client having to do lots of work to send multiple sets of messages to multiple hosts for each write request as this will likely slow down the user’s experience dramatically while waiting for the write request to process
  + I haven’t yet dealt with a situation in which a host receives what turns out to be a valid earlier request that conflicts with a later one it has already received
    - Let’s return to the scenario with a request R from user X to modify item Y
      * The request is managed by master host M
      * host A has already sent out change request G, which modified item Y to increase the reputation required to modify its name
      * this time we will assume that A sent out request G within the required timeframe and has not breached any rules
    - it is possible that M asks for the latest updates about item A from many hosts, but none of them have yet received G
    - if G is created only a split second before R then this is quite possible
    - for a time there will be some hosts receiving G first and some hosts receiving R first, then some hosts will receive the other request and discover the conflict
    - hosts that receive G after R should undo action R
    - hosts that receive R after G should not perform R because it is invalid
    - eventually M will receive G and know that request R was invalid
    - M should be responsible for notifying user X that its request R failed
    - I don’t think there would be any reason for any hosts to notify M immediately when they discover that R will fail
    - M will receive the conflicting change action G in due course, and will only be flooded with duplicate requests if extra hosts tell it about G
    - Mind you, it is possible, though unlikely, that M will happen to not receive G if the propagation is done in the standard probabilistic way
    - Perhaps every host, when it discovers the conflict with M’s item, should slightly increase its probability of including M in its list of recipients of message G
    - This will mean M will probably receive a lot of redundant messages, but the extra probability could be low enough that M is unlikely to get overwhelmed by requests from a huge number of hosts at once, while still being extremely likely to receive at least one so that it can notify user X of the failure of his request

# Thu 23-Dec-10

* How should the authenticity of results be checked for a request about the current value of an item?
  + Investments are likely to be the most volatile type of data, since they will be frequent and will have regular opposing effects on the value of an item
  + There should be a buffer period during which an item’s current value is not required to be consistent across hosts
  + At a given time T, an item’s value is expected to be consistent for all operations with timestamps less than or equal to P, where P<T
  + T-P should be a period of time during which it can be reasonably expected that all hosts will have received a message sent at time P
  + When a user requests the value of an item at time T, the most recent value which is consistent across all asked hosts should be returned
  + To facilitate discovering this most recent consistent value quickly, all hosts contacted should return a range of item values matched against timestamps between P and T
  + The set of timestamps chosen would be the timestamps of the actual investment events between P and T
    - Since the timestamps of investment events will be the same on every host provided each host has received that event, there will be a high chance of timestamps overlapping between hosts
  + The client should report to the user the most recent valuation of the item which is consistent across the majority of hosts
    - The user would be allowed to determine the percentage of hosts that are required to agree for a result to be valid
    - This way the user can trade off accuracy against currentness
  + If a host’s response is not consistent with the rest at timestamp P, then it should be considered to be in error (whether deliberately or accidently) and should be punished accordingly
    - The host would more importantly be notified of its anomalous result so that it can fix its database
  + Why not just tell the client the value of the item at time P, where consistency is guaranteed?
    - P might be a noticeable time, for a human, before the present, perhaps even in the order of 30 seconds or more
    - This would not be a problem for most users, but some users would want to do the Zutha equivalent of high volume trading
    - I don’t want to encourage high volume trading as it won’t add any value to the system
    - However, I don’t want to give an unfair advantage to users willing to go to the trouble of getting the latest information
    - Even if there is a rule that hosts don’t give up information about item value more recent than P, someone who controls a lot of hosts would be able to get more recent data than at P, even if it is not 100% certain data
    - Therefore someone with a lot of resources could get an unfair advantage over the average user by getting hold of valuation data that is more recent and still quite reliably correct
    - If everyone receives the most up to date information that it is possible to get at whatever level of reliability they choose (trading off promptness for reliability) then there is no way for users with resources to get better information than that
    - The system described above will give the most recent reliable results obtainable (on average) because it gets the very latest information from a random selection of hosts and a random selection of hosts is as good as any arbitrary selection of the same size because there is no way to predict which hosts will have the latest information at any given time
      * Random variation will still cause some users to get more recent results than others, but there would be no way to get better results by deliberate means
      * If you received information from a lot more hosts than the normal amount contacted for a request, then you could probabilistically increase the accuracy of your results
      * Perhaps clients should be allowed to increase the number of hosts they contact per request to a very large number (more than could realistically be controlled by a single entity) if they choose
      * Most users would have no need to contact a huge number of hosts in a single request so wouldn’t bother
      * But the advantage is still taken away from someone with a lot of resources, because any user can do just as well as anything they can do, just by changing a setting
  + Hosts will have to do a lot of reprocessing of the latest investments
    - When an investment event A is received by host X, the investment is given an h value based on the current value of the item in that host’s database
    - If investment event B is received later, which has an earlier timestamp than A, host X must use the item value at the time of B’s timestamp, to calculate B’s h value. X must then recalculate A’s h value based on the new value of the item at A’s timestamp
  + The lack of extremely up-to-date information about the value of an item will prevent High Volume Trading, which relies on being extremely quick to respond to very short-lived patterns existing at an extremely short time resolution
    - This is a good thing, as I think High Volume Trading would have a very negative effect on the Zutha economy (as it does in the real economy incidentally)
    - Generally speaking, the inability of users to get accurate data in the very recent past will make it pointless to try to make zuth based on patterns at a very small time resolution
    - It would be best if data cannot be trusted for at least quite a few seconds before the present
    - The longer the better
    - I want investors to make decisions based on long term trends and the actual perceived value of the item, not write bots to look for meaningless short-lived patterns in investment behaviour
* An unpredictable, dynamic broadcasted key for producing statistically random results that are consistent across all hosts
  + Such a key would need to be generated by some trusted 3rd party
  + It would be important that the 3rd party itself not be able to predict what the next key will be
  + I’ve thought about whether some kind of uncontrollable and unpredictable, but universally readable signal could be used for this purpose, such as cosmic background radiation
    - The risk here is error in reading the signal such that different hosts get different results
    - Provided the same signal can be read by the vast majority of hosts, the incorrect hosts can be corrected by the majority
  + The key would change at known time intervals
  + The next key would be available to each host sufficiently in advance of it becoming active, that all hosts will have it by the time they need to start using it
  + This would mean that at any given time, only 2 or maybe 3 keys that are ever going to be valid, would be known to anyone
  + This key would be combined with the hash of a message to determine such things as which host will be the master host for creating a message and which hosts that master host will initially send the message out to
  + The benefit of the unpredictable changing key is that it would be impossible to predict long in advance the result of a host selection algorithm for a given message
    - This stops anyone knowing which host will be the master host for a message long enough in advance to take control of that host
  + Why can’t the current time be used as the changing key?
    - You could craft a message such that, at a known time in the foreseeable future, you will know which master host will be chosen to handle it
    - You could then work to compromise that host in the intervening time
    - The fact that the set of hosts would change unpredictably would substantially limit that predictability
    - To be safe, though, I think it better if you can’t know the result of a host selection algorithm for a given message more than a couple of time intervals in advance, which might be in the order of one minute
  + It still remains to ensure that you can’t craft a message for the current (or next) dynamic key, that will be sent to a host of your choosing
    - It would be extremely difficult to specify a specific desired result of the algorithm and craft the message to produce that result
    - A strong cryptographic hash function is designed to be impractical to reverse engineer, and since the attacker only has about a minute to do so, the difficulty is even more extreme
    - The more realistic approach for an attacker would be to generate huge numbers of messages that vary in ways that don’t matter to the attacker, check which master host would be selected for each message, and then send the first message that yields one of many acceptable results (to the attacker)
    - Statistically, the attacker should only have to generate a similar number of messages as there are hosts to get one that will choose the host he desires
      * This is assuming that any set of messages, regardless of in which particular ways they differ, will result in a uniformly random distribution of hashes, and that given a uniformly random set of hashes, the algorithmic selection is uniformly random in selecting any host
    - One way to hinder the attacker’s ability to craft many messages that don’t differ in ways important to him, is to disallow certain kinds of data in messages that look like such superfluous padding
      * It shouldn’t be hard to require that every part of a message have an effect
      * An attacker could still write a request to modify the description field of an item and fill it with (partly) gibberish, which he can vary arbitrarily to generate a large range of hashes, without affecting the action he actually wants to perform
      * Messages that contain gibberish should be disallowed, for a start
        + This would be worth doing anyway to prevent petty vandalism
        + I think it would be hard to tell the difference between text written in very poor English, and well-crafted gibberish comprised of lots of words often used by people who are bad at English
      * The attacker could always randomly select from a large range of valid content
      * It wouldn’t really be a problem if an attacker could have a request to modify an item’s description be controlled by hosts of his choosing
      * It might be more concerning if an attacker could have hosts of his choosing control a request to modify permissions, for example
      * But a request to modify a permission would be very simple and wouldn’t provide scope to add filler for the purpose of manipulating the master host selection
        + A request to modify a permission would only specify the permission in question and the new value
        + I guess the permission value (reputation requirement) could be varied among a large range of acceptable values (for the attacker)
        + Perhaps reputation requirements should be bound to a discrete (though unbounded) set of possibilities e.g. {10,50,100,500,1000,5000,10000, …}
        + This would prevent fine variations in the number for the purpose of manipulating the chosen host
      * An attempt should be made to limit the ability of significant requests to be varied greatly without much altering the core effect of the request
    - The ability of the attacker to generate enough messages to find the result he desires can be reduced by minimising the window in which an unpredictable (dynamic) key is known
      * If the dynamic key was generated independently by each host (or maybe just many hosts), perhaps by approximating the data read from the cosmic background radiation over a specific time interval, then there wouldn’t need to be as long a wait to be confident every host has the next key
      * If absolutely every host is required to measure the CBR signal independently, then there would be no network delay involved in ensuring every host has the next key
      * The only factor in determining the minimum key lifetime would then be the time taken to read and process an arbitrarily small duration of CBR signal
      * An attacker would only have this small window (the key lifetime) to generate all the host selection combinations enabled by that key, and send off the message that gets him what he wants
      * Actually, I just realised the creator of the initial message can select a creation timestamp for the message anywhere within a period of time that is a realistic delay between himself and the master host
        + If he generates a message using a timestamp (and corresponding dynamic key) from too long ago, even if this message/key combination chooses a compromised master host, the master host will not be trusted if the creation time of the message is too long ago
        + There will be 2 creation times sent in the message: the client creation time and the master host creation time
        + If either creation time is too long ago, other hosts will reject the message
        + The master host creation time will be the official one that decides the item timestamp; the client creation time will be tacked on purely to check for this exploit – that is, making a message with an ancient timestamp that will cause a master host that you control to be chosen, which you can then command to assign a current timestamp
        + I’m assuming in all of this that it is possible using cryptography to prevent a compromised master host or other agent in the chain to modify crucial data
      * Therefore there probably isn’t any point making the dynamic key lifetime smaller than the time taken for every host to receive a message (perhaps in the order of 30 seconds, though hopefully much less)
    - The ability of the attacker to generate the particular result he wants could be reduced by making the result of the host selection algorithm non-uniform in various ways
      * For a given broad category of messages, only a subset of hosts might be candidates for selection
      * The particular subset of hosts associated with a particular category of messages would vary based on the dynamic key so that the same hosts aren’t always processing the same kinds of messages
      * This would mean that the kinds of changes available to the attacker to craft his message (the ones that don’t significantly change the message) would only vary the result of the host selection algorithm among a subset of hosts
      * The attacker could still wait for many periods, trying many variations of his message for each one, until the desired result is obtained
      * For choosing a particular master host, this approach would eventually work, even if it took many periods
      * If they attacker wanted to select the set of initial hosts the master host sends the message on to, this would be impractical
        + The number of combinations of initial hosts will be many orders of magnitude greater than the number of combinations of the master host selection
        + To pick a particular set of initial hosts, the attacker would have to wait a number of periods of a similar order to the number of combinations of initial hosts, which could easily be made to require centuries or even trillions of years if the number of hosts and the number of initial hosts was large enough
        + If there are 1000 hosts and 10 initial hosts are selected, there are 1000 nCr 10 ~= 10^23 possible selections of initial hosts
        + If every time period allowed only 10^6 of these combinations for any given kind of request, then it would take 10^17 time periods to choose a specific set of hosts
        + If you own 50 hosts, then 50 nCr 10 ~= 10^10 initial host combinations are acceptable so it will only take 10^7 time periods to get one of them
        + There are about 1,000,000 thirty second time intervals in a year so it would take about 10 years to for an acceptable combination of initial hosts to be selected
        + Increase the total number of hosts to 10,000 and you’re looking at a serious wait to achieve a single slightly controllable action
    - I don’t think it will be practical to stop the master host from being selectable, but the master host should not be trusted anyway, so this shouldn’t be much of a problem
* The list of active hosts should be updated at discrete pre-specified time intervals
  + It will be crucial that every host associates the same list of active hosts with a given timestamp
  + Otherwise different hosts would calculate a different master host (and initial hosts) for a given message
  + When a new host X wants to join, it sends out a request to join the network which should be propagated to every host
  + Like any other request, the request to join the network should have master host selected which determines its official timestamp
  + Let’s say the join request is given timestamp S, the time between active host list updates (perhaps the same as the time period of the dynamic key) is P, and the time that the new host X is added to the list of active hosts is T.
  + At time S+P, every host should have received the join request
  + T should be defined as the next host list update timeslot after S+P
  + From time T onwards, every host should consider host X part of the set of active hosts
  + Between T and T+P, a host may still receive a request that has a timestamp before T. In this situation it should use the active hosts list for the timeslot T-P to T, which will not include host X
* There could be difficulties in getting banks to accept a distributed, non-human entity (namely Zutha) as the owner of an account
  + A bank is likely to expect Zutha to not only have the password to its net banking account, but have a residential address and formal human identity
  + At least early on, I could be the official owner of Zutha’s accounts, but just let Zutha control the flow of capital
  + It is very common for computers to control what is done with money, but there will always be some person who can be held accountable for what it does – and can therefore be pressured into taking control back off the computer
  + I don’t want it to be possible to take control of Zutha’s funds via the human who officially owns its accounts, especially if that human is me
  + Perhaps a future with ubiquitous anonymous electronic cash would remove this dilemma
* eCash
  + If all payments to Zutha were paid in the form of electronic cash, which would just be cryptographic numbers, then Zutha would not need to have a bank account; it would just need to keep all those numbers secure
  + It is likely to be very difficult to keep enormous quantities of numbers secret and in the exclusive control of the cloud.
  + The ability to keep all those numbers in exclusive control of the cloud would rely on them being encrypted with a public key for which the corresponding private key can never be known by any individual, but only by the cloud as a whole
  + It would probably be possible for an eCash identifier to be encrypted with a public key in such a way that to decrypt the number, multiple different private keys would be required, which are each held by different hosts
  + How would the eCash identifier be prevented from ever being knowable by any of the individual hosts though?
  + If the encrypted identifier needed to be decrypted serially by a set of hosts, one after the other, then the last one in the chain would know the identifier itself, and would be able to spend it how it wished if it chose
  + The only solution I can think of is for the bank that issued the eCash to be the one to reconstruct the identifier from multiple pieces, each sent from different Zutha hosts
  + Why am I considering the bank more trustworthy than any of my hosts?
  + Zutha is designed to operate using reputational incentives
  + Why can’t I rely on reputational incentives to trust high-reputation hosts with managing some of Zutha’s finances each
  + If a host ever disobeyed Zutha’s command with a particular piece of eCash it would be blacklisted and lose all the benefits of the reputation it worked so hard to get
  + A host would only have access to a single small amount of currency for a short amount of time
    - I’m referring to the host at the end of a chain that has just decrypted an eCash identifier
    - At the point that this host decrypts the identifier, it may choose to either obey Zutha or not
    - If it disobeys Zutha and keeps the money for itself, it will gain that small amount of money and lose all its privileges as a Zutha host
    - If it obeys Zutha, it will send the money to someone else and immediately lose control of it
  + This is perfect because a host can never accumulate eCash and defect with all of it at once; it only has the capacity to defect with the small amounts it is ever granted control over at any one time
  + There would need to be redundancy in possible paths that can be taken to decrypt a particular identifier
    - If a host defects, or simply leaves the network or is unavailable at a particular time, it needs to still be possible to decrypt every identifier
  + If, for every identifier, it was publicly known which hosts have the information required for each stage of the process of decrypting that identifier, then it would be undesirably easy to compromise a sufficient set of those hosts to access the information required to decrypt a given identifier
  + Identifiers could be constantly sent on a (metaphorical) journey of being partially encrypted and decrypted by various hosts, until they are ready to be used, at which point they will be fully decrypted by (metaphorically) retracing the steps of their journey
  + The initial encryption of the identifier which is done by the paying client, should still require multiple stages to decrypt
  + From that point onward, though, the identifier will be encrypted with many more keys, each of which will be distributed among only a few hosts each
  + In this way, the security of the identifier is maintained by the fact that the cooperation of a large number of effectively randomly selected hosts is required to disobey Zutha
  + It would be impossible to predict which hosts need to be compromised to obtain all the information necessary to decrypt an eCash identifier because at every time step, an unpredictable new host is added to that list
  + I just realised that this “sending the identifier on a journey of encryption” idea won’t work
    - The initial encrypted identifier will still be available so the only encryption keys required are those initial ones
  + The eCash identifier could be regularly exchanged for a new one via the issuing bank
  + Each time an identifier is sent to the bank to be re-issued, the bank will be given a new public key with which to encrypt its reply
    - Each of these public keys will require a different set of private keys (each held by a small group of hosts) for decryption
  + How to make an asymmetric encryption algorithm with one public key and multiple private keys?
    - In order for the private keys to be truly independent private keys, the agent making the public key would have to be ignorant of them
    - Let’s define our variables
      * P = singular public key
      * S1, S2, S3 = private keys owned by agents A1, A2, A3
      * P1, P2, P3 = public keys of agents A1, A2, A3
      * B = the agent who creates P
    - Agent B needs to be able to use P1, P2 and P3, to create a public key P, which requires private keys S1, S2, S3 to reverse
    - I’m not sure if this is mathematically possible, but if it is, it would be very powerful
    - What would be even better is if B can encrypt data (without knowing S1, S2, S3) that can only be decrypted by P
      * B might have an additional private key S4, that was used to generate P from P1, P2, P3
      * B can encrypt data that can be decrypted only with P, which enables him to prove his ownership of P
      * Data encrypted with P, though, requires B’s private key S4, as well as 3 private keys he doesn’t know
      * B can identify himself to a 3rd party as the owner of one of the private keys associated with P
      * The 3rd party can then be confident in encrypting its reply to B using P, knowing that B (though not only B) will be required to decrypt the reply
  + A worked example
    - Agent B has just decrypted identifier X1
    - Agent B generates a new public key/multiple private keys set (P,S1,S2,S3,S4) using public keys P1,P2,P3
    - Agent B encrypts X1 with S4 and sends it to the bank along with…
    - Wait a minute; someone has to choose the new identifier!
    - In fact, this is a problem when Zutha first receives money from a user: a new identifier has to be chosen by someone to take on the value of the identifier held by the user
  + How to select eCash identifiers so that no individual can use it without the cooperation of random others from the Zutha network
    - I think I’ll need the bank’s cooperation to make this distributed ownership of cash possible
    - What the bank would normally do
      * Someone sends the bank an identifier X, encrypted using the bank’s public key, with which they want to associate some money
      * The bank decrypts X with its private key and stores it
      * When someone wants to associate a new identifier Y with the money currently associated with X, they send the bank both X and Y, both encrypted with the bank’s public key
      * The bank decrypts both with its private key, finds the record associated with X and associates it with Y
    - I need the bank to use a different key for assigning a new identifier and checking an existing identifier
      * Let X1 be a number that is given to the bank to associate with an amount of money
      * Let X2 be a number with a special mathematical relationship to X1 such that when decrypted with another number (probably P – see above) it will yield X1
      * The bank should require X2 to be sent in order to unlock the funds associated with X1
      * Agent B should find X1 and P (using P1,P2,P3) such that decrypting X2 with P yields X1
      * B does not know X2 though
      * X2 can only be found using X1 as well as S1,S2,S3,S4
      * If cryptography can do this I will be awed
* Multiple owners
  + It would be handy to be able to assign multiple owners to an item
  + This is obviously applicable if a book has two authors, for example
  + This could be done by creating a group specifically for the purpose of assigning two authors the ownership of an item, but this would be a pain to do, and the group really doesn’t have any kind of identity in and of itself in this context
  + A group is applicable for assigning ownership to such things as committees, families, or companies, for example
  + When there is simply an ad-hoc collection of owners that don’t really have an identity as a group apart from sharing ownership of the item in question, it would be better to model this by simply assigning them all as owners of the item in a single association
* How should ownership be portioned among multiple owners or members of an owning group?
  + Like everything else, I think this should be dynamically decided by weighted consensus i.e. using investment of zuth
  + Let’s take the specific example of the book Daughter of the Empire by Raymond e. Feist and Janny Wurts
  + If the authors themselves could be contacted, I guess they could be asked to agree on a fair division of ownership that reflected how much they thought they each contributed to creating Daughter of the Empire
  + A much more difficult scenario, though, would be an open source software project
    - How would you get all the contributors to agree on a fair distribution of ownership across contributors
  + At any rate, the owners are often-times not going to be contactable, or not going to be able or willing to agree on a division of ownership
  + So what if there were two items representing the relative degree of ownership of each of Raymond E. Feist and Janny Wurts in owning Daughter of the Empire
  + Users could come along and invest in the ownership items to vote on how much they think each respective author contributed to the item
  + The percentage of ownership of each author would be given by the worth of their ownership item divided by the collective worth of all the ownership items
  + One problem I foresee is that the first person to invest in an ownership item may invest in just one, giving the corresponding owner effectively 100% ownership over the item, even if a really tiny amount of zuth was actually invested in the ownership item
  + Should there be a separate set of ownership items for each of the books in the Empire series that was co-authored by Raymond E. Feist and Janny Wurts?
    - It seems most appropriate to treat the Empire series as a whole as a single collective work by the two authors
    - It would be inappropriate, I think for the relative contributions of the two authors to be determined separately for each book
    - I think there should be a single set of ownership items to determine the relative contributions of the authors for that particular project: the Empire trilogy
    - The question is how best to model this
    - The two authors could be part of the group “Authorship of the Empire trilogy” which would capture the fact that the group is specifically for the authorship of that one project, and would not encompass other separate books written by the same two authors
* If a piece of eCash was stolen, it would be valuable to be able to figure out for sure which hosts were responsible for releasing the necessary keys to acquire that piece of eCash
  + It would still be important that no specific host be necessary to decrypt any given piece of eCash
  + It could be possible to design a cryptographic system such that, say any 5 out of a possible 10 keys are required to decrypt something
  + This way, every host could have its own private key (or maybe multiple), which it is solely responsible for
  + Whenever a piece of eCash is used, the bank that issued it should release some identifying information about it publically
  + This information should identify which 5 of the 10 available keys were used to decrypt the eCash identifier that was sent to the bank
  + I’m being very optimistic about what I can expect banks to do to make all this work
  + If I had the banks’ cooperation I might not need all the complex cryptography
  + Let’s work through a possible scenario assuming the bank will do whatever we want to help us along
    - Agent B has just decrypted an identifier and been commanded by Zutha to take ownership of the associated eCash by asking the issuing bank to issue a new identifier that will be controlled by Zutha
    - But instead of asking for a conventional single number identifier in return, B sends the bank a list of public keys (say 10) which should be associated with the piece of eCash in question
    - The bank is told to only release that piece of eCash when it has received the same instruction regarding that piece of eCash from at least 6 different entities that are verifiably the owners of one of the 10 public keys
    - When the bank receives such an instruction, it should publish publically which of the 10 public keys were used to authenticate it, and possibly what the instruction was as well
    - The instruction sent by 6 of the 10 current owners of the eCash might be: now associate this piece of eCash with this new list of 10 public keys
    - Or: now associate this piece of eCash with this single public key, which would be the public key of some 3rd party receiving money from Zutha
  + This is no longer anonymous eCash
    - There is no need for anonymity of the actions of Zutha – in fact the whole point of Zutha is that every action is public knowledge and can be censored by reputation based discipline
    - There is, however, a need to be able to make the givers and receivers of money to and from Zutha anonymous, even to the bank
* How to transition from the open, distributed method of authenticating eCash transactions, to the anonymous, standard eCash authentication system
  + User X has a piece of eCash Z which he wants to pay to Zutha
  + X’s client uses the host selection algorithm to determine a set of (say 10) hosts that it must distributively assign ownership of Z to
  + X sends its signed eCash note, Z, to the bank and asks the bank to change to the distributed method of authentication for Z
    - The bank has no way of knowing when and for whom it signed Z for, because it signed it blindly
    - The transaction from user X to Zutha is publically known, because all information in Zutha is publically known (except each host’s private key(s) ), however, nothing is known about the real world identify of user X through the eCash note Z
  + X accompanies its message with a list of 10 public keys owned by the 10 hosts it selected to transfer ownership of Z to
  + Every Zutha host is sent the message that X used to pick its set of hosts to assign ownership of Z to
  + Every Zutha host runs the host selection algorithm on X’s message to check if they are one of the chosen hosts
  + All (or perhaps just 6 chosen by another host selection algorithm) of the selected hosts now send the same message to the bank asking it to reassign ownership to another set of (say 10) hosts (identified to the bank as their public keys)
  + If and when the bank publically reports the success of this latest transaction request, Zutha will know that client X transferred the money correctly and will accept X’s payment
  + Now that Zutha own Z, it might regularly reassign ownership of it to new sets of hosts
  + Zutha now wants to send eCash note Z to another 3rd party, Y
  + The hosts that currently own Z all send a message to the bank telling it to assign ownership to Y (identified by Y’s current public key)
  + Y should then contact the bank to confirm its ownership of Z
  + Y should report back to Zutha whether it successfully confirmed ownership of Z
  + When Y wants to anonymously send the eCash to someone else, he asks the bank to transform the method of authentication for Z back to the standard method of blindly signed numbers
  + Y sends the bank a blinded message containing the number he wants to have Z be identified by, and has the bank sign it
  + Y can now send Z (identified by the number only he knows) to anyone he wants, completely anonymously
  + The only non-anonymous transactions are between users and Zutha, but they are public knowledge anyway because any communication with Zutha is public information
* One concern I have about relying on banks tailoring their services to Zutha’s needs is that it would be possible for a powerful agent (a government, say) to pressure those banks to stop providing the tailored services needed exclusively by a distributed entity like Zutha
  + If the distributed kind of authentication I desire were useful to many other powerful entities, then they would help protect the banks’ rights to provide that feature
  + The merits of distributed ownership of money have intrinsic benefits not exclusive to an application like Zutha
  + One possible application is arbitration by the consensus of multiple parties
    - Two parties in dispute may decide to use arbitration to resolve it, but are unwilling to entrust a large amount of money to a single arbitrator for fear that arbitrator will steal the money (however irrational that believe the fear may still exist)
    - So the parties grant distributed ownership of their money to, say, 5 independent arbitrators
    - They may specify that 3 of the 5 arbitrators are required to unlock the money
    - The 5 arbitrators then collectively come to an agreement about a fair resolution to the dispute
    - If all 5 cannot agree, a majority of 3 may overrule the other 2 and make a decision
    - Since only 3 are required to take control of the money they now distributively own, they can do this
    - The ability to have a dispute resolved more democratically like this might also appeal to many parties in a dispute
    - Each party could select a few arbitrators each and they will then both be confident that the result of the arbitration is not in the hands of the other party through secret dealings with a lone arbitrator
    - Resolving a dispute like this may take longer than with a single arbitrator, because of disagreement between arbitrators, but the arbitrators would have their reputation on the line as an incentive to try to resolve disputes as quickly and fairly as possible so I’m sure they would do a better job than governments currently do in tribunals and such.
    - Obviously the disputing parties are still trusting the bank in this processes, but a bank has such an enormous amount to lose by breaching that trust, and people are already so used to trusting banks with their money, that this is a trivial consideration
    - Another benefit of distributing trust across multiple arbitrators is that cheaper arbitrators with less reputation to lose can be used without as much risk
    - The ability to use cheap arbitrators would be necessary if you’re going to use like 6 of them; otherwise the cost of the arbitration would be ridiculous
    - Disputants could even use this technique to get their mates, or anyone associated with the case, to democratically resolve the dispute
* Someone could control a large set of hosts and wait until a subset of them own a particular piece of eCash
  + They could then take control of that particular piece of eCash
  + However, the hosts that were responsible for giving an unauthorized (by Zutha) command to the bank regarding that piece of eCash will by publically known and will be blacklisted
  + It should require the accumulation of a large amount of reputation (which should require a large amount of time of obedient service to Zutha) to be allowed to participate in owning eCash
  + Therefore an attacker may steal a small amount of eCash, but he then loses all the hosts he used to steal it and which he worked so hard to get into a position of such high reputation
  + Provided every eCash note is of small enough value each, it would cost more money than it would be worth to steal money from Zutha in this way
  + For this reason, it may not be necessary for Zutha to regularly transfer ownership of eCash to different sets of hosts
  + The ownership of a piece of eCash would still be transferred when it is running low on available hosts able to unlock it
  + If a host ever dropped below a minimum required level of reputation (including being fully blacklisted), then all pieces of eCash that it participates in owning would have their ownership reassigned
  + If an agent controlled a very large number of hosts, it may be able to steal quite a large number of pieces of eCash at once before Zutha can respond by taking away control of any more pieces of eCash in semi-control of the now identified corrupt hosts
  + The resources required to get a very large number of hosts into a sufficient position of high reputation in Zutha would be enormous, though, so it would more than likely not be worth it
  + High reputation hosts will be paid for their services, as well, so they have more to gain by earning money legitimately by serving Zutha, than by stealing from it once and losing all the benefits of being a high reputation Zutha host
* If eCash is used for Zutha’s finances, what will happen if there is a major dispute between two factions of hosts?
  + Let’s say that roughly half the hosts in a network decide they aren’t willing to accept a particular change to the network’s rules that was decided by the network’s users
  + This dissenting faction decides to split off from the network and start a new one
  + A bit less than half of the money owned by the network will now be owned by the dissenting faction
  + A bit less than half of the money will be owned by the original network
  + If the process of the split is gradual, such that there is a continual process of host defection for a while, then there may be a period of fighting over eCash
    - When a host defects, the original network will take away its partial ownership of all the eCash it can
    - When a large group of hosts defect at once, they will take complete control of any eCash they now collectively hold majority control over
  + Some money will now be inaccessible to both factions because the hosts that own it are roughly evenly split across factions
    - For a given piece of eCash, a majority of its owners are required to access it
    - If neither faction owns a majority of the owners of a particular piece of eCash, then neither faction controls that piece of eCash
  + The two factions may engage in negotiation to evenly distribute between them the eCash that is not under either’s control independently
  + This is a much better state of affairs than if whole bank accounts are in dispute between factions
  + This concept of having money itself (as opposed to an access key to the money) distributed in ownership across hosts is perfectly suited to the whole philosophy of Zutha, and makes for a very resilient system
* Since bank’s will know which Zutha hosts own each piece of eCash, a powerful agent like a government could force the banks to stop funding to Zutha
  + For this reason, it will be necessary to find a way for the banks to not know the identity of the owners of a piece of eCash, even their public key
  + The owners of an eCash “note” could be identified by a number which requires a sister number for a transaction authentication
  + Each owner (host) generates a new pair of sibling numbers for every eCash note it wants to partially own
  + The first number of the pair (call it A) is sent to the current owner of the eCash note, who passes it on to the bank, asking for ownership to be transferred to it
  + To unlock the eCash note again, the second half of the number pair (B) is required
  + The command from the new item owner as to what to do with its eCash note should be encoded with B such that the bank knows it is sent from the owner of number A
  + This is just a standard asymmetric key system; a private key/public key pair
  + All that is needed is for hosts to generate a new private key/public key pair for every piece of eCash they are going to partially own
  + The public key of the would-be owner (Y) is secretly shared with the current owner (X) of an eCash note, who would then secretly pass it to the bank as the new owner of the note
  + The bank thus does not know anything about the owner of Y’s public key.
  + The bank does know that X was the one to transfer the ownership of this eCash note to Y’s public key
  + An agent trying to freeze funds to Zutha would have to find and interrogate the instigator of every eCash ownership transfer transaction (agents like X), to find out who they were transferring the eCash to
    - X would have no reason to tell the interrogators, but if a few like X did tell, it could only ever deprive Zutha of those specific pieces of eCash
    - Even if every interrogation of transactors was successful, the laborious process of interrogating every single transactor would be required continuously if an agent was determined to continuously deprive Zutha of funds
    - I just realised X can be completely anonymous to the bank by using one-time public keys and multiple remailers, so it could easily be made impossible, not just impractical, for an agent to obstruct just Zutha’s money stream
    - Even if bank customers like X do not employ high level schemes to avoid identification by the bank, it would still be an extremely laborious process to track the physical owners of vast numbers of anonymous public keys
    - The bank may choose not to record the information (IP addresses etc.) that would allow it to expose its customers’ identities if put under pressure
    - Also, it is likely that once a system of ubiquitous anonymous eCash is in place, standards will be in place that make tracking the physical identities of transactors practically impossible.
    - An agent trying to stop funds to Zutha may just force the whole bank to shut down, but that would get really powerful people offside and, anyway, Zutha will have its funds distributed across many banks
  + Every transaction of Zutha users will be public information
  + It would be important for anonymity for clients to interact with the bank using a different public key than the one they use to interact with Zutha
  + For every payment by a user to Zutha, the set of hosts that will own the money is public knowledge, but the particular public keys they use to identify their ownership to the bank, will be known only by each host individually, the paying client, and the bank, though the bank won’t be able to associate the public keys with any entity
  + Only the paying client can associate every public key with its corresponding host
  + Provided the hosts and the client don’t share any of the public keys with anyone, it will not be possible to search for the presence of Zutha owned money in a bank
  + A particular client or a particular host may be malicious towards Zutha and report the public keys they know, but that will only allow a single eCash note to be identified as owned by Zutha
  + This is why it is essential for hosts to keep using new public keys, because once a Zutha host can be associated with a public key, money owned by that public key can be looked up in the bank’s system and potentially frozen or even commandeered by a powerful agent like a government.

# Tue 28-Dec-10

* Could ideas from the medieval Icelandic system of government be incorporated into a distributed system of government based on Zutha?
  + According to David Friedman in The Machinery of Freedom, Iceland used to have a decentralized form of government in which laws were enforced by private groups
  + The law itself was universal, and there was a court system similar to current civil court systems to decide all cases, including those that would be decided by a criminal court in most western countries today.
  + If a ruling demanded a perpetrator pay damages to a victim and the perpetrator does not pay, the victim may take the issue back to court and possibly be given the right to further punish the perpetrator, perhaps by killing him if he has not either paid the damages or left the country after a set period of time
  + If the victim does not have the power to enforce the punishment on the perpetrator or extract the damages payment from him, he may sell the right to do so to someone with such power
  + There is an incentive for the victim to do this because he gets some payment where he might have gotten none, and he gains a reputation for ensuring punishments against his abusers are enforced.
  + A powerful party has an incentive to purchase from a victim the right to extract damages from the perpetrator of a crime, and then use its power to extract the damages or an appropriate punishment.
    - If the powerful party is able to extract damages payment, it makes a profit
    - If it is not able to extract payment, it still has an incentive to punish the perpetrator in order to gain a reputation for accepting and justly carrying out the right and responsibility for punishing criminals
    - This is likely to allow him to purchase more rights to extract damages from future victims
  + The concept of selling the right to punish or extract damages from a lawbreaker could be useful in developing a law-enforcement framework around Zutha
  + What would be the different factions in a Zutha system?
    - It could be that separate networks are the factions that can trade rights to punish with each other, but different networks are likely to have different laws and different currencies so this would not be very viable I think
    - Different hosts, or different owners of hosts, may be the entities with which the right to extract damages are sold
  + Eventually, powerful machines of war could be owned by Zutha and controlled in a distributed way by different hosts the same way eCash would be
  + If Zutha could control such war machines in as reliable a way as it could distributively control eCash, then it would not be necessary for hosts to be petitioned individually and sold the right to enforce a punishment declared by Zutha; Zutha could simply control a law enforcement force itself using the rules decided collectively by the community
  + That is assuming, though, that machines controlled by a narrow AI agent can effectively enforce laws
  + It is going to be necessary for many circumstances, though, to have laws enforced and actions taken by independent intelligent beings (initially humans; perhaps eventually independent AI agents).
  + Factions within the Zutha community would be able to develop forces comprised of intelligent beings and machines and offer their services for profit to enforce punishments and extract damages payments.
  + If such a force was to purportedly “extract damages”, without having been granted the right to do so via the Zutha network, they will be considered to have broken Zutha law and the same process of enforcing Zutha law against them may be used recursively
  + A robotic military force in the hands of the Zutha community would be an additional aid to enforcing laws, particularly against such mercenary forces who turn against Zutha
  + A robotic force owned by Zutha could be temporarily place under the control of a human general so that the forces can be used more intelligently than a narrow AI agent would be capable of using them.
    - Since Zutha would still have ultimate control over the force, though, the Zutha community would easily be able to remove control of if from the elected human general removing their support from him and possibly selecting another in his place.
* A democratic poll
  + It would be useful to be able to create an item on the fly to be used for making some group decision
  + Often it would not be fair or applicable for some users to have more potential influence than others in deciding the outcome
  + A good example (and the context in which I thought of this) would be deciding which movie to watch on a bus trip
  + A period of time could be specified for the voting process, during which people would invest in any number of movie items they would like to watch
  + Every person on the bus should be given an equal influence in this situation since the decision is not a matter of making the “best” decision, but of choosing the movie that will satisfy the most people.
  + When selecting the domain of an item, there should be an option to select “Democratic”
  + The domain of an item determines which kinds of reputation is used for determining the investment cap for each user
  + If the Democratic domain is selected, all users are given the same cap
  + The cap now needs to be specified manually
  + A higher cap will give greater power to the rich; a lower cap will increase the degree to which each item’s value is selected democratically
  + A rich person will always be able to invest in more candidate items, but that won’t provide the more power in determining the actual winner
  + In the case of selecting a movie on a bus trip, it might be that every person on the bus is given a certain amount of zuth to invest, perhaps 2 or 3 times the investment cap for the competition. That way people with very little or no zuth can still invest in several of their favourite movies.
  + If zuth was going to be given to people it would probably be desirable for it to only be usable for this specific poll, otherwise it would be expensive for whoever gives the zuth and there would be too much of an incentive to keep the zuth for later
  + Thus, there should be a provision to create poll items which are an isolated system
    - Users who are participants would be given a fixed amount of poll-specific zuth
    - An investment cap would be chosen which is greater than or equal to that amount of zuth
    - Users could be banned from using their existing wealth to invest in this poll in order to keep it completely democratic

# Thu 30-Dec-10

* The amount of community support required to make a particular code change should depend on how critical the code change is to the fundamental principles of Zutha
  + Zutha’s constitution should thus be embodied within the core classes of the source code
  + These core classes should require a huge amount of community support (measured as the size of investment in the change as a percentage of the total Zutha economy) to change
  + The source code would need to be structured so that a wide range of general improvements are possible by extending the existing classes, without allowing core principles to be modified without huge community support
* When a user adds a new owner to an item, he should be required to invest a positive amount of Zuth in that owner
  + If the existing owner currently has no investment in it, the creator of an additional owner should be required to invest in the existing owner item as well
  + The relative amounts invested in each owner item by this initial investor will determine the relative proportion of the item owned by each owner
* The concept of multiple owners can be used to deal with situations in which someone draws substantially from another’s work in creating a new work of their own
  + If an item is no more than a regurgitation or compilation of others’ ideas, then those others should be given the larger part of the ownership of the item
  + If an item takes ideas or creative works from others and adds to them, then both the item creator and the creators of his source ideas should be awarded partial ownership of the item
  + It will be left up to the community to determine what a fair allocation of ownership is in each case
  + Sometimes a work of creativity may use a copyrighted icon as subject
    - The copyright owner may demand credit for the work purely because of the use of a copyrighted concept
    - If the community largely judges that the copyrighted concept is not the primary source of the work’s value, they may vote to assign little or no ownership to the copyright owner

# Mon 3-Jan-11

* Zuthanet
  + I’ve thought of a name for the distributed Zutha network I’ve been mentally developing: Zuthanet
  + Zuthanet could become a successor to the internet, or even just a more advanced parallel technology
  + Zuthanet will be upgradeable
    - one of the major flaws I see in the internet is the incapacity to effectively upgrade its infrastructure and standards
    - When the Zutha community agree on an update, they select an exact time in the future for it to become active
    - Hosts and clients are expected to update in advance of the release date and they should be reminded automatically to do so
    - When the time comes for the update to become active, all hosts and clients simultaneously start behaving in the new way
    - Hosts or clients that have not upgraded by the time of an update may be blocked from the network in various way – not necessarily completely – until they upgrade and start behaving correctly
    - If a large enough group of hosts disagree with an update, they will have to start a parallel Zuthanet
  + If there were multiple parallel Zuthanets with different standards, how much of a problem would this be?
    - It would obviously be very beneficial for there to be a universal standard for actual communication of data so that all networks can interact easily
    - There are also significant benefits to allowing a divergence of standards so that better standards can be found using the forces of competition and evolution
    - All the networks (that want to share data with other networks) would provide their standards open source so that translation protocols can be developed between the various networks
    - It is likely that the degree of divergence of standards between networks would fluctuate as ideas are tried by some networks and the good ones eventually adopted by most

# Tue 4-Jan-11

* User specifies criteria for a notification list of items he would want to be notified of
  + This might include
    - Direct messages from high reputation users or a specific set of friends, or possibly all direct messages
    - Investments in new items by a select group of users
    - Pending changes to Zutha policies/settings/code in particular domains
* Conversation item with priority connection between participants
  + The purpose of this would be for instant messaging between two or more participants, sort of Google Wave style
  + Such a feature would facilitate real-time collaboration of users on various kinds of projects
  + By “priority connection between participants” I mean that the propagation of messages across the network would be tweaked to facilitate the fastest possible update times between current participants in the conversation
    - To achieve this, every message should include information about the current known participants and the initial hosts this message is sent to should prioritize getting the messages to those clients
    - I haven’t yet thought about how pushing data to clients would work from the distributed network; I’ll do this later
* Record average investment size as a measure of how much each individual likes an item
  + As well as the positive, negative, and total worth of items in both currency types, every item should also display the average investment size
  + The average investment size could perhaps be more vulnerable to manipulation, but, then again, perhaps not
  + For a new item, it would be easy to make the average investment size high, but this isn’t a problem; in fact, it’s a good thing since one of the main purposes of the measure of average investment size would be to help users find items that aren’t well known about which are really well liked by those who do know about them
  + For a high worth item, it would be much harder to significantly influence the average investment size (provided no single agent was able to become powerful compared to the masses as is the intention)
  + For a high worth item, therefore, the average investment size would be a good measure of how much users like the item, as opposed to just how many people like it
  + Combining positive and negative investing with a measure of average investment size would provide the benefits of both 5-star or x/10 based rating systems and up/down voting systems, eliminating the main disadvantage of both (namely, the absence of the other).
* Users can donate partial ownership of themselves to other users to acknowledge positive influences on them (like mentors/teachers/parents/friends)
  + I’ve been thinking about a model for education that doesn’t require government subsidies, but which allows people with limited means to gain a good education
  + The ideal incentives would be created for educational institutions if their remuneration was directly related to the success in life of the students they trained
  + Such an incentive structure could be created by having students donate (perhaps as the condition of their education) partial ownership of everything they subsequently produce in life
  + A student may offer, say 5% ownership of their user on Zutha in exchange for a few years of education in a particular institution
  + When that student subsequently creates things or perform actions which attract investment, 5% of their reputation and income will go to that educational instutition
* Automatic increase in worth/reputation of high performance host should scale slower than linearly.
  + To become a really high reputation host, investment from users would be required.
  + A host will be expected to scale its hardware as it gains reputation or risk losing the gains
  + High value hosts should be given responsibility for more traffic, eCash, and data
* How can relevant data be pushed to clients in real time?

# Fri 7-Jan-11

* Teachers “invest” in their students by teaching them for free in return for a proportion of their income over their lifetime
  + This is a better model than education paid by taxes, I think, because it allows competition and isn’t forcing anyone to pay money they don’t’ want to
  + It will also create an extremely powerful incentive for the teacher to do a good job because the success of the students is fundamentally connected to the teacher’s income – and by a market mechanism that will be much more effective at measuring success than whatever an educational bureaucracy would be capable of
  + One issue might be that teachers will only want to teach the most promising students because they will be the most profitable
  + If teaching is a lucrative enough profession, though, there will be a lot of competition for the top students and the second rate teachers will be forced to compete with the second rate students and similarly with 3rd rate teachers and students
  + This seems like an unfair model, yet it would very likely produce a far better education for everyone, despite greater divergence of the best and worst education available. It would be a similar dynamic to the comparison between socialism and capitalism; capitalism is less fair, but more beneficial for everyone
  + Many good teachers may still want to teach disadvantaged kids (or adults), and they will still have the huge incentive to do the best job they possibly can
  + The market model may also lead to less promising students needing to offer a larger proportion of their life income in order to get a better education. This looks like another case of unfairness, and yet allowing this to happen would not actually be the cause of the unfairness (in fact allowing this dynamic would make things better, all things considered, for the disadvantaged); the cause of the unfairness is the natural unfairness of life – some people get born with more talent or more capable parents or a better environment to grow up in than others.
* How does a teacher finance herself when she is first starting out?
  + People could invest in teachers the same way teachers invest in students, or really more like the way venture capital firms invest in startups
  + If an up and coming teacher looks promising, they should be able to attract investment to start a business as a teacher
  + After a few years when some of their students are starting to earn money, they will earn a proportion of this and be able to be self-sustaining
  + The agent who invests in the teacher may ask for a proportion of the teacher’s income over their lifetime, or maybe just for a limited number of years after the teacher starts earning money. There could be any number of models for the investor to earn return on his investment.
* How would this “tax” on students be enforced in an electronic economy?
  + If the student had an identity on Zutha, then a proportion of money and reputation earned through that identity could be passed automatically on to the teacher
  + But it would be very easy for the student to abandon the identity they had while a student and start a new blank identity with no obligations once they finish their education.
  + A society could demand that every person has a single identity associated with biometric markers like DNA and fingerprints
  + A collaboration of Zutha-like networks could share biometric identity info between them to ensure each person has a linked identity across all networks
  + It would be possible for separate networks to allow new members to join without attaching them to the identity they have on other networks
  + Networks that do this would be mainly catering for people who want to break rules on other networks and so they would gain a reputation as such
  + Two independent reputable networks would probably find that is in both of their interests to combine identity information so that they can help each other enforce each other’s contracts, facilitating greater prosperity for both
  + It will eventually be impossible to reliably identify intelligent agents (a category that will probably eventually include more than humans) even by biometric signs like DNA and fingerprints
  + Even if it is impossible to directly enforce contracts such as the one between a student and teacher, it may be possible to enforce them indirectly through cultural norms
  + Some teachers, even early on, may decide not to bother making a binding contract with their students about how much of their income the student must pay them over their lifetime. Instead, they may rely on the student being honourable enough to offer a fair attribution of appreciation to their teacher.
  + If a person has not declared any indebtedness to anyone in an online identity, this would say something about them to the rest of society which is unlikely to be beneficial
  + People are likely to come to expect others to have shown appreciation towards people who have had a positive influence on them; if they encounter people who are not contributing anything to others, they are unlikely to want to invest in them or engage with them in productive activities like trade or collaboration, thus making it very difficult for that person to earn income.
  + Then again, if a person produces things of value, they are likely to be able to be able to get rewarded for doing so by people who are only interested in the thing of value rather than its maker
  + Still, scorning ones teachers (whether parents, professional teachers, or other sources of learning), is likely to have negative consequences in ones dealings with others
  + Teachers will have an incentive to teach to their students the value of the attribution of appreciation
  + There are a lot of human behaviours that don’t make a lot of sense from a selfish point of view, but which are valuable to a whole society when the majority of people act that way. One of the challenges to societies of the future will be to foster the most beneficial of such behaviours (including new noes) even as it becomes easier for individuals to control their own internal “wiring”

# Sun 9-Jan-11

* The incentives appear to be in the right place for Zutha to act as a beneficial governing organisation for all its participants
  + Thinking about Zutha for a moment as an intelligent entity in its own right, it is in its interests to increase the prosperity and quality of life of as many people as possible so that they will have more resources to contribute to making stuff that people like. The more happy people there are in the world and the more good ideas and good stuff there is for them to like, the more they will invest resources in Zutha to show their appreciation, profit from advertising the good stuff they find, and help perpetuate the engine that helps make them more prosperous.
  + Zutha is not a single intelligent entity, though, it is a collective identity whose actions are governed by the community of people who mutually share in and contribute to its profitability.
  + The Zutha community thus has the same incentive as the conjectural Zutha entity – to improve the prosperity and quality of life of as many people as possible and encourage them to contribute to the Zutha community.
  + Even if some of the people Zutha makes prosperous do not directly participate in the Zutha community, they may produce wealth – in the form of ideas or physical stuff of value – which can facilitate greater wealth of members of the Zutha community. Thus it is still in the interests of the Zutha community to work towards to the prosperity, and ability to contribute value, of all people.
* How Zutha can support public goods
  + Zutha funds private initiatives in proportion to their relative Zutha reputation (which is variable in real time according to the Zutha investment market)
  + The amount of Zutha funds invested in any given public good is determined by the same community voting mechanism
  + Example: public toilets
    - Individual entrepreneurs or firms can start up a business building public toilets
    - When individuals visit a (free) public toilet, many of them may – once a culture is in place to facilitate this – decide to express the degree of their satisfaction with the quality of the toilets (which they may measure by such things as wait times and cleanliness) by investing a small amount in the Zutha item representing the toilet facility in question.
    - A certain proportion of Zutha funds will be allocated to paying for public toilets, and this amount will be further divided among individual toilet facilities based on their relative reputation
    - If a particular facility becomes very popular, it may become overcrowded, resulting in the diminishment of its income as well as a corresponding increase in the relative income of competing toilet facilities
    - There would need to be some way to make sure a fair amount of resources were allocated to any given area
    - In wealthier areas, people will be able to invest more in their toilet facilities, thus dragging resources unnecessarily and unfairly away from poorer areas
    - Perhaps Zutha’s resources could be first allocated to regions according to some metric like population, and then allocated within each area according to the relative reputation of competing firms.
    - This method would increase the effectiveness of the resource reallocation effects of a facility becoming overcrowded and thus less popular.
    - A firm running a toilet facility which is becoming overcrowded would thus have an incentive to use the resources it is granted by the current popularity of the facility to build more facilities, thus avoiding the overcrowding and resulting diminishment of the value of its assets.
    - If there were not enough resources being allocated to public toilet facilities, firms may not be able to afford to build more facilities even as the current ones become overcrowded
    - The community would have to recognise this problem and decide collectively to allocate relatively more Zutha resources to public toilets.
  + It may be desirable for the resource allocation between departments to be decided independently for different areas.
    - Each area would be allocated a certain amount of resources according to its population
    - Within each area the community would determine the relative amount of their local resources to allocate to any given public good
    - It may not even be necessary to restrict the influence over allocation in an area to people who live in that area
    - People will (roughly speaking) act according to their own self-interest, so people who live in an area will be interested in participating in the discussion about the allocation of resources in their area
  + What happens if every competing source of a public good has a negative reputation? (probably due to lack of funding for that public good)
    - The resources allocated to that public good should still be provided, otherwise things would be even worse
    - There needs to be a way to determine relative allocations appropriately even for negative reputation
    - If firm A has reputation -4 and firm B has reputation -8, the sensible allocation of resources would be 2/3 to firm A and 1/3 to firm B because firm B is twice as bad as firm A so it should get half as much
    - If firm A has rep -4, firm B has rep -8 and firm C has rep -8, allocation should be 50% to firm A, and 25% each to firms B and C
    - 20/4=5; 20/8=2.5; firm A gets 5/10, firms B and C each get 2.5/10
    - What if there are some negative rep firms and some positive?
    - The resources should just be divided among the firms with positive rep, I think
    - Actually I don’t think there is any other way that makes sense; if a firm has 0 rep they clearly get 0 resources based on any natural formula for allocation between positive weightings, so why should firms with less than 0 rep get more than those with 0?
    - If only a single firm has positive reputation then they should receive all Zutha resources allocated to the public good they are providing. If they use those resources inefficiently and someone else decides they can do a better job, that competitor is likely to very quickly start taking away income from the inefficient former monopolist.
  + It will take an initial sum of capital to enter many public goods markets and compete for Zutha funds, but this is not really a hurdle; it should be no harder to get investment than for any other business despite the fact that the income for ventures into providing public goods will come mostly from a government-like entity.
    - There will be no requirement that firms competing for Zutha funds must be non-profit organisations
    - If a firm can provide the public good and also make profit in a competitive market for Zutha resources, good for them
    - Indeed, the only way the investment is likely to be available for ventures into these public goods markets is if they can be profitable
    - If there doesn’t appear to be enough resources provided by Zutha for a particular public good, firms will not enter the market and the public good will not be provided
    - If enough people in the community want the public good, though, they will chose to allocate enough Zutha resources to provide it

# Tue 11-Jan-11

* Many items should have subitems that represent aspects of them that can separately be praised or criticized
  + E.g. a comment item might have subitems like “objectivity”, “accuracy”, “tone” etc.
  + Investment in the attribute items aggregates into the overall worth of the item
  + Users can also invest in the item itself if they can’t be bothered providing a more detailed breakdown of their opinion
  + This system also provides more scope for being an early investor because there are multiple separate investment pools for each item
* Ownership should be able to operate in a hierarchical arrangement
  + E.g. Harry Potter
    - J.K. Rowling owns Harry Potter (novel series)
    - The book series owns each of the individual Harry Potter books as well as characters that appear throughout the series.
    - An event that occurs within the book Harry Potter and the Goblet of Fire would be owned by the book item
  + E.g. Comment items
    - Comment items would be owned by their author
    - Comment items may own attribute subitems such as “objectivity”, “accuracy” etc.
  + The worth of an item (which is shown in its worth bar wherever it appears) is the sum of investment in itself and recursively all the items it owns
  + On the Stats subpage of an item, the breakdown of the sources of the item’s worth should be shown
    - At the top of the list should be the direct contribution of investment in the current item
    - Following should be direct children (in terms of ownership) showing the amounts invested in each of them directly
    - Under each of these direct children, in a tree structure, should be each of their children and so on covering all descendants of the current item
    - It would be handy to have a feature to collapse a node so that none of its descendants are shown. The collapsed node should then show its total worth (the sum of investment amounts in itself and its descendants) instead of just the investment size in itself.
  + If the worth shown in the worth bars of item summaries was an aggregate, it wouldn’t really work to have an interface to invest in items through the summary form of an item that appears in search results
    - I’ve been weighing for a while the idea that investment should only be possible through a somewhat more rich interface which would only be accessible for the current item (no through the summary interface that appears in search results)
    - The richer interface should provide a box for adding a brief comment justifying the investment
* The concept of comments and investments could potentially be merged
  + It should possibly be compulsory to write a comment for every investment
  + There are some clear cases where this would be essential, such as when someone investments negatively in a user, or in a user’s action (like an investment or modification of an item)
    - In these sorts of cases it would be essential for actions to be justified so that there is transparency involved wherever there are actions that potentially have a powerful impact on the cultural and paradigmatic trajectory of the site.
    - If other uses enter a dispute with the intention of trying to resolve it, they should be very critical of users who have acted strongly with investment without justifying their actions.
    - Justifications which are poor should be criticized in their own right, but if there is a minimally credible justification at all, the user should not be punished for misconduct, only confronted and disagreed with for being mistaken.
  + In some cases, a comment as information-barren as “I love it” might be considered acceptable, say for such things as songs where it would be perfectly acceptable for a person to be unable to find words to describe why they like something.
  + Even in these cases, though, I think the comment should be compulsory anyway in order to draw out as much tangible sentiment as possible to the investments
  + Requiring comments to be associated with investments would also be effective at reducing the degree to which the investment market becomes a purely mathematical profit seeking venture
  + If investments are always comments, then they should be shown in the comments page alongside all the comments that don’t have an associated investment
  + Investment would semantically be a sub-type of Comment
    - Actually it might be semantically cleaner to have items created as a result of investments have multiple inheritance from both Comment and Investment
    - Investment would act like a Scala Trait, providing the fields for specifying the amount invested as well as what item was invested in
  + Investment items should show the amount invested, just as a number, in the summary form of investment items that appear in a comments page
    - There should be a facility (probably using scope) to determine which occurrences, if any, appear in an item’s summary form
* Investments or comments on an attribute sub-item of a parent item should also appear in the comments page of the parent item
* It could be handy to be able to write a single comment which is associated with multiple investments, probably one investment into each attribute sub-item of the focus item
  + Such a comment item would really be closer to a review, so perhaps it should be described semantically as such
  + A Review could be a sub-type of Comment
  + Reviews could then optionally “mix-in”, Scala style, an Investment-Suite
  + An Investment-Suite would allow multiple fields specifying investment/target pairs
  + An Investment-Suite should probably only allow investments in a single item and its attribute children
* Users should be able to star items they like
  + Every item that a user has invested positively in should be automatically starred
  + This feature will allow users to have their own personal bookmark system using the Zutha semantic architecture
  + Browser plugins could be created to allow easy creation of and starring of Zutha items that correspond to web pages, as well as an integrated browser of a user’s starred items in Zutha – probably filtered to web pages for a browser plugin
  + Starring a page should be a one click process
  + If a user clicks to star a page and it does not yet exist on Zutha, a simple interface should appear to allow the user to give the page a unique name and basic description and categorize it using the Zutha semantic architecture.
  + Information about which items a user has starred should not be used automatically for public ratings of items
  + It would be useful to be able to find out which items a particular trusted user (or group of trusted users) has starred, though
  + Finding the items starred by a particular set of users would be easy to do with a search using the semantic query language I’m going to develop
    - E.g. find items starred by my friends
      * Starred:Starred-by(starrer=friend:friends-with(friend=Chris\_Barnett))
      * Starred-by(starrer=(friends-with(friend=Chris\_Barnett)))
        + Association filters that do not give an explicit output role should result in all items with roles in the matching association items, except for items that played the role(s) on which the filter was placed.
        + So Chris Barnett would not be in the result set for “friends-with(friend=Chris\_Barnett)” because in the only matching associations, Chris\_Barnett is playing the role that satisfies the filter condition.
        + If there was Friendship association between the Chris\_Barnett item and itself, then in that association, one of the Chris\_Barnett roles would match the filter, and the other would be added to the results.
      * Starrer=(friend=Chris\_Barnett)
        + This finds items that play any role in any association in which the “starrer” role is played by a different item which is playing any role in any association in which some other item plays the role Friend.
    - E.g. find items starred by users who starred the book The Machinery of Freedom
      * Item:Starred-by(user=user:starred-item(item=The\_Machinery\_of\_Freedom))
      * Starred-by(user=starred-item(item=The\_Machinery\_of\_Freedom))
      * Starrer=(starred=The\_Machinery\_of\_Freedom)
      * I’m not yet whether the role name should indicate the semantic role of items that play that role, or the type restriction on items that play that role
      * When I lay out the issue explicitly like that it seems clear that role names should describe the semantic role they represent
      * Starred\_Item:starred-by(starring\_user=starring\_user:has-starred-item(starred\_item=The\_Machinery\_of\_Freedom))
      * This is much clearer, though unfortunately verbose
      * Ctrl-space completion would be really useful for writing such queries; perhaps I can implement that with javascript eventually.
  + There should be a shortcut from every item X to a search for items starred by users who starred X
    - It would be good to be able to combine this search with the feature that filters the worth of items by investments made by investors in X
    - It would be neat to achieve this using only the query language
* The query language should include a feature to filter not the results, but the investment items that contribute to the worth of the results
  + E.g.1 find all books, filtering their worth by investments by investors in The Machinery of Freedom
    - type=Book wfilter:owner=(type=User owns=(type=Investment target=The\_Machinery\_of\_Freedom))
  + wfilter mean “worth filter”; perhaps worthfilter should be used instead, or at least allowed as a more self-descriptive alternative
  + The query assigned to the wfilter function will return all items owned by investors in The Machinery\_of\_Freedom, not just investment items, but the wfilter operation will only use the results that are Investments to determine the worth of the search results
  + If would more efficient to do these sorts of searches if there was a direct association between investors and the items they had invested in
  + E.g.2 same query as e.g.1
    - type=book wfilter:owner=user:invested-in(item=The\_Machinery\_of\_Freedom)
* A search item should cache its results
  + This would be particularly useful for very expensive searches
  + When the results of a search item are requested, the cached results should be returned immediately, and, depending on the refresh-rate settings of the search item, a live search should then be performed and the results sent in real time to the client
  + The search item should show how up-to-date its results are
* The ability to perform really resource expensive searches should be restricted to high reputation users
  + If a user X does not have sufficient permissions to perform a particular expensive search, he may request that a high reputation user Y turn the desired query into a saved search item and send X back the link. X can then see the cached results, but is unable to request a refresh of the results.
* Pretty much every decision that has to be made through Zutha should be crowd-sourced via the investment market
  + For the sake of speed, many decisions should be allowed to be made immediately by high enough reputation users, but would then be open to dispute
  + Any user should be allowed to open a dispute about an action (say giving a user ownership over an item).
  + A dispute should have a predetermined (eventually also determined by the market) expiry time, depending on the type of the action that is being disputed
  + Actually there should be a period of time that must elapse before a decision made by the investment market can take effect; a kind of activation delay
  + Between the moment of activation and the moment of expiry, if the absolute (in the case of a Boolean decision) or relative worth of a decision item in the dispute passes a certain threshold, that decision is enacted.
  + Perhaps it should also be required that the dispute remain for a minimum period of time at a particular verdict before that decision is enacted.
  + If a user does something that’s obviously stupid, it would be important for it to be possible for another user to quickly reverse the error (or misconduct as the case may be)
  + The default status while a dispute is being resolved should probably be this reversed position – that is, the status that existed before any action was performed.
  + Actually the act of reversing an action should be synonymous with starting a dispute
* Prediction Markets
  + Any user should be able to set up a Prediction Market, which they propose to host
  + The creator and host of the Prediction Market item is the one who determines the winning outcome after the expiry date
  + A Prediction Market host’s legitimacy at correctly determining the winning outcome will be policed only by reputation
  + Hosts will have to build up a reputation for being serious about getting it right; if a host specifies a winner that is widely disagreed with, his reputation will suffer as a result, making it more difficult to get users to participate in future Prediction Markets that he hosts
  + While a Prediction Market is in progress, normal investment mechanics will operate across candidate items in both the positive and negative pools
  + At the moment of expiry all investment operations are frozen on all candidate items
  + When the winning candidate is determined, all currency invested is aggregated into a single pool and divided proportionally among winners
  + If there is a single candidate item and, say, the result is positive, all the currency from both positive and negative pools is distributed among investors in the positive pool in proportion to each investor’s stake
  + If there are multiple candidate items and one winning candidate, there will be a single positive pool winner and many negative pool winners
  + How should the winnings be divided up among these positive and negative winners?
  + If the winnings were simply divided among investors in proportion to their investment size, regardless of whether in the positive pool or a negative pool, then there would be more incentive to invest in negative pools since there would be far more chance of being right
  + This problem would be a vicious cycle because the more people that invest in negative pools, the less, proportionally, of the winnings would go to the single winning positive pool, making it yet less attractive to invest in positive pools.
  + I think the winnings should be divided so that a proportion of it, say half, is divided among investors in the single winning positive pool, and the other half is divided among investors in the many winning negative pools
  + This would provide a greater potential profit opportunity to investing in positive pools, but higher reliability of positive returns of investing in a negative pool
  + There should be a greater incentive to invest negatively in a pool that you believe to be overvalued
  + At the moment, if there is a candidate that everyone believes will not succeed, everyone will want to invest in it. If this unpopular candidate does lose, the huge size of its negative pool will draw profits away from the investors in riskier negative pools who were actually contributing more valuable information
  + The proportion of the winnings distributed to a particular pool should be inversely proportional to the size of the pool
    - Let W=total winnings; X=sum of worths of winning pools; x1,x2,x3 = sizes of winning pools; w1,w2,w3 = amount of winnings distributed to respective pool
    - So W=w1+w2+w3; X=x1+x2+x3
    - Let’s take x1=5; x2=5; x3=10
    - X=20
    - Pools 1 and 2 should each receive 2/5 of winnings, pool 3 should receive 1/5
  + Should the one winning positive pool participate in the same relationship?
  + Let’s take the case of a certain outcome as a guide
    - The most aesthetically pleasing outcome would, I think, be to have equal pools in every anticipated (with certainty) winner
    - This outcome would be achieved by having the lone positive winner participate on equal terms as a winner as all the negative items
    - If there was an anticipated winner (whether the positive one or one of the negative ones) that had slightly less worth than the others, it would be slightly more profitable than the rest on market expiry and would attract investment until it equalizes with the rest of the anticipated winners
  + What about if the outcome was perfectly uncertain?
    - Mathematically this implies that every single outcome has an equal probability from the perspective of every investor
    - Every pool, positive or negative, of every candidate should be equal
    - This single absolute value (the size of every pool) would be determined by the salience of this particular Prediction Market item
    - If a single pool is slightly smaller than the others, it would provide a slightly higher expected return, since all pools have equal probability of success (50%) and a pool being smaller than the others means it will draw slightly more winnings per unit of investment.
    - Similarly, if a pool is slightly larger than the others it will have a low expected return than the others and investors will sell until it equalizes with the others.
  + My intuition says if these two extremes result in these neat outcomes, that the possibilities in between will produce elegant behaviour also
* Conditional Investment Markets
  + After the condition has been determined, the candidates that satisfy the condition participate in the winnings distribution as described above
  + Candidates that don’t meet the participation condition have their investments liquidated and returned to their owners
  + Liquidation is done by virtually selling off every investment from the top down and returning the proceeds to the owner of that investment
  + The market is frozen at the moment the condition is determined
  + The market may have to remain frozen for a long time until the conditional outcome is determined
  + If it is looking likely that a candidate will not meet the condition, there will be an incentive for early investors in that item to sell and earn profits in the standard way (buying early and selling at a higher price), rather than wait for the item to be liquidated.
  + Since this effect will apply to both positive and negative pools in items that are going to be liquidated, the relative sizes of these candidates whose conditions will not be met, should not be substantially warped.
  + If the condition is going to be influenced by the outcome of this prediction market such as if the condition is a policy decision which will be made based on the recommendation of the prediction market, then this effect will lead to decreased salience (investment volume) of the candidates that are not looking likely to pass the condition
  + This could actually be a useful effect, particular in the case of basing a policy decision on a conditional prediction market
    - The best policy decision would be indicated not just by the predictions associated with making that decision, but by the increased relative volume of investment in candidates associated with the predicted policy decision
    - The policy decision predicted by the market may be based on more factors than what people actually believe the best policy to be, since the actual policy decision is likely to be made based on more factors than what this prediction market indicates is the best policy.
    - If there was some way to set up a rigorous rule-based system with which policy decisions would be based directly on the decision indicated as the best one by this prediction market, then the effect of increased investment volume in candidates of the likely decision could provide a feedback loop which strengthens the signal as to the decision suggested by the market.
    - The paradox is that if there was a rigorous rule-based system for determining which policy decision is based on the predictions of the market, then no further indicator of the best decision is necessary.
    - Perhaps, if the decision-maker simply had a reputation for aiming to make the best decision based on the market predictions, then the market could help that decision-maker along by indicating which decision the crowd thinks is the best one – that is, the one the decision-maker should pick if he indeed is trying to make the best decision.
    - This structure has self-referential characteristics and I’m beginning to see that such characteristics may be problematic. I think I need to learn and think more about self-referential markets so that I can better understand what undesirable consequences they might be prone to inducing.

# Wed 12-Jan-11

* Prediction market division of winnings
  + I got my probabilities wrong when I was working out whether winnings should be divided evenly among all winners, or first divided in half between positive and negative winners
  + If there are 10 candidates, a negative investment has a 9/10 chance of succeeding, while a positive investment has a 1/10 chance of succeeding
  + If winnings are divided evenly between positive and negative winners, then for pools of equal size, the positive pool will receive 9 times as much than the average negative pool, which is the desired result given the probabilities
  + Thus for perfectly uncertain candidates, expected return reaches an equilibrium where all pools are the same size
  + There still isn’t any incentive to invest negatively in a candidate you believe to be overvalued
  + I think this would be important; one problematic scenario is a single powerful agent investing hugely in a particular candidate, perhaps for the purposes of influencing a policy decision
  + If the potential winnings of a negative investment was positively related to the size of the corresponding positive pool, then there would be an incentive to invest negatively in this pool when the positive side is too high
  + These mechanics would be achieved if every candidate operated as a separate competition; the winnings for a given pool comes only from the directly opposing pool
  + If a particular candidate is very unlikely to be the winner, then there will not be much incentive to invest positively in it, thus not much incentive to invest negatively in it, and this vicious cycle will likely prevent it receiving almost any investment at all
  + The most sensible extension of the single candidate scenario seems to be to have the same number of pools as possible outcomes
  + So perhaps, except for Boolean competitions, the negative pools should not be used
  + If a single candidate has too much relative worth, there will be an incentive to invest more in the other items since the over-valued candidate, if it ends up losing, will provide correspondingly more profit to the winner.
  + These mechanics also mean there will always be an incentive to invest in any candidate if you believe it is the most likely to win, because whichever candidate wins, it will make substantial profits (more if there are more candidates and more uncertainty)
  + Conditional prediction markets are still easy: the candidates that don’t meet the condition are liquidated and the currency in the remaining candidates is divided up among the winner
  + What if someone invests in every candidate in proportion to its relative worth?
    - He will successfully eliminate all risk and all profit
    - If there is a tax on winnings this behaviour will result in losses equal to the tax with 100% probability
    - Actually perhaps I’m wrong about this
    - If there are two possible outcomes with equal pools in each and you invest equal amounts in each, then when one wins you will get the winning investment back plus the appropriate proportion of the losing pool, which will be the same size as your investment in the losing pool
    - No, I’m right. Phew! That would be a disastrous loophole if it existed.
* How to get around online gambling regulations
  + It might be possible to have real money investments in Zutha legally considered charitable gambling, which may make it more accessible to more of the world
  + It could even be possible to set up a legal framework whereby investments are legally defined as donations to the item owner, but the item owner has the right to elect to have some of his proceeds be returned to investors when they sell, as a reward to encourage further donations.
  + In this way, an experiment could be conducted to compare the relative effectiveness, for earning money for item owners, of donations versus profitable investments.
  + An added benefit of such a legal scheme is it would make it much less attractive for wealthy entities to engage in trading in really high volumes purely for the purposes of making profit. This fact would reduce the profitability of the site, both for Zutha itself as well as item owners, but it may be beneficial for the site’s reputation.
  + One problem with this mechanism is it may not be possible for Zutha to tax such “donations”
  + It would still be worth doing in order to generate incentives for people to create valuable content and put it on the site
* Real money donations can be allowed early on
  + Real money donations should act as permanent investments, where the whole value of the investment, instead of just 10% or so, goes to the item owner
* Investments could be passed around as “donations” of financial assets between investors and item owners
  + When a user invests in an item, it could be considered the donation of a financial asset to the item owner
  + Zutha’s tax could be incorporated as part of the financial asset, basically being a property of the asset that when you sell it for cash, you must pay Zutha about 1% of its value
  + The item owner can chose to “donate” back to the investor, a proportion of the investment asset
  + Item owners can arbitrarily chose how much of the investment they donate back; it could be 0%, 90%, 50%, or whatever they think will yield them the greatest revenue
  + This system is even better than the flat 10% commission to the item owner that I had planned, since it removes yet another arbitrary parameter from central control and allows it to be determined more optimally by a market mechanism.
  + It may also be good that a large amount of the worth of an item is owned by the item owner, since he has conflicting incentives to sell for profit or retain the high worth of his item. A major tipping factor to these opposing incentives will be the item owner’s own perception of the value of his item, or at least his prediction of how valuable it will be to others in the future.
* Should this same mechanism be applied to the Zuth market?
  + The problem in the virtual currency market is there would be no incentive at all for item owner to ever sell his own share of investments in his items as the item’s worth contributes to his reputation and thus his Zuth income
  + Another problem is that I don’t want easy and arbitrary mobility of zuth between users, and this would be made too easy by donations
  + Then again there are situations where it would make sense to donate Zuth to someone instead of temporarily investing it; paying someone for answering a question of yours for example. In this case, it wouldn’t really make sense to be able to take away the investment later, since nothing about the value of the one-time service would subsequently change.
  + Also, I don’t think it is practical to try to stop people moving money around by technical barriers; the whole basis of the site is that money moves around and if someone wants to get money directly to a particular user, they will find a way.
  + I think the only way to avoid the problems of easy Zuth transfer (such as transfer of reputation to an undeserving recipient) is policing through reputation itself.
  + If user A donates a huge amount of zuth to one of user B’s items and other users think the transfer unwarranted, they can invest negatively both in item B’s item and user A’s donation action, thus reducing the reputation of both participants in the unwarranted trade of reputation.
  + User B may not be a participant in the misconduct, but he would not be being punished; his undeserved reward would merely be taken away.
  + If User B sells the huge donation, he will then have a huge amount of cash but probably low or even negative reputation (if he was a low enough reputation user).
  + The users who were investing negatively to counteract the unwarranted donation might now decide to sell their investments because they are no longer counteracting the reputation effects of the unwarranted donation.
  + User B now has a huge stash of Zuth that he doesn’t deserve and he has reputation similar to what he had before
  + This is not good
  + If user B starts investing undeservingly in his own reputation or passing the cash on to further undeserving users, he may be punished somewhat, but he essentially has nothing to lose so he can donate the zuth on again and abandon the user B account if necessary.
  + The main problem with the ability to donate, is that it allows users to pass all their zuth to other users without being punishable
  + With investments there is not as much of a problem because a misbehaving user can be punished by having his reputation pushed into the negative; eventually the negative salary imposed by negative reputation will force him to sell the offending investments.
  + I think, therefore, that the Zuth economy should be based exclusively on investments – no donations
  + A tax (determined by the community) is taken off every investment when it is sold, but no zuth goes directly to item owners
  + Item owners earn income only through the salary they receive in proportion to their reputation, which is determined by the amount invested in their items
  + The total amount of zuth distributed as salary should slightly exceed the total amount taken in tax, at least for the developing stage of the Zutha economy. This will cause a sort of inflation, though there are no real assets for the currency to be inflated against.
  + The increase in currency supply will be necessary as the number of users and items increases
* How should this mechanism apply to negative investment?
  + If the item owner had any control over the amount of investments he kept, he would just keep all of it and immediately sell, making negative investment redundant
  + Negative investments cannot really be modelled as donations, because they are not intended to reward, but to punish, the item owner
  + Actually negative investments could sort of be considered as donations, just not donations to the item owner; they would instead be donations to the community as a whole – a philanthropic act of punishment.
  + Zutha is really the representative of the community as its resources are (or will eventually be) controlled by the community
  + A negative investment should be considered a donation to the Zutha community
  + The Zutha community should collectively vote on the proportion of such donations it chooses to donate back to donors as a reward for donating
  + The amount the Zutha community choses to keep corresponds to what used to be the Zutha tax
  + The share of the investment owned by Zutha would be immediately sold, making it functionally equivalent to a tax – or even the equivalent of the small percentage donation often given to non-profit organisations for operating costs, when people donate (or micro-lend) through it to another entity.
  + The investment asset itself still has an associated tax which is extracted on exchange of the asset for cash
  + There would thus be two different taxes taxes on negative investments, one on purchase and one on sale
  + Positive investments would have two kinds of taxes as well; one on purchase taken by the item owner, and one on sale taken by Zutha
* I think Zutha will need to have become a fully non-profit organisation (that is, no profit for any kind of central management or owners) by the time real money investing is activated, in order to make all this work
  + As early as possible, I want to start assigning a large proportion of Zutha’s income to rewarding contributors – identified and portioned proportional to reputation
  + Most of the rest of Zutha’s income should be divided among various causes chosen by the community
  + A proportion should also be assigned to remunerating hosts for their operating costs
  + The portioning of Zutha’s income among all these outlets should eventually be completely determined by the community
  + Eventually – in fact as soon as possible – my only income from Zutha should be through rewards for having a high reputation, and remuneration for operating a host.
* How should community decisions with a continuous range of options be made?
  + Examples include the determination of Zutha’s cut on negative investments, or its tax on converting investments to cash
  + If the available range was bounded, the weighted average of investment in the two extremes could be used
  + So for determining percentages, there would be a candidate for 0% and a candidate for 100%
  + Users would invest positively in the 100% candidate or negatively in the 0% candidate if they want the average to go up and the opposite if they want the average to go down
  + It seems like negative pools are unnecessary in this situation, just as with multiple candidates in a prediction market
  + This system wouldn’t work if the range was unbounded, but I cannot think of a decision that would vary across an unbounded range
  + Negative investment in the anchor candidates would allow an unbounded range to be possible, but it would be a very confusing dynamic
  + One problem with this system is that it would be very difficult to get to either extreme – in fact any positive investment in an anchor prevents the extreme represented by the other anchor from being reached
  + It would also be disproportionately easier to push the weighted average towards the mid-point between extremes
  + When the outcome should be ranging around 1-3%, as with the tax parameter, it would be dangerously easy for someone to invest a large amount in the 100% candidate and drastically push the average higher.
  + This could be done with a whole lot of discrete candidates where the outcome is determined by the weighted average of all candidates
  + This doesn’t stop someone creating a candidate with a very high value and investing a lot in, magnifying his influence by the extremity of the candidate’s value
  + What if the median were used instead of the mean?
  + Only I can’t figure out how the median would work with weighted values.
  + I don’t think using median can produce a result between two candidates (except if the 50% point is exactly divided at a gap between two candidates)
  + What if the candidates were first normalized to a uniform unit scale for calculating the mean, and then the calculated mean rescaled according to the actual separation between candidates?
  + I just tried this method and it works admirably
  + If simple mean is used, extreme outliers with very small weight can have a huge impact. If the mean is calculated on the candidates’ index, rather than their values, such outliers are practically ignored (but not completely, which is good)
  + This method does not work with negative weight values, though, so negative investment should not be used for these decisions
  + This method will allow decisions across a potentially unbounded range, though the allowed range of candidate values can easily be limited if the range should be bounded (such as for a tax percentage).
  + I just realised this system can be rigged by creating a heap of candidates with the same extreme value and investing a small amount in each. This hack actually quickly becomes more effective than investing in a single outlier in the standard mean method
  + If every candidate indicated a range then the median (the halfway point in terms of area) would be an appropriate compromise of all candidates, I think
  + Perhaps every user should provide a desired range with every investment
  + Every investment would be like a rectangle whose length and x position express the parameter range which is acceptable to the investor, and whose area represents the size of the investment.
  + The chosen parameter value would be the x value which results in 50% of the collective areas of all the rectangles lying on either side of it.
  + A visualization of the distribution might look rather like a city
* Affiliate marketing integration
  + When a user purchases a product via Zutha, the vendor may choose to reward Zutha with a proportion of the sale price
  + The amount given in reward to Zutha by the vendor should be given to the buying user as an investment in the purchased product’s corresponding item on Zutha
  + Zutha will take its normal tax off when this investment is sold
  + Eventually a system should be set up for vendors to accept “purchases” directly through Zutha
  + A user may even be able to simply press Buy on the Zutha item page, or they may identify their Zutha user account to the vendor during purchase
  + An investment will be created equal in value to the price of the product
  + The vendor will have determined how much of this investment it returns to buyers as a reward. This amount may differ from the amount returned when the product is not being purchased.
  + Once the vendor is notified of the successful investment in the relevant Zutha item, it can send its product to that user by whatever means the user provides for contact (i.e. address for delivery; email for activation link or product key etc.)
  + The vendor now owns an investment in its own product item on Zutha which it chose to manually or automatically sell at any time
  + The buyer also owns an investment in the product item which she may sell at any time
  + Both vendor and buyer have an incentive to leave their investments in the Zutha item if they have reason to believe that the product will continue to gain popularity
  + They take a risk by doing so, however, because they may lose some of their investment value if earlier investors subsequently decide the product is going to lose popularity and sell.

# Thu 13-Jan-11

* Users should be able to create requests for a Prediction Market item and ask a reputable judge to take control of judging it
* Conditional prediction markets should be integrated into Zutha’s internal policy making process
* I need a good way to reward prediction markets on continuous variables
  + I can use the system I devised for voting on decisions about a continuous variable
  + When the correct value of the continuous variable is known, the winnings are divided among investor’s whose rectangle crosses the variable’s actual value
  + Winnings will be divided among winners in proportion to the height of their rectangle at the winning point
* It would be good if there was a mechanism for investors benefiting from speculation before the continuous variable being betted on is actually determined
  + This is possible in conventional prediction markets because bets are assets which can be bought and sold either long or short
  + I think I can expand the vase concept used in my standard items to 2 dimensions, such that it can be imagined as a kind of trench where the position along the length of the trench corresponds to a value of the continuous variable
  + When users invest in a continuous variable item, they select a range (of that continuous variable) over which they wish to invest.
  + This is effectively like making a separate investment (in the standard Zutha way) for every infinitesimally small increment of the continuous variable
  + Such a scenario is perfectly suited to integration
  + When users sell an investment over a region of the continuous variable, it is like selling their investment in all the individual investment vases corresponding to that region
  + All the dynamics and incentives that apply to the 0 dimensional investment scenario extend to the 1 dimensional scenario
    - There is an incentive to invest early over a particular region of the continuous variable
    - If others invest in that region after you, you can sell your investment at a higher value, taking that value by slightly shrinking the value of all the other investments across that region
  + It should be possible the sell an arbitrary region of your existing investments
    - When selling, you would specify the range of the continuous variable over which you want to sell, as well as how much of it you want to sell
    - The amount you own at any given point across the region may differ
    - If a user asks for 50% of his investment to be sold, should all points across the sold region be reduced by 50% or should the highest value points be prioritized? The lowest value points?
    - Why not let the seller chose those things as well
    - An investment market could be used to determine the default behaviour
* The investment model can be extended to arbitrary dimensions
  + Consider a prediction market to determine the location of a missing person in 2 dimensional space
  + Users wager on the latitude and longitude of the person’s position
  + The conceptual model of such a 2-dimensional continuous variable investment scenario would be that for each point is an associated investment vase
  + This prediction market could also be modelled as two completely independent markets; one on each position dimension
    - This would mean you could win by just getting the longitude right, but being way off on the latitude
    - I’m sure such a market would still yield useful results, but I think the problem space is better expressed by the variables being dependent
    - An investor shows more relevant knowledge by predicting close to the actual position of the lost person rather than just what his longitude is
  + An example of a scenario that would be modelled on more than 2 dimensions: the trajectory of an asteroid in time and space as it passes near earth
    - There might be any number of functions that could express the trajectory, each with their own set of continuous variables
    - These trajectory functions would be in terms of the variables (x,y,z,t), but the coefficients of the terms in the function would be the continuous variables that would be better upon
    - Users should be able to propose new functions with a specified set of coefficients to bet on
    - The investment model will cope with multiple independent multi-dimensional spaces all part of the one market
    - When the correct coefficients are (approximately) determined as a multi-dimensional region in one or more of the discrete solution spaces (each representing a different trajectory function), the winnings are divided among investors whose investment parameters lie in the winning space – divided proportional, of course, to the relative size of each of the investments in the winning space.
  + What happens if there are no investments in the winning space?
    - I can’t see any other option but to liquefy all the investments in the same way as investments in a conditional prediction market item that does not turn out to match the condition
    - The fact that there may be no winners should not prevent the dynamics of the competition from encouraging investors to select investments based on the best of their knowledge.
    - Investors who are early investors in regions that later become popular can still profit by selling their investment after it increases in value
    - Making profit this way would be technically dependent on anticipating the moves of the market rather than actually correctly anticipating the final outcome
    - The more unlikely it is perceived to be that anyone will actually guess correctly, the closer this dynamic will approximate an investment market in opinions – such as the worth of a book.
    - The key to why I think both of these self-referential scenarios will still yield useful results, is that no individual can influence the mechanics driving other people’s decision making process; instead, they must assume the majority of others are basing their decisions on the advertised premise of the item, whether it be its relative subjective value or the best approximation of the trajectory of an asteroid passing near earth.
    - For all intents and purposes, betting on what other people will think other people will think other people will think is the relative worth of the movie Avatar compared to vanilla ice cream is the same thing as betting on what you think is the relative worth of the movie Avatar compared to vanilla ice cream.
      * This is not the same thing as specifying how valuable the movie Avatar is to you relative to vanilla ice cream
      * Instead, the incentives encourage you to determine in a more objective way what their relative worth is across *all people*
      * I think this will actually yield even more useful results than if everyone was just offering their subjective tastes
      * There are things people enjoy a lot that they know are not really very good quality
      * To take an extreme, but illustrative example, a Zoophile might enjoy fucking sheep, but is unlikely to expect to profit from investing positively in an item representing this act, since he is unlikely to consider it an objectively good thing; that is, he will probably be aware that his passion is not shared by many other people and that, thus, not many others are likely to invest in the item, making it unprofitable for him to do so.
      * On the other hand, someone might think the iPhone is objectively a good quality product, but may not like it for himself
        + Such a person may invest in the iPhone even though he does not value it personally
        + I would fit into this category
      * Thus, such an investment market is measuring, not utility, but something resembling objective quality. In fact, perhaps there would be no better way to define objective quality than the output of this market.
    - Similarly for the case of the asteroid trajectory, for all intents and purposes, betting on what other people think other people will predict the trajectory of the asteroid to be would be no different from using all the knowledge at your disposal to predict what the actual trajectory of the asteroid will be.

# Fri 14-Jan-11

* Zutha constitution
  + There is no such thing as a victimless crime
    - Zutha law may not include laws which punish people for harming only themselves
    - Examples of current victimless crimes that would thus not be permissible as crimes by the Zutha constitution:
      * Gambling
        + A gambling addiction may lead a parent to be unable to afford to look after his family
        + I think there is merit in considering it a crime to harm one’s dependents
        + Even so, the punishment should be for harming one’s dependants rather than gambling in and of itself
      * Using drugs
        + If the use of, or addiction to, drugs is responsible for some other crime like violence or theft, then these crimes should be punished in their own right
        + Being punished for violence done while on drugs is thus a deterrent against, and punishment for, the harm done to society, rather than a deterrent against a behaviour that society considers immoral in some abstract way not directly connected with its actual cost to society
        + It could be argued that actually seeing someone dangerously overdosed on a harmful drug could be stressful to some people and that drug overdose should be punished for its harmful effects on observers
        + The same argument could be made for making it a crime to insult another person or to “be a jerk”
        + In fact, these minor “crimes” are already punished by social norms and I see no reason to believe a central powerful entity would do a better job at appropriately punishing such crimes according to the degree of their harmful effects
        + Actually the concept of down-voting (or investing negatively in) another person could play a useful role in crowdsourcing punishment for such minor crimes
    - Marrying and having kids could be considered as accepting a contract to contribute to the welfare of those people
      * In the case of kids, a parent is responsible for their existence; the parent has caused the addition of another agent in society which has rights, and thus should be partly responsible for having those rights met
      * A very poor person may be unable to adequately support the needs of their child
      * Should having a child one cannot support be a crime?
      * Actually I think it is conceivable that such a law could be a net benefit to society, though it would clearly be unpalatable to a lot of people
      * A more palatable model for many people might be to require a certain proportion of one’s income and assets to be invested in one’s children
      * If a wealthy person squandered all their wealth (on gambling, say) such that they could no longer afford to support their children, this would be a crime, but there would be nothing specific about gambling that makes such behaviour a crime
      * Spending all one’s money on shopping for oneself such that no money is left to support one’s family would be considered a crime in exactly the same way
  + No agent can ever be given any amount of power through Zutha which is not ultimately dependent on the support of the community
    - Example: a military leader over Zutha’s armed forces could not be granted office for a fixed term that cannot be shortened by community action through the investment process
      * A military leader might be assigned command of Zutha’s armies for a maximum period of, say 3 years, but during that time he would be liable to be voted out given enough public sentiment against him
      * The amount of sentiment against him that is required to unseat him would be defined as the minimum worth, of the action item to unseat him, as a percentage of all zuth in the economy
  + The maximum percentage of zuth in the economy that can be required for an action to take place should be limited to something substantially below 100%
    - This maximum should also be determined by the zutha community
    - The absolute limits of this maximum should be determined in the constitution
    - The absolute limits of this maximum should be a wide range, but one that excludes the unrealistic extremes
    - 1% to 25% might be a realistic range
  + The positive margin on an action item, as a percentage of zuth in the economy, required to change the constitution needs to be specified by the community as well
  + There should be different levels of the constitution, requiring different margins as a percentage of the economy to change
    - Actually there need be no distinction between low level rules and the constitution, just a continuous range of margins required for modification
    - Every type of action should have an associated maximum and minimum allowed margin requirement
    - Actually every type of action should have a default margin requirement
    - One type of action, A, would be changing the margin requirement of a particular type of action, B
    - Action A would have a higher margin requirement than action B, because it would allow changing the margin requirement of B
  + It may be that margin requirements are not best measured in percentage of the economy
    - As the economy grows, the number of items to invest in and the degree of distraction of every user is increased
    - Thus it would probably be much easier to get 1% of all zuth in the economy to be invested in a decision when Zutha is small, than when it is huge
  + Defining margins in absolute terms would be risky as well
    - It will become a lot easier for an item to get a worth of 1000 zuth when the total amount of zuth in the site is 1,000,000,000 than when it is 5000
* Will the incentives in Zutha prediction markets lead to relative market prices of candidates that reflect the aggregate opinion about the actual relative probabilities of each candidate?
  + I thought about this last night and came to the (wrong, I now realise) conclusion that the relative values of the candidates would not reflect 1 to 1 users’ beliefs about the relative probabilities of the candidates
  + I made a mathematical mistake. I calculated expected value of an investment by multiplying the probability (as believed by the investor) by the *profit* the user would make if successful
  + The correct way to calculate expected value is to multiply probability by the total amount of money returned if successful
  + To calculate expected profit, the amount invested is subtracted from the expected eventual value of the investment
  + A worked example
    - In a prediction market about the event A, the current state of the market is worth(A)=80; worth(A’)=20
    - User X believes P(A)=75%, P(A’)=25%
    - User X wants to invest 1 unit in the prediction market
    - For simplicity we will ignore the effect of this 1 added unit on the market value (worth) of the candidates
    - On expiry of the market, the expected value of X’s investment for each candidate is:
      * E(1 unit in A) = 0.75 \* 100/80 = 0.9375
      * E(1 unit in A’) = 0.25 \* 100/20 = 1.25
    - Thus user X will invest in A’, bringing the relative market values of the items closer to X’s assessment of their relative probabilities
    - If the market values of the candidates reach worth(A)=75; worth(A’)=25, user X should be indifferent between investing in either one
    - In this case the expected value (on market expiry) of X’s investment for each candidate is:
      * E(1 unit in A) = 0.75 \* 100/75 = 1
      * E(1 unit in A’) = 0.25 \* 100/25 = 1
    - If the investor’s effect on the relative values of the items is taken into account, the best course of action for investing in an item that already matches the investor’s beliefs about the relative probabilities of the candidates, is to maintain the same ratio of candidate values
      * This requires, in this case, that investor X invest 0.75 in A and 0.25 in A’
      * Elegantly, this ratio is exactly X’s beliefs about the relative probabilities of A and A’
      * E(0.75 in A, 0.25 in A’) = 0.75 \* 101/75.75\*0.75 + 0.25 \* 101/25.25\*0.25 = 1
      * Whereas, to be exact, E(1 unit in A) = 0.75 \* 101/76 = 0.9967
    - Unfortunately, there is not really any incentive to invest in a correctly priced (by your beliefs) item at all, since the expected profit is exactly 0 – and less if winnings are taxed
  + If each independent user has the incentive of moving the relative market values of candidates closer to his own beliefs about their relative probability, then the aggregate effect of all users trying to do this will be to move the relative market values of the candidates to their relative probabilities as found by aggregating all investors beliefs about their probabilities.
* It would be good if the incentives for prediction market investing still worked for the first few investors
  + There’s something called the “thin market” problem which I think is relevant here
  + The very first investor in a prediction market as currently defined has no incentive to provide information about his knowledge of the relative probabilities of the candidates
  + The only incentive to invest is based on the belief that others will invest after him so he can sell his investment (before expiry of the market) for greater value later
  + The same incentives as are provided in a dense market could be provided to a thin market by providing a fixed prize that gets added to the pool of winnings
  + For the first investor, the prospect of winning this prize provides incentives to according to his beliefs about the relative probabilities of candidates
  + Let’s say the sponsor of a prediction market on a question of interest to him (the likelihood of event A) adds $1000 to the pot to start off the betting
  + The very first investor has no incentive to invest any more than an arbitrarily small amount in order to maximize his expected return at this point
  + Of course, the first investor will know that others will invest after him and quickly dilute a small investment, so he might as well invest a reasonable amount right from the start
  + Let’s say the first investor invests $10. How should he divide it among the candidates assuming he believes event A to have P(A)=0.75 ?
    - E($10 in A) = 0.75 \* 1010 = $757.5
    - E($7.5 in A, $2.5 in A’) = 0.75 \* 1010/7.5\*7.5 + 0.25 \* 1010/2.5\*2.5 = $1010
    - E($10 in A’) = 0.25 \* 1010 = $252.5
    - It looks like it doesn’t matter how he divides his investment as long as he invests some money in each
  + Let’s assume for the moment that investor 1 invests $10 in A
  + What about the 2nd investor? We’ll assume the same probability expectations for now.
    - E($5 in A, $5 in A’) = 0.75\*1020/15\*5 + 0.25\*1020/5\*5 = $510
    - I think this is the best strategy for investor 2
    - No, checking on WolframAlpha, there is no global maximum; 2nd investor’s incentive is simply to invest as much as possible in A
    - Now that I think about it, this makes sense since investor 2 is purely competing with investor 1 for the dominant share of A
  + Now let’s assume investor 1 invests $9 in A and $1 in A’
    - E(investor 2 invests $x in A, 10-x in A’) = 0.75\*1020/(9+x)\*x + 0.25\*1020/(1+10-x)\*(10-x)
    - Maximum is 530 at x=7.772
    - Huh!
    - That leads to a ratio of candidate values of (9+7.772)/(1+10-7.772) = 5.2
    - But the ratio should be 3 – i.e. 0.75/0.15…
  + What is expected return for investor 1 now?
    - E(prize for investor 1) = 0.75\*1020/(9+7.772)\*9 + 0.25\*1020/(1+10-7.772)\*1 = $489.5
* Let’s check whether I was right about the incentives without the fixed addition to winnings of $1000
  + An example
    - User X believes probabilities P(A)=0.75, P(A’)=0.25
    - Candidate values are: worth(A)=75, worth(A’) = 25
    - User X wants to invest $1
    - E(x in A, 1-x in A’) = 0.75\*101/(75+x)\*x + 0.25\*101/(25+1-x)\*(1-x)
    - Maximized at x=0.75
    - (75+0.75)/(25+1-0.75) = 0.75/0.15 = 3
    - Good!
  + Let’s check another example
    - User X believes probabilities P(A)=0.75, P(A’)=0.25
    - Candidate values are: worth(A)=80, worth(A’) = 20
    - User X wants to invest $20
    - E(x in A, 20-x in A’) = 0.75\*120/(80+x)\*x + 0.25\*120/(20+20-x)\*(20-x)
    - Maximum: 20.36 at x=13.12
    - (80+13.12)/(20+20-13.12)=3.46
    - Damn!
  + Another example
    - User X believes probabilities P(A)=0.75, P(A’)=0.25
    - Candidate values are: worth(A)=80, worth(A’) = 20
    - User X wants to invest $1
    - E(x in A, 1-x in A’) = 0.75\*101/(80+x)\*x + 0.25\*101/(20+1-x)\*(1-x)
    - Maximum at x=0
    - (80+0)/(20+1)=3.81
    - I think I’m beginning to see why the probabilities and market values don’t match up immediately after a single investment
    - In this case it’s obvious that user X isn’t investing enough funds to make up the difference between the actual market values and his believed probabilities
  + What about if user X is investing a huge amount?
    - User X believes probabilities P(A)=0.75, P(A’)=0.25
    - Candidate values are: worth(A)=80, worth(A’) = 20
    - User X wants to invest $200
    - E(x in A, 200-x in A’) = 0.75\*300/(80+x)\*x + 0.25\*300/(20+200-x)\*(200-x)
    - Maximum at x=152.797
    - Apparent probability ratio = (80+152.797)/(20+200-152.797)=3.46
    - The amount X is investing makes no difference – this is the ratio he is targeting: 3.46… not 3. Damn!
  + Interestingly, 3.46 is between the ratio predicted by the other investors and the ratio predicted by X
  + I’m hypothesizing that another independent investor predicting 75/25 will bring the ratio even closer to 3
    - User Y believes probabilities P(A)=0.75, P(A’)=0.25
    - Candidate values are: worth(A)=93, worth(A’) = 27
    - User X wants to invest $20
    - E(x in A, 20-x in A’) = 0.75\*140/(93+x)\*x + 0.25\*140/(27+20-x)\*(20-x)
    - Maximum: 20.1036 at x=13.78
    - (93+13.78)/(27+20-13.78)=3.21
    - Perfect!
  + These mechanics will actually be beneficial to the accuracy of the equilibrium, I think, because they cause the equilibrium to be found more democratically
    - One huge investor – even one who invests a large amount compared to the total amount of money in play – will only skew the ratio partially towards his predicted ratio
    - Assuming the huge investor is playing for profit rather than purely to skew the results, he is likely to be betting close to the consensus prediction anyway
    - If a huge investor just wants to skew the results, he can just invest as much as he wants to push the consensus prediction arbitrarily close to what he desires – profit incentives have nothing to do with this situation
    - The good thing about the way the incentives work is that if two powerful investors oppose each other, they will find an equilibrium point somewhere in the middle, whereas if the most profitable way to invest was to push the probability distribution of candidates as close as possible to your own prediction, two opposing investors would be constantly pushing the consensus distribution back and forth between their two private distributions
  + Let’s test whether two opposing investors reach an equilibrium
    - Investor X has distribution P(A)=0.60, P(A’)=0.40
      * preferred A/A’ = 1.5
    - Investor Y has distribution P(A)=0.70, P(A’)=0.30
      * preferred A/A’ = 2.33
    - X invests 6 in A and 4 in A’
    - E(Y invests x in A, 10-x in A’) = 0.7\*20/(6+x)\*x + 0.3\*20/(4+10-x)\*(10-x)
      * Maximizing x = 7.03
      * A/A’ = (6+7.03)/(4+10-7.03) = 1.87
    - E(X invests x in A, 10-x in A’) = 0.6\*30/(13.03+x)\*(6+x) + 0.4\*30/(6.97+10-x)\*(4+10-x)
      * Maximizing x = 6.57
      * A/A’ = (13.03+6.57)/( 6.97+10-6.57) = 1.88
    - E(Y invests x in A, 10-x in A’) = 0.7\*40/(19.6+x)\*(7.03+x) + 0.3\*40/(10.4+10-x)\*(2.97+10-x)
      * Maximum = 20.1147, at x = 7.01
      * A = 19.6+7.01 = 26.61
      * A’ = 10.4+10-7.01 = 13.39
      * A/A’ = 1.99
    - E(X invests x in A, 10-x in A’) = 0.6\*50/(26.61+x)\*(6+6.57+x) + 0.4\*50/(13.39+10-x)\*(4+10-6.57+10-x)
      * Maximizing x = 6.03
      * A = 26.61+6.03 = 32.64
      * A’ = 13.39+10-6.03 = 17.36
      * A/A’ = 1.88
    - E(Y invests x in A, 10-x in A’) = 0.7\*60/(32.64+x)\*(7.03+7.01+x) + 0.3\*60/(17.36+10-x)\*(2.97+10-7.01+10-x)
      * Maximizing x = 7.03
      * A = 32.64+7.03 = 39.67
      * A’ = 17.36+10-7.03 = 20.33
      * A/A’ = 1.95
    - I think I’m going to need excel to follow this pattern
    - It looks like there’s going to be a fair bit of oscillation after all
    - I can’t find a way to calculate the local maximum with excel, so I’m giving up on this for now
    - I’ll use Mathematica or Matlab or something some time to plot this interaction between 2 investors
    - It could be interesting to expand the simulation to many investors with different prediction distributions
  + If an investor is small enough that he is not going to significantly influence the ratio A/A’, his best bet is to invest on whichever side he believes is undervalued
    - I think it is clear that the mechanics of this investment mechanism align incentives most elegantly when each investor is an insignificant player
    - Roughly speaking, though, I think it will still yield useful results even with only a very small number of players
* A decentralized society based on distributed intelligence would be a potentially very potent adversary, I think
  + Such a society is very unlikely to initiate conflict because it is driven by what is in the best interests of every member, and war is not in the best interests of many members of a society
  + If such a society (let’s call them Zutha societies from now on) were attacked by some powerful entity, I believe it would hold up disproportionately well in comparison to its man-power, military strength and other tangible resources
  + The prediction markets and distributed decision making mechanisms of Zutha would enable a vast amount of very useful information to be generated for both sides of a conflict
  + The members of a Zutha society may be more adept at using the information generated by Zutha, and this may give them a competitive advantage over a centrally controlled attacking entity, but there would still be huge benefits to secrecy in some aspects of waging a war
  + It would be possible for a society running on a Zutha platform to achieve some degree of secrecy, but it would struggle to do so reliably, I think
  + The community would be able to select a team of war leaders who henceforth, until the war is over, have the power to make decisions among themselves only about how to use some of Zutha’s resources
  + There would be a difficult balance to find regarding what and how much information should be disclosed publically by the war leaders so they can be adequately judged and so that the community can participate to greatest effect in decision making.
    - If too little information was disclosed it would give the war leaders too much power to turn events to their own private benefit and away from the benefit of the community as a whole
    - If too much information was disclosed it would sacrifice secrecy, which is a valuable asset in war
    - If the group of war leaders was chosen well enough, they should keep each other in check
    - The main risk of a small group being in power comes when they begin to approach a single entity rather than multiple independent agents
  + The most difficult part of maintaining secrecy would be to ensure that the war leaders themselves are not spies for the enemy
    - Given the degree of anonymity potentially permitted by a Zutha society, this could be very difficult
    - Then again, it might eventually be that identity is even more secure in Zutha societies of the future than it is now.
    - It might eventually be required that every “user” be associated with biometric identification information
    - At least, it would be possible to know that a user is associated strongly with a biological identity and war leaders can be chosen from among this group
    - A Zutha society is likely to be able to know a great deal about the secrets of its enemies because many people from all walks of life may be attracted to the decentralized society philosophy.
    - Then again, the fact that secrets which are given to Zutha become public knowledge may make it very difficult for a spy to safely leak information because his superiors would immediately know what had been leaked and may be able to trace the leak to a small number of candidates including him.
    - So it is uncertain whether Zutha would be able to know reliably such things as whether one of its biologically identified users is also working for the Chinese Government or a powerful company’s private intelligence service.
    - I wonder how reliably it would be possible to judge a user’s character and loyalties just from his history of interactions through Zutha
    - Given how successful spies have historically been at fooling people about their true character and loyalties, I’m not sure there is much cause for optimism here.
    - Then again, I think becoming successful on Zutha would expose far more of one’s true self than rising up the ranks of a bureaucratic government intelligence agency.
    - Gaining the reputation required to be voted a war leader would require a kind of unique and impressive personality that I think spies for a government agency would find it very difficult to fake; in fact, if they could fake it realistically, they would probably be at risk of turning to the dark side and switching loyalties to the Zutha community
    - Unlike a spy in a bureaucracy who can easily study the norms of behaviour of the people he is trying to integrate with, becoming successful on Zutha would require individuality; one would need to add a substantial amount of new input which is appreciated by the community.
    - A spy who is a committed enemy of Zutha would be unlikely to be able to contribute new insights that the community of Zutha users would consider valuable and in line with their philosophy, which, though very diverse, would still be in line with such values as liberty and improving the quality of life for all.
    - The power of the wisdom of crowds in identifying the subtle queues that would identify the unauthenticity of a spy is likely to be far greater than a bureaucratic filtering process.
    - Once Zutha societies have been around long enough, a larger proportion of users’ lives will be known by their interaction with other Zutha users
    - For Zutha users who have a public history of interactions with other Zutha users since early childhood, it should be relatively easy for the community to distinguish friend from foe
  + During the strategic chess-match of a war, the Zutha community would have no constraint upon it about speculating on the decisions – and their likely consequences – of both their own and their enemies’ commanders
    - The war leaders anointed by the Zutha community under attack would be well versed in using such information to their advantage
    - The game would be of a different kind than warfare has been throughout history
    - Historically, the game of outwitting one’s opponent was based on anticipating him and making the best possible move in response that the enemy cannot anticipate of you.
    - With accurate prediction markets and open source war advisors that are far more effective than any individual or small team of war experts, the game would be all about execution, because the best possible moves of both sides are likely already laid out in public view for many moves ahead.
    - There will surely be a degree to which both sides have secrets not known to the prediction markets that cause their decisions to deviate from the suggestions of the market, yet I think the degree to which this is so will be far less than it is now.
    - Individual agents on Zutha’s side may communicate their discovered secrets only to the war leaders and not to each other or the global community, yet all of those agents still know that information, and if they participate in the prediction markets, they will – perhaps even unknowingly – contribute their knowledge to the pool.
    - In this way, the prediction markets may still incorporate more knowledge than any war commanders can gather and use from their spy networks – perhaps even including most of the knowledge brought in by the spy networks.
  + It is possible that a secretive war leadership team would not even be necessary
    - The decision-making ability of the whole of humanity, if harnessed effectively enough, may potentially be so far superior to the capacities of a centralized authority that a Zutha society may not even need to keep any secrets to defeat it.
    - The Zutha society would simply act exactly, in every minute detail, according to the recommendations of the market
    - Many decisions would be made that no individual would be able to understand the wisdom of, let alone the commanders of the centrally controlled enemy.
    - Perhaps the only way to have a chance of defeating such an intelligent entity would be to base your decisions on its advice as well
    - The Zutha community would be gathering and broadcasting just as much useful information about what the enemy should do as what the Zutha community itself should do to win the war
    - Unfortunately for the enemy, I don’t think a centralized command would be capable of handing over their control to the distributed intelligence that exemplifies their opponent.
    - The enemy’s commanders would surely try to glean as much useful information from Zutha and similar tools of their own as they can, but the whole nature of a central authority – the conflicting incentives; the bureaucracy – would make them incapable of using that information effectively enough to defeat the ultimate masters of using that information: the producers of it.
  + What would a war between two Zutha-like distributed societies look like?
    - I don’t’ think it would, or even could, happen
    - Consider that no democracies have ever fought a war with each other (I read that claim somewhere at least)
    - If even democracies, flawed in many ways as they are including their propensity to drift towards inefficient central control, are intensely averse to going to war with one another, then how much more averse to war would be a society that is driven by public opinion and the common good to an axiomatic extent?
    - War is so clearly harmful to the societies of both sides, and peace and trade so clearly beneficial to both sides, that I simply cannot conceive of a scenario in which the incentives would be contrived to have two Zutha societies vote to go to war with one another.
    - Actually I can imagine what it would look like for two Zutha societies to go to war with each other: it would look like two teenage brothers bickering, and their whole extended family would step in to stop them, or, more appropriately would combine their exceptional parenting skills to teach the boys the wisdom to stop wanting to fight each other.
    - Any subset of the global Zutha community disposed to mutually harmful bickering are unlikely to include the most intelligent and powerful members of the community; these would be the members with global respect and an interest in global prosperity and would be uniquely (by today’s standards) positioned to pursue these interests. They would be the leaders of the vast intellectual and material might of the distributed Zutha community.
    - Any destructive behaviour arising from such powerful Zutha leaders would quickly suck away their power, but while they remain immensely respected, the resources at their disposal to increase that respect would be immense.
    - The power of any subsets of the Zutha community with destructive ambitions would be vastly outmatched by the resources at the disposal of the remaining global community whose interests are in global peace and mutual prosperity.
    - With those powerful leaders to catalyse the resources of the masses into action, unruly behaviour would be quickly calmed, or, if necessary, stamped out.
* The margin of item worth required on an action item for it to succeed should be measured in units of fractions of the total amount of reputation in the site across all users
  + “Net reputation” is a better “inflation index” than total currency because it represents the purchasing power of the community
  + The sum of all users’ reputations is exactly equal to the theoretical maximum worth a single item can have, because every user is limited in the maximum amount they can invest in an item, to the level of their reputation
  + Reputation will be roughly proportional to the square root of the total amount of currency in the site
    - Not all items will have owners to which their worth can contribute reputation, so net reputation will be less than the square root of total currency
  + The average user will have access to much more currency than his reputation, allowing him to invest the maximum allowed amount (as restricted by his reputation) in many different items at once
  + As net reputation rises, the practical ability of the community to invest a given amount in an item rises
  + The margin of zuth required to activate an action item should thus increase as net reputation (and the economy’s “purchasing power”) increases
  + The margin of zuth required to perform an action should be specified by users in terms of Zuth, since that will be the measure of value most familiar to people
  + As net reputation rises, these margins, initially specified in absolute Zuth terms, should be increased automatically to keep up with “inflation”
  + Whever the market wants to change the value of a particular margin, they may do so in absolute Zuth terms and the new value will be mapped to units of percentage of net reputation in the background.
* Links to item versions
  + Some associations should form a link between a particular version of an item and other items
  + E.g. a comment about a fact in a document which is subsequently fixed
    - The comment should link to the old version of the document prior to the fix, since the comment does not really apply to the new version of the document
  + There will be a need for a deeper level id system to facilitate associations with old versions of items
  + An alternative would be for the association to store the version number as a reified occurrence
    - The older version could then be extracted from data stored in a single item (rather than having separate items for each version)
    - Older versions could be calculated using the data stored in event items for the relevant item

# Mon 17-Jan-11

* How to decentralize Zutha’s finances
  + It may be a long time before eCash becomes a flexible and widely enough used technology for Zutha to use it to control its finances without any single point of control
  + It may, however, be possible to use existing legal electronic currency transfer technologies to support a decentralized framework of finances
  + Though it would require placing more trust in the hands of Zutha’s financial sponsors than the eCash system would have required, it should still be practical to give control of small fractions of Zutha’s wealth to many different financial sponsors, all of whom agree to do what Zutha tells them to do with the money it entrusts to them
  + Any kind of entity that is legally allowed to manage money could theoretically act as a financial sponsor – actually “financial manager” is probably a better term.
    - Big companies
    - Small companies, including companies owned by one or two people even
    - Banks
    - Individual people
    - Communities of people
  + In every case, Zutha would need to be confident that the entity’s reputation mattered enough to it for it to be trusted with some of Zutha’s money
  + Let’s say a large company, say Google, volunteered to support Zutha by taking control of some of its finances
    - When a user X pays money to Zutha, say to invest in an item, one of Zutha’s financial managers is picked at random
    - Let’s say that Google is picked for this transaction by user X
    - X is instructed by Zutha to pay his money through an electronic system (say Paypal) to an account owned by Google
    - X is also given a number Q to send to Google along with his money so that Google can tell Zutha it has received this particular transaction
    - Once Google has received the money it sends a message back to Zutha saying transaction Q completed successfully and Google has received the correct amount of money for that transaction
    - Zutha will now update its database recording how much money Google holds in its name
    - Later on, a user Y wants to withdraw money he has earned on Zutha
    - Zutha has a publically visible record in its database identifying the electronic account Y wants funds to be deposited into
    - Zutha randomly chooses one of its financial managers (say Google) and instructs it to send the appropriate amount of funds to Y’s account
    - The institution that manages Y’s account would then send a signed message back Google indicating the transaction has completed successfully
    - Google would forward this signed message on to Zutha so it knows the transaction it requested has been performed
    - If Google was not able to produce such a signed message proving that it completed the transaction, Zutha would publish the failure and ask the public to reprimand Google accordingly for breaking its deal with Zutha
    - A company like Google would place a high stake on its reputation and so this means of deterring cheating would probably be sufficient
  + This system would be less frightening to governments than an anonymous eCash system, I think
    - No transactions would be anonymous so governments could still use their standard techniques to prevent money laundering
    - For example, Google could be forced by a government to check for suspicious transactions among the transactions it performs for Zutha
    - If a terrorist was asking for money out of his Zutha account and Google was the financial manager chosen, it might detect that the receiving account is marked as suspicious and prevent the transaction
    - Google will also keep a record of all its transactions which can be examined by it, or directly by government institutions to search for suspicious behaviour indicating money laundering
    - If Google refused to complete a transaction for Zutha due to suspicious activity, it would notify Zutha of this, and Zutha would notify the Zutha community
    - It would be up to the Zutha community to establish whether it thought Google’s actions were fair
    - If, for example, Google refused to send money to an organisation like Wikileaks on behalf of Zutha, because it regarded Wikileaks as a terrorist organisation, the Zutha community might decide that this does not constitute a fair withholding of funds, and would invest negatively in Google’s reputation in response.
    - In a situation like this, there could of course be much disagreement in the Zutha community, leading to large amounts invested both positively and negatively in Google’s reputation in response to it withholding funds from Wikileaks.
    - Google would have to take into account the likely effect of its actions on its reputation in Zutha, when it decides whether or not to withhold funds from Wikileaks
    - Of course, Zutha also has the option to ask another of its financial managers to send the money to Wikileaks
    - If a trusted institution like Google refused to give money to a particular Zutha user on the grounds that it is owned by a criminal, then Zutha should take this seriously and put to the Zutha community the question of whether to persevere in getting the requested money to the alleged criminal.
    - If the community votes strongly enough to send the money to the alleged criminal, then Zutha will continue instructing one financial managers at a time to send the money until one accepts.
  + Entities that offer to manage some of Zutha’s money should be allowed to keep the interest on holding that money
    - This would be an effective incentive for them to initially offer to partner with Zutha in this regard, and to remain loyal
    - In most cases, it should be more profitable for a financial manager to continue managing Zutha’s funds and make profit on investing those funds, than to steal the funds and have the source of funds dry up as well as its reputation trashed
* Users should be able to set preferences to filter messages sent directly to them by the reputation of the sender
  + This would facilitate high level users sending out bulk messages to a group of users they think should, or would want to, know about some piece of information
  + High reputation users would have enough at stake to create a strong disincentive to spam
  + High reputation users that spammed would quickly lose a lot of reputation as the many users they spam collectively downvote them by investing negatively in them
* A case study: how would ambulance service be provided in a Zutha society?
  + One strong argument, even from a libertarian point of view, for a policy that ambulances collect any severely injured person without stopping to check their paperwork is that the time taken to check said paperwork would cost many lives
  + The other main reason, and the one many people would make the only one, is that society values the right of any individual to have his life saved by the state regardless of whether he possesses the resources to pay for the interventions required to do so
  + The first reason for ambulances’ policy of rescuing and asking questions about people’s wealth and health insurance status later, may be made obsolete by sufficiently advanced technology that allows instant identification of a victim’s financial means and/or health insurance status
  + The second reason is the kind of subject that would make a quintessential topic of debate between pro and anti-libertarians
  + Rather than deciding yet what I think is right or wrong on this subject, I’ll examine what I think would be a likely state of affairs in a Zutha society
  + I think there are enough people who would value a policy of not letting poor people die within easy access of life-saving interventions, even if they can’t afford them, that the Zutha community would invest both real money and Zuth into the cause of providing free health care to people most in need.
  + The investment of Zuth would draw real money from Zutha’s pool of income to the cause
  + There might arise many independent organisations who actually provide the service; they would be responsible for filtering those in sufficient need from those who can afford to pay their health costs themselves
  + As with the provision of public toilets that I discussed earlier, the allocation of resources for the provision of this public good would be controlled by investment by the Zutha community
  + Those free health-care providers who receive the highest praise from society would receive more funds to do their work
  + It is of course impossible to be sure without testing this theory (which I thoroughly intend to), but my intuition says that in a community built on reputation the way Zutha will be, with much higher median wealth, and without any enforced community contribution through taxes, social norms and individuals’ desire to give and be seen giving, would lead to substantially more resources being donated voluntarily from all classes of society.
  + With the additional efficiency achieved by the absence of government bureaucracy, I feel confident that such a scheme relying only on philanthropy would result in far greater public wellbeing than is achieved by government funded (borrowed really) social welfare.
  + As for how this voluntarily provided social security would apply to ambulance service; organisations who manage the philanthropic service to provide free health care for the people most in need may collectively give a guarantee to the companies providing ambulance service, that they will cover the costs of the ambulance for picking up any person in a severe medical conditional.
  + It should not take long in a technologically advanced society to figure out, first, whether the patient needs the support of the philanthropic health care services, and, second, whether, if the patient or his family can afford it, they will choose to spend the money on whatever medical interventions are required or recommended.
  + Let’s say a child has had his eye poked out and his parents are wealthy enough to pay for an expensive operation to replace the child’s eye with an artificial one (assuming an advanced society here).
    - The parents may chose not to spend the money required for the more advanced operation, and opt instead for a cheaper operation that will leave the child with only one eye for the rest of his life.
    - The philanthropic organisations might have policies that dictate that they would not, by default, provide funding in such situations because the patient is not in sufficient financial need, and the organisation would not be able to afford a reputation for being too lenient for fear of being sucked in to paying for rich people to do expensive operations at the expense of saving poor people’s lives.
    - It may still be that the patient’s wider friends and family, or even strangers who become aware of the situation, volunteer money for the operation because they feel a particular connection to the patient.
  + This dynamic seems to be quite an appealing dynamic to me, though I’m sure there are many situations I’ve not thought of that would lead to unpleasant scenarios.
  + Given how horribly medical resources are allocated in rich countries when controlled by bureaucracy, though, I’m not that concerned that a more libertarian and philanthropic (driven by a reputation based society) based system would turn out any worse.
  + The most significant uncertainty in this vision, I think, is how effective a platform like Zutha will be at promoting philanthropy through reputation based incentives
* People could offer to provide services for free, particularly when they are first starting off their careers, and earn reputation and possibly money as well, after they’ve provided the service
  + Recipients of such free service would have an incentive to invest both real money and Zuth in the service provider if they think the provider deserves it, because they will expect others to invest in said provider after them, and they can thus make profit by being an early investor
  + As with investing in any other item, the service recipient here is putting their own wealth on the line to advertise the quality of the service, and they are rewarded for doing so if they are right enough about the worth (as perceived by others) of the service they invest in.

# Tue 18-Jan-11

* What currency should real money be measured in in Zutha?
  + One possibility is to define the value of the currency as a weighted average of many other world currencies
    - This would be problematic, I think
    - If one currency went through extreme inflation, it might still significantly alter the relative value of the Zutha currency
    - Hyper-inflation can cause increases in price level of many orders of magnitude, which would have an effect on the average of many currencies even if the hyper-inflating one was very diluted
    - It also would just seem strange to have a weighted average of currencies that are already orders of magnitude different from each other in their fundamental units
  + Another possibility is to define the value of a given amount of Zutha dollars as a bundle of goods
    - This could be possible, and useful, even if there is no system in place to actually exchange Zutha dollars for the actual bundle of goods
    - One challenge with defining value in terms of a bundle is that the increasing rate of technological progress is likely going to increase the instability of the relative price of any given good
    - Ideally it would be possible to slowly add new goods to the bundle in order to keep it relevant with the times
    - This would be practical, I think
    - Small amounts of new kinds of goods could be slowly added to the bundle, increasing the value of the currency accordingly
    - This would offset to devaluing of the currency due to its bundle of goods becoming less relatively valuable as technology leaves them behind
  + It would also be possible to define Zutha dollars as a bundle of other currencies
    - This is actually what I think the idea of a weighted average of currencies would really come down to
    - I can’t really envision what defining Zutha dollars as a weighted average of other currencies would mean; the closest thing would be to define a Zutha dollar as a particular amount of US dollars, Australian Dollars, Pounds, Euros etc…
    - The good thing about defining Zutha dollars as a bundle of other currencies would be that it could easily be exchanged into any currency at its defined value simply by using the current exchange rates
    - There wouldn’t be the same difficulties associated with backing the currency with a huge and complex array of goods
    - Let’s say we define 100 Zutha dollars as 20 Euros, 25 US dollars, 25 Australian dollars, 10 Pounds, 10 Japanese Yen and 10 Chinese yuan
    - If someone paid, say, $10 USD to Zutha, and the financial manager chosen was Google (to keep our old example), then Google would receive $10 USD, but would be expected to hold the equivalent value in Zutha dollars to return to Zutha later
    - It would be up to Google to transparently convert the money coming in from Zutha into the appropriate bundle of currencies in order to protect themselves from risks associated with changes in the relative value of currencies
    - Of course, a financial manager may choose to keep any bundle of currencies it liked, but its contract with Zutha (enforced perhaps only by reputation) would be to be able to supply in terms of Zutha dollars, whatever comes in in terms of Zutha dollars
    - As with the bundle of goods, ideally it would be possible to slowly change to bundle of currencies a Zutha dollar is defined as
    - This is because the relative value, stability and global significance of currencies is likely to vary over time and so the best possible bundle will not be static
    - It should be possible to trade one currency for another in the bundle at the current exchange rates, thus modifying the distribution of the bundle without creating significant arbitrage opportunities
    - As with every other decision in Zutha, the investment market would have to be used to decide the distribution of the bundle
    - Zutha users would vote on when they think the distribution should change and what it should change to
    - The distribution could be chosen by allowing users to invest in candidate items representing any currency in the world
    - Users would invest in each candidate in proportion to how big a part of the bundle they think it should be
    - When the decision is made to move to the new bundle, the existing bundle would be exchanged for the new bundle at current market prices
    - Example: a bundle is made of up 50% USD and 50% Euros
      * The distribution is changed overnight to 40% USD and 60% Euros
      * At the moment of the change in bundle, a Zutha dollar that used to consist of 50% USD and 50% Euros would now be worth an amount consisting of 40% USD and 60% Euros which is worth the same value at current exchange rates
      * Everywhere around the world, financial institutions claiming to hold its customer’s accounts in units of Zutha dollars should trade their existing bundle of currencies for the new bundle in order to protect themselves from an unfavourable change in exchange rates which leaves their old bundle worth less than their advertised holdings of Zutha dollars
    - On the whole I think this is the best strategy
    - The bundle of currencies chosen can start very simple; in fact it can even start as a bundle consisting of a single currency, say US dollars
    - Later on, the bundle can be slowly modified to include more and more currencies in order to increase stability at the expense of complexity for financial managers
    - Early on, it will be more important to make it an easy process to contribute to managing Zutha’s finances so that it is possible to get enough entities to volunteer
    - Later, once Zutha is bigger and being one of its financial managers is substantially more profitable, it will be possible to ask more of them

# Wed 19-Jan-11

* Inspector service.
  + Companies are incentivised to let them in if they have a high enough reputation because not letting them in will lead to suspicion and a negative effect on their reputation.
  + If the company is behaving in an unpopular way they have to weigh up the consequences of exposure versus suspicion against them. Either way, they are punished for their misconduct by reduced sales as people in a reputation based society will pay more attention to reputation.
  + Other companies that deal with them will be similarly punished for dealing unjustifiably with irreputables.
  + This reputation based policing will require a very high level of openness.
  + If a company's openness about transactions and relationships is considered a significant enough factor in determining their reputation then openness could be self-enforcing.
* Land ownership.
  + Should zutha claim ownership of any land where zutha societies establish themselves?
  + I can't think of a sensible or ethical way to do this. What's important is the existence of property rights, not who starts with them. If an agent (person/company/group) finds empty land, they should be able to claim ownership of it. Later entrepreneurs can purchase that claimed land from them at whatever the market price turns out to be for the undeveloped land. They can develop the land and sell it in pieces for profit.
  + What if there is conflict for land?
  + You should have to actually use a piece of land for a period of time to claim ownership. If two agents both want to use the same piece of unclaimed land they can work out the dispute among themselves. If they fight, so much the worse for both of them. They are better off agreeing on a price for one to pay the other in exchange for them not starting a fight. Emotion might lead to more fighting than would be rational, but as long as they only hurt each other, again, so much the worse for both of them.
  + What about the strong oppressing the weak and stealing their claimed land from them by force?
  + The weak should collectively purchase the services of a private protection agency. The protection agency could offer far more resources for an individual customer under threat than that customer could afford independently.
  + A powerful independent entity could take on a protection agency but that would no longer be the strong versus the weak and the disincentive of huge mutual expenses would reduce the risk of battle and increase the likelihood of financial based conflict resolution.
  + Two weak entities both with protection agency support would be forced to accept the form of resolution desired by their protection agencies which is likely to be financial compromise where one party lets the other have the land in exchange for a payoff.
  + What about invaders taking the land of natives?
  + I think this might be an area zutha would need to police. The zutha community could decide on a case by case basis whether a piece of land should be considered owned by existing natives.
  + Deciding the extent of the territory which the locals deserve to own would be the hard part here, I think
  + Approximating fairness is the best that would be possible I think. Some natives might end up hard done by; some might get more than makes sense because of zutha community generosity.
  + Natives granted property rights by zutha can then chose to sell or rent their land if they want to.
  + What if power brokers among the natives want to sell their land for their own profit at the expense of their own community?
  + Zutha should distribute its grant of ownership across the whole native community according to the judgement of the zutha community. The native community must then join zutha in order to exercise the ownership rights zutha has granted them. They will each be granted some initial reputation within zutha with which to start making collective decisions. Since their land is jointly owned, any revenue earned by selling or renting it is distributed across all the owners (the whole native community).
  + Natives who do not join zutha will not be able to access the wealth earned by their inactive account until they join. The zutha community will have created People items on zutha for as many members of the native community as they can identify. They will not be turned into user accounts until the associated person authenticates herself with zutha as that person.
  + Undoubtedly many native communities will take a long time to adapt to the alien technologies and concepts on which zutha operates. This is OK. They can take their time. But until a critical mass of them are actively participating in zutha they will not be able to make decisions requiring the input of the whole community, like selling or renting their land.
* YouTube owns part of every item if type YouTube video.
  + The community determines this proportion.
  + This way YouTube earns money from the popularity of items it helps distribute to consumers.
* Invent non-fatal weapons that drug targets into unconsciousness.
  + These could be mounted on tiny unmanned aerial vehicles.
  + Whole armies could be disabled remotely using swarms of “knock-out” bots
  + This is really starting to look like the Darknet from Daemon and Freedom TM
  + It’s even likely that computer game experts will be the best candidates for controlling these robotic armies

# Thu 20-Jan-11

* People items on Zutha can be linked to the public profiles of those people by the Zutha community
  + It will not even require direct contact with the person in question in order to give them control of their item on Zutha
  + E.g. a Zutha user could link J.K. Rowling (the item) to J.K. Rowling’s public Facebook account
  + J.K. Rowling can be contacted through Facebook, or by other means, to tell her she has been granted ownership of an item on Zutha – herself
  + J.K. Rowling can log in to Zutha using her Facebook account and immediately have access to the zuth, reputation and real money she has earned in Zutha
  + Person items should be linked to multiple public profiles if possible, for security reasons
  + Let’s say someone has managed to create a Facebook account that claims to be owned by J.K. Rowling, and the account is considered credible by many, but the real owner is not actually J.K. Rowling
    - This is not likely in the case of someone as prominent as J.K. Rowling, but it could easily happen with less well known people
    - If the J.K. Rowling Zutha item was linked only to the counterfeit Facebook account, then that identify thief would immediately have control over J.K. Rowling’s account on Zutha
  + To avoid this, more popular items should require more highly authenticated 3rd party identities, and more of them
  + 3rd party identities should be authenticated by the same means as everything else is done in zutha – by investment of zuth in candidates by the Zutha community
  + If there are disputes about control over a Zutha Person item, high level Zutha users can contact the real life Person directly through reputable institutions like a university, and then vouch for the candidate identity the real person indicates is them
  + So the fake J.K. Rowling might be claiming ownership of J.K. Rowling via the counterfeit Facebook account, but when a high level user contacts J.K. Rowling’s publisher and gets directly in contact with J.K. Rowling (maybe via proxy of the publisher) to ask which identities she wants to use to take control of her Zutha item, she might give an email address and a couple of OpenIDs, perhaps including OpenIDs she just created explicitly for the occasion.
  + In a dispute about identity, both claimants may create Zutha users in order to push their case with the Zutha community
  + So there might be an item called J.K. Rowling (Facebook account) which is owned by the identity thief who owns the counterfeit Facebook account, as well as an item representing J.K. Rowling’s email address, and another representing one of her OpenID accounts
  + These items could be created by the Zutha community and linked with the respective 3rd party identities, or they could be created by the owners of those 3rd party identities when they first log in to Zutha
  + These items representing the 3rd part identities would be turned into User items so that their owners can use them to post comments in Zutha and push their case for ownership of J.K. Rowling
  + Once a set of identity items have been authenticated as identifying a particular item (in this case J.K. Rowling) the identified Person item should be turned into a User item, and the User features of the Identity items should be disabled
  + Identity items should only be allowed to identify a single item
  + In this case, the Identity items representing J.K. Rowling’s email address and her two OpenID accounts will each be linked to the J.K. Rowling Person item
  + Nothing will happen to the Identity item representing the counterfeit Facebook account
  + The owner of the counterfeit Facebook account can continue to use the identity item itself as his User item, or he can create a new Person item with whatever name he chooses as his User item, and link other identity items to it that he owns
* Links to items that represent a web entity (identified by a URL) should have a “web” icon next to the main text of the link which goes directly to the external web address
  + I had previously planned to make the main link go to an external web address for items that are purely representations of links
  + I think it would be better to maintain consistency, though
  + People will get used to needing to use the “web” icon whenever they want to go to an external site
* What should be done about deposits of natural resources that come under the Zutha umbrella?
  + The problem with natural resources is that they aren’t very conducive to local competition
  + The most efficient way to mine natural resources is at large scale; it doesn’t really make sense to have lots of small mines competing against each other for… what? Dominance of the market? Efficiency?
  + Another big problem with natural resources is that you don’t have to be better at producing the product to sell it, because the resource has its own inherent value. If you mine diamonds less efficiently you might make less profit, but it won’t change your ability to sell the diamonds
  + It seems unappealing to let some large corporation commandeer a mineral deposit and reap massive profits from selling the minerals without contributing anything to society except the minerals that anyone could have mined relatively easily.
  + If the profits ended up back in the local community then the whole situation would look far more desirable
  + Zutha could claim ownership over natural resources and keep the revenue made from them
    - This idea sets off alarm bells for me because of the undesirable consequences of putting governments in control of commercial operations
    - Zutha is nothing like a government as it exists today, though, so maybe the disadvantages of “government” control will not apply for it. Maybe.
  + If the mining company was a public company then investors would gain the proceeds of the mine
  + Perhaps Zutha could require that only publically traded companies are allowed to bid to start a resource extraction enterprise on Zutha territory
  + This means people who contribute capital to get a mine underway are the ones that profit from it. This seems reasonable. There needs to be an incentive for investors to inject that initial capital after all.
  + This still leaves the challenge of who should get the resource rich territory in the first place, though
  + I already thought a bit about how land might be allocated without conflict if all interested parties were either powerful or backed by a powerful protection agency
  + If the land is worth an enormous amount, though, the dynamics might be far more difficult
  + What if, as will be likely, many powerful parties are all vying for ownership of an unclaimed piece of resource-rich land?
    - One party will not be able to afford to buy off all the other parties because the maximum one party will be willing to spend (something proportional to the estimated value of the land) will be how much all the other parties will demand
  + Zutha could auction off land
    - This could work quite well if Zutha were powerful enough in its own right, but I’m not sure how this would work if, as I think likely, a Zutha society will be largely relying on private protection agencies for law enforcement rather than a unified law enforcement agency under Zutha control
    - But Zutha is really a proxy for the collective community identity, which means ownership of land by Zutha means ownership by every member of the Zutha society (weighted by reputation). If Zutha’s (the community’s) policy about land rights is that all land that comes under the Zutha banner is owned by Zutha by default, then all members of the Zutha community will have an interest in enforcing that rule, because they benefit from the proceeds of each land auction.
    - This means that all the private protection agencies will be expected by their clients to enforce the rule that new land is owned by Zutha and auctioned off to the highest bidder.
  + Prospectors could make money by investing resources in finding deposits of natural resources and buying the land cheaply from Zutha before the true value of the land is discovered. They could then either mine the natural resource themselves, or, my likely if they specialise in prospecting, sell off the land to a mining company for a profit
* How should zutha deal with land currently “owned” by people or organisations in areas not yet controlled by Zutha?
  + I’ve already decided that land occupied by indigenous peoples should be granted to them according to Zutha, at the discretion of the Zutha community
    - Unfortunately, if the alternative is selling off that land for a profit, the incentive for the Zutha community is to be cruel rather than kind to indigenous peoples
    - I’ll come back to this issue later
  + If the existing occupant of a piece of land is a corrupt organisation operating by military force in a destitute country effectively devoid of property rights, I think Zutha’s approach should generally be to commandeer the land and hopefully re-employ whatever native workers were working there under a kinder boss (kinder because the boss wants to make profit and understands incentives)
  + If the land is occupied by a family owned grocery store in a poor community with badly enforced property rights, the most just outcome seems to me to be that the family retains ownership of the land under Zutha once Zutha extends its umbrella over the area.
  + What if the land is occupied by a popular brothel in a poor neighbourhood where human rights abuses occur, but not severe ones and no worse than the average underground business in the area? I don’t know.
  + There will be a many-dimensional continuum of circumstances that Zutha communities will have to deal with as they expand. I don’t have to think out every scenario now; Zutha communities can decide on a case by case basis what should be done with each piece of new land they encounter
  + Ideally, though, a framework should be developed within the Zutha constitution which creates incentives for Zutha communities to make decisions which are just, or at least not obviously barbarous
  + The most powerful force for creating such incentives in the Zutha framework is the reputation system. If some Zutha communities choose to be cruel to natives when they conquer, other Zutha communities will punish them by diminishing their reputation. This will have implications for members’ influence in other Zutha communities they are a part of, trade with other Zutha communities and the amount of investment that comes into their community from other Zutha communities. Even without these tangible disadvantages, I predict that being a part of a Zutha community will make people far more sensitive to having a poor reputation, the more so the wider spread and the higher the magnitude of the poor reputation.
  + If reputation fails, other Zutha communities can intervene directly to discipline misbehaving Zutha communities
  + I don’t think there is any way to create net tangible incentives for Zutha communities to give land they conquer back to the natives. It seems to me inherently more advantageous to commander the land and profit from it oneself.
  + Therefore reputation-based enforcement of ethical behaviour will be essential in such cases, I think
  + Consider the situation of an indigenous tribe whose tribal lands extend over a rich gold seam.
    - Might not a Zutha community consider it ethical to commandeer this gold-rich land and use the profits from mining it to do all the great things Zutha can do with resources, including helping to improve the quality of life of the natives?
    - If Zutha really can do great things for everybody with its resources, then maybe even libertarian-minded people would feel inclined to support such an action – more so, at least, than they would support such behaviour from a government.
    - The prospect of Zutha taking over the mine and using the profits for good (squirm) actually seems more inviting than auctioning off the land in the free market to, most likely, a powerful international company
    - So letting a multinational take a goldmine from under the feet of the native tribe that lived on the land is unappealing.
    - Letting Zutha take on the role of a paternalistic appropriator of resources for the “common good” seems more appealing, but rings more alarm bells for me
    - If these two options were unavailable to a conquering Zutha community because of the Zutha constitution, but they were serious about obtaining resources for Zutha to invest in the common good, what would they do?
    - They could give the land to the native community and then try to convince them to sell or rent the land out to earn money from it. Zutha could invest hugely in an existing or new mining company and encourage the community to sell or rent the land to that company (or they could just be the highest bidder) so that lots of the proceeds would go to Zutha and could be invested into the development of the community.
    - This option seems like a nice melding of capitalism, and thus the efficiency and innovation that comes with it, and a community-empowered drive to support the common good. Much better!

# Fri 21-Jan-11

* Advertising fee determined, per commercial product, by community.
* Allow donation to zutha via the Zutha item. This will have its return percentage set to 0 and will automatically sell donations it receives. Hence the real money worth of the zutha item will always be 0.
* Encourage donations to zutha before monetizing it. The bulk of zutha income should be divided among users and philanthropic enterprises by reputation. People that like the incentives this creates will hopefully feel drawn to donate to the cause.
* Commercial products don't receive money from zutha proportional to their reputation. They must instead pay for the advertising they get on zutha by having a high reputation - though the amount they pay should be determined by a continuous variable "prediction" market (that specifies rather than predicts). Commercial products still earn money from real money investment though.
* It should be possible to create independent communities with independent reputation, without the need to separate zutha into fragmented networks.
  + Domain specific reputation could facilitate this
  + A local community could set up a reputation domain for their community
  + The founders of the community (or those that bring Zutha into the community initially) could grant an initial set of community members some reputation, after which all external investment into the community will not contribute to the domain specific reputation of that community
  + The initial reputation holders in the communities will invest in items owned by other members of the community, thus adding to their community-specific reputation.
  + With community-specific reputation, communities can make independent decisions that concern their own community, without the intervention of users with high reputation outside their community
* If a single user is the source of all reputation in a community, and that founding user loses all his reputation, what will be the effect on the reputation pool?
  + Case study 1
    - A is a founding user who starts with a pool of reputation of 100
    - A invests 100 zuth in users B and C such that B and C now both have reputations of 10
    - User B invests 10 zuth in 10 different items owned by C, boosting C’s reputation to 20
    - C invests 20 zuth in 5 different items owned by B, boosting B’s reputation to 20
    - A’s reputation is pushed to zero (by some external investor or something)
    - A is forced to withdraw his investments of 100 zuth in each of B and C
    - The reputation of B and C falls to 10 as a direct result of the loss of A’s 100 zuth investments in them
    - C is forced to sell half of each of the 5 investments of 20 he made in B’s items, so B’s reputation falls to sqrt(50) = ~7
    - B has to reduce the size of his 10 investments in C’s items to ~7 each, so C’s reputation falls to sqrt(70) = ~8.4
    - C has to further reduce the size of his 5 investments in B to 8.4, further reducing B’s reputation
    - I think this chain reaction will actually settle down to an equilibrium with positive reputation remaining in both B and C, but it’s a disconcerting effect even if that’s true.
  + Case study 2
    - A is a founding user who starts with a reputation pool of 100
    - A invests 100 zuth in users B, C, D, and E
    - B and C both invest 10\*10 zuth in each other, boosting both of their reputations to 20
    - D and E both invest 10\*10 zuth in each other, boosting both of their reputations to 20
    - C and D both invest 20\*20 zuth in each other, boosting both of their reputations to 40
    - A loses all his reputation
    - B, C, D and E lose 10 reputation each
    - Pairs B,C and D,E retain their investments in each other since those investments only required 10 reputation and all 4 still have at least 10 reputation
    - C and D must reduce their investments in each other from 20\*20 to 10\*20, leaving reputation in each of 10+sqrt(200) = ~24
    - So the equilibrium is B,E=10; C,D=24
  + With far more users investing in each other in far more complex ways, I think even a system founded by just a single reputation provider can quickly become quite robust and independent of the founder
  + It is also likely that most investments will be much smaller than a user’s maximum allowed investment (equal to their reputation)
  + If users’ salary from Zutha is 1\*reputation per week, then a user will be unlikely to want to invest all of this in just one item for the whole week
* A bot could have many users invest in each other’s items and build up the reputations of all of them arbitrarily
  + But where would he get all the zuth from to do this?
  + The reputation system might not prevent a user from investing his maximum allowance in thousands of another user’s items to pass on the reputation, but to max out just one item requires a week’s salary.
* Usernames should be accompanied by a status indicator wherever they appear
  + It would be convenient when reading through a discussion on Zutha, to be able to know which of the discussion participants are currently online so that you can invite them to continue the discussion with you.
* Dividing ownership of an item among many users yields greater net reputation from the worth of that item
  + Starting with users of zero reputation, if an item worth 10,000 zuth is owned by one user, that user gets a reputation of 100. If the item is owned by 100 people, each get a reputation of 10, making a total reputation of 1000.
  + This is a loophole that allows people to exploit the system to gain control of more reputation than they deserve. A product purportedly made by a large company could be given thousands of owners, all of which are actually or effectively controlled by one individual.
  + The reputation contribution of every item could be calculated separately, such that a 10,000 worth item would contribute a total reputation of 100 which is then divided among all its owners.
  + This would mean that it is far more effective for accumulating reputation to make a heap of somewhat popular items, than to make a small number of extremely popular items
  + Let’s say an author writes a bestselling novel that accumulates a worth of 1,000,000, yielding a reputation of 1000 to the author.
  + A prolific forum hound could accumulate the same reputation by writing 100 comments which each achieve a worth of 100.
  + This actually seems pretty fair to me.
  + With the old system, the forum hound would need to write 10,000 comments worth 100 zuth each to match the reputation of the author.
  + Given the tendency of popular items that reach a certain threshold to explode disproportionately in popularity, I think it’s pretty fair that acquiring reputation in this way be given a relative disadvantage to accumulating reputation through a lot of low yield hard work.
  + This system makes it easier to transfer reputation to another user
    - Let’s say user A has 10,000 zuth and 100 reputation and wants to transfer as much reputation as possible to user B
      * He will invest 100 zuth in 100 different items owned by user B
      * With the new system (sqrt on individual item worth), this will produce sqrt(100)\*100 = 1000 reputation for user B
      * With the old system (sqrt on net worth of items), this will produce sqrt(100\*100)=100 reputation for user B
      * Here the old system is preferable.
    - What if user A only has 1000 zuth available?
      * New system: B reputation = sqrt(10)\*100 = 316
      * Old system: B reputation = sqrt(10\*100) = 31.6
    - What if user A only has 400 zuth available and only 20 reputation
      * New system: B reputation = sqrt(20)\*20 = 89
      * Old system: B reputation = sqrt(20\*20) = 20
    - The difference in the capacity to pass on reputation between the old and new system appears to be a factor of sqrt(A reputation).
    - The old system behaves better for this scenario because A is only able to pass on reputation to B of a similar order of magnitude to his own reputation.
  + The ideal scenario would be to have the behaviour of the old sqrt-after-aggregation system for situations that allow passing lots of zuth to a single user, but the sqrt-before-aggregation system for situations where the zuth is divided up among many owners of an item.
    - The problem is that these are two completely different behaviours and it’s not clear what should be done about the case of just 2 owners.
    - If sqrt-before-aggregation was used for items with multiple owners, but sqrt-after-aggregation was used for items with one owner, then exploiters would simply pass reputation through many items all owned by the same pair of recipient users. Since such items would be owned by more than one user, sqrt-before-aggregation would be used and each owner would earn reputation ½\*sqrt(donor\_rep)\*(donor\_cash/donor\_rep). This is far more than would be obtainable by a single user owning many items because then sqrt-after-aggregation would be used and recipient would earn just sqrt(donor\_cash)
  + It’s clear that a more elegant solution is needed.

# Sat 22-Jan-11

* Zutha should support temporarily secret communication between elected commanders and zutha.
  + This data would be released once the community votes to release it - presumably after the conflict is over.
  + It would be easy to use Zutha’s distributed network to allow conventional secret communication between individuals using public key cryptography
  + It would be a very difficult cryptographic challenge, though, to allow such secret communication in a temporary way such that the data can be decrypted later by decree from the community
  + It would be harder again to allow commanders to give secret commands to the Zutha network itself such as to command robotic forces owned by Zutha
  + I just realised there will be no way to stop users from communicating secretly if they want because all users’ public keys are available for signing purposes
    - This doesn’t make redundant the idea of supporting temporarily secret communication, though, because an expectation can be made of commanders that they use this temporarily secret communication system so that their actions can be examined later.
    - What is to stop commanders from deciding to communicate entirely secretly for some things but just use the temporarily secret communication system for enough things to keep suspicion away from them once the conflict is over and this information is released to the public?
    - Well, all their actions (at least those done through Zutha and those publically knowable) will have to be accounted for. If commanders are wily enough, they might be able to construct elaborate alternative justifications to everything they do, for the benefit of the public, but given how much reputation they will have on the line if they are found out, this is an unlikely scenario I think. The easiest course of action for these commanders if they are interested in maintaining their reputation and influence in Zutha will be to just use the temporary communication system for everything so they don’t have to worry about inventing elaborate explanations for everything in addition to their real communication. They will of course need to invent the alternate discussions (the one’s intended for the benefit of the public) in real time, because Zutha will know exactly what time they say everything that is said over the temporarily secret communication platform.
    - Perhaps the strongest force to compel commanders to use the temporarily secret system is that there will be multiple commander with an independent desire to maintain their reputation and so it will be very difficult for one of the commanders to convince the others to break the rules given that the status quo (particularly among commanders selected for their loyalty to Zutha) will be to follow the rules and use the temporarily secret system. A rebel could communicate secretly with each other commander, but if any of them was not convinced, by this first attempt, to join the rebel in duplicitousness, the rebel’s behaviour will be exposed to the rest of the commanders over the temporarily secret channel and the rebel will likely be expelled from his commander role
  + Now, how to allow temporarily secret communication?
    - It would not be too hard to divide the data constituting a private key into multiple parts which are all required to reconstruct the key.
    - One of the selected commanders could be chosen at random to generate a public-private key pair, divide the private key into pieces, and send each to a different Zutha host. The hosts that receive one of the pieces would be selected using the same selection algorithm used to select hosts in the process of creating items. The selection of the commander to generate the key would also use the selection algorithm.
    - The commander who generates the key would also share it with each of the other selected commanders.
    - The commanders would encrypt all their communication with the public key(signing it with their personal private key) and read it with the temporarily secret private key
    - When the Zutha community decides to release the commanders’ communication, the hosts holding the pieces of the private key will share the pieces with the whole network
    - Why is this any better than simply expecting the commanders to share their private key when the community orders it?
      * Before a conflict starts, all the commanders are likely to be far more loyal to Zutha than they are to each other
      * Once they are broiled in the conflict, their loyalty to each will likely increase, as well as their distance from the Zutha community. This is a likely effect of extended secret collaboration under immense pressure, I think.
      * The commanders will likely not have a problem using the temporarily secret channel initially, but they may later become more reluctant as difficult moral decisions are made and guilt and the fear of public misunderstanding come into play.
      * If this happens, the public will still have available, the record of communications up to the point that the commanders become collectively uncomfortable enough about communicating through the temporarily secret channel that they stop doing so (for the sensitive things at least).
      * If however, the public is relying on one of the commanders to explicitly give them the private key, especially if they ask this of them during the conflict, they are likely to encounter a lot of resistance. The commanders will likely have formed enough of a bond to team up in keeping secrets from the public if they feel embarrassed or vulnerable about something. The prospect of losing reputation for failing to give up the key when demanded may not be strong enough to compel the commanders to give it up, and they are likely to be able to gain some leeway by touting the security risk of releasing the information. If the community already has the key (conditional on enough people agreeing to release it), then the power is in the community’s hands. The commanders can try their best to convince them of the security risks of prematurely releasing the information, but if enough Zutha users are suspicious enough of them, they can override the commanders.
      * Once the commanders’ communications are actually revealed, it will be a far less ambiguous process for the community to assess their guilt than if they needed to do so without any information, as in the situation where they relied on the commanders to voluntarily hand over the encryption key.
    - What is to stop the creator of the key pair from giving the wrong private key to the Zutha hosts?
      * The best that can be done is to make sure that the commanders all need to make the decision collectively to give the community the wrong key
      * Before the commanders have bonded with each other, it would be very difficult and unlikely for them to collude with each other in this way
      * All that is required is that the hosts holding one of the pieces of the private key send their piece to each of the other commanders (than the creator of the key) and every commander assembles the pieces and confirms that the result matches the key given to him by the key creator.
  + How could Zutha be given commands that are secret to the public until the conflict is over?
    - Let’s say the commanders want to order Zutha’s army of Unmanned Aerial Vehicles to fly to a particular enemy target without the enemy being able to find out that they are coming.
    - The army of UAVs would be owned distributedly by many hosts.
    - The owner of a UAV would have absolute control over its actions, including the ability to remotely upgrade or modify its software in any way
    - The expectation of hosts contributing their UAVs to Zutha would be that they program their UAVs to obey any orders given to them by a minimum percentage of a valid set of Zutha hosts, where the validity of the set would be determined by running the selection algorithm on the message sent.
    - If one of the UAV owners caused their UAVs to deviate from these expectations, the remaining UAVs – all owned by independent hosts – would turn on the misbehaving ones, perhaps destroying them, though this should not be done without giving the owner of the deviant UAVs a chance to correct a possible mistake. Perhaps the swarm should only resort to destruction of deviant hosts if the deviant behaviour was in a category of severe misbehaviour, such as that which would endanger the swarm of the Zutha community. In other cases, the Zutha community would need to vote to destroy the deviants, which they should only do after establishing that their owner is either deliberately causing them to misbehave or is no longer in control of them.
    - A temporarily private key could be shared between the commanders and the UAV swarm to secure communications between them. Many hosts could be used to relay the messages, but only the UAV devices themselves would be able to decode them.
    - The biggest challenge is how to keep knowledge of the UAVs actions from their owners
    - An owner could have his UAVs behave in all ways correctly, except that some of them sometimes send messages to their owner about what their orders are.
    - It could be possible to have other UAVs scan for unauthorized transmissions by their companions. If one was detected, all UAVs under the same owner could be destroyed or deactivated.
    - This seems remotely plausible, but I would much prefer a fundamental solution than one which would require a constant battle of wits between Zutha and malicious device owners.
    - Why do devices need to have a single owner?
      * Well the expectation is that device “owners” pass ownership of their devices to the entire Zutha community
      * Actually this is not quite true – I had allowed that owners have the power to modify their devices’ software remotely
      * Why couldn’t full control, including the ability to send software updates, be granted to Zutha? Each command would be given by a different set of hosts selected using the selection algorithm.
      * The reason for decentralizing the control of devices (like UAVs or nanobots) is so that a hacker has to hack many independent entities in order to control a whole swarm of devices.
      * This isn’t quite achieved if the selection algorithm is used to determine the source of every command, because if Zutha itself is hacked then no one can independently retake control of a subset of the device swarm. Hosts can refuse to participate in commands to devices if they morally disagree with them or believe Zutha has been hacked and the commands are not actually coming from the community, but this doesn’t give anyone the power to give alternate commands to the devices, only contribute to ensuring they receive no commands at all.
      * What if groups of devices were randomly assigned to small groups of hosts?
      * If the owners of a group of devices are a known fixed group, then this group can collaborate to disregard Zutha’s orders, whereas if the owners of a given device is newly determined by the selection algorithm for every command, then it would be extremely difficult to coordinate a rebellion against Zutha because effectively all Zutha hosts would have to coordinate at once to reliably give commands to the devices. The selection algorithm is explicitly designed to prevent prediction in advance of which hosts will be required for an order before it is given, so it would not be possible to decide on an order to give the devices and then find just the hosts required to give it.
      * The owners of devices need not be hosts, I just realised; they can be any entity with a public key
      * It would be better to use groups of users as device owners, I think, as they can be selected based on a more intimate public understanding of their personality.
      * An issue remains as to how to ensure the devices are securely controlled by their supposed group of owners
      * Device manufacturers could embed hidden logic that overrides the standard software controller at a time of the manufacturers choosing, or they could just cause the device to secretly send them input and output data wirelessly.
      * Devices could be produced open-source such that every aspect of their design can be analysed by the whole Zutha community
      * Someone still has to actually make a given device, though, and the maker could make something indistinguishable from the publically accepted design that nonetheless contains hidden backdoors.
      * If making devices for Zutha was a lucrative enough business, the cost of losing that business might be a sufficient disincentive to betrayal for Zutha to rely on high reputation manufacturers that have been loyal to Zutha for a long time.
      * Devices could be subject to random inspections by many independent Zutha-loyal entities. If inspectors found backdoors in any of the devices built for Zutha by a manufacturer, all devices made by that manufacturer would be recalled and the manufacturer would lose its lucrative business with Zutha.
    - If a device was controlled by a set of randomly selected owners, then those owners would have to collude in order to give their device an order to share its orders with them.
    - For owners that are randomly selected from among high reputation Zutha users, such collusion is extremely unlikely. It could be made even more unlikely by regularly reassigning device owners. The existing owners would be told by Zutha to transfer ownership to a new set of algorithmically selected owners. All orders sent between a group of owners and their devices would be public knowledge, so it would not be possible for them to give a different order to their devices which only partially transfers control or something.
    - When it comes time to transfer control of a swarm of devices to Zutha’s military commanders, all device owners would command their devices to listen to orders sent from the military commanders.
      * The military commanders would then send a temporarily private key to the devices.
      * The hosts holding the parts of the temporarily private key would send them to an algorithmically selected set of devices so that those devices can confirm they received the same private key as the Zutha community did in distributed form.
      * From then on, the military commanders can give temporarily secret orders to the devices which only the devices can currently decode because they have a temporarily secret private key which is only currently known by them and the commanders.
    - The commanders should only be allowed to give a limited set of commands to the devices. They would not be allowed to give any commands that modify the devices’ software, only commands necessary for actual military manoeuvres with the devices.
    - All communications between the commanders and the devices should be required to go via a programmatically selected set of hosts
      * This is so that all communications with devices are recorded by Zutha and can be publically examined after the conflict.
      * The devices should be programmed to enforce this rule. They should only accept commands from commanders that have been signed by a minimum number of the set of hosts necessitated (via the selection algorithm) by the message and the current time.
  + I’ve thought of something that might stop commanders from being able to communicate secretly independent of the Zutha network
    - A user’s public key might be publically known, but his physical location is not.
    - Unless two commanders know each other personally (in which case nothing whatsoever can prevent private communication between them) they must use the Zutha network to get messages to one another.
    - It might be, though, that the commanders are public people with publically known contact details like an email address
    - It is likely that at least one of the commanders is an anonymous user with respect to his or her real world identity. If there is even one such member among the commanders, then it would be impossible for them to coordinate duplicity against Zutha because they would not be able to get in contact with that member secretly.
    - Actually I’ve realised that it probably isn’t really that important to prevent secret communication between commanders because everything they can do that is powerful will go through Zutha and be recorded anyway.
    - What if the commanders were faced with a choice about whether to rescue a village of innocents and they chose not to because they don’t really care about the wellbeing of the primitive people that live there? They might be ashamed to let the Zutha community know this is the reason they decided not to protect the village. If they can communicate secretly, they can agree to not care about the village privately, but then give some “official” excuse for their abandonment of the village on the temporarily secure information channel.
    - It would be good if commanders were unable to communicate secretly and were forced to discuss their decisions openly (eventually) because it would likely improve the moral substance behind their decisions. I think this ideal is likely to be achieved relatively consistently because of the independent nature of the commanders and the possibility that some might be impossible to reach except through Zutha. Even if sometimes the commanders do form enough of a bond to start doing some communications secretly, they still have to be able to justify all their actual actions to the satisfaction of the Zutha community that this, I think, will be good enough.
  + All the commanders should need to agree unanimously on their decisions
    - This would be enforced by having Zutha’s resources (such as war devices) only accept commands that are signed by all the commanders.
    - Actually it might be better to allow commands to pass with only a significant majority of support from the commanders; otherwise a single stubborn member could detrimentally slow response times, which are of critical importance in a military conflict.
    - It would be important if a non-unanimous command were to be passed, that all commanders were notified of the command so that the commanders could not exclude a member without him knowing.
    - If a member believes he is being unfairly excluded or that the other commanders are behaving in a misguided way, he would be free to turn to the Zutha community to complain. His own reputation would be on the line if he revealed information that was detrimental to the war effort, though.
  + For added security, many temporarily secret keys could be used by the commanders at any one time, and they can change their keys over time.
  + The commanders should be able to exclude any of their number from further communication if they suspect a compromise
    - This could be made possible by having one temporarily private key per commander instead of a common one.
    - When a commander sends a message to the others, his client would separately encrypt his message with each commander’s temporarily private key. All of these separately encrypted versions of the same message would be put in the one message item and posted to the commanders’ discussion stream.
    - The actual items posted by the commanders to each other will be publically visible, but their content will be encrypted. The actual encrypted data should be publically viewable using a normal client. This is in line with the principle of providing any information accessible to resourceful agents, to the whole public.
    - If the commanders decide to start excluding a member from their communication, they simply have to all modify their client’s settings to stop encrypting their messages with the excluded commander’s key.
    - The excluded member will know he has been excluded – as will the whole Zutha community. The remaining commanders should probably be expected to make a statement explaining in at least rough terms why they evicted one of their members.
    - The evicted member may dispute the claims of the other commanders.
    - It would probably be very useful for commanders to be able to selectively reveal past messages. An evicted member could do this to illuminate corruption among the commanders and/or an unfair eviction and the remaining commanders could reveal enough conversation to justify evicting the member they did.
    - Selective revealing of messages should not require any changes to standard cryptographic practice to enable. Messages are usually encrypted using a randomly generated symmetric key, and this symmetric key is then encrypted with a public or private key, depending on whether the message is being signed or made private. Any of the commanders can reveal the symmetric key used to encrypt a given message, and the message can then be decrypted by hosts and made visible to the whole community.
* There’s no need to make a relationship directly with banks or companies for managing Zutha’s finances
  + The existing reputation held by such institutions would be a bonus, but I think it would be very difficult to get through bureaucracy and convince them to implement the software to allow distributed control of accounts.
  + To do business with Zutha, a company would have to implement the Zutha client software that knows the selection algorithm and can verify that a message was sent by an appropriate set of hosts.
  + Zutha users already work with the client software in all their communications with Zutha. It shouldn’t be too hard to write and distribute software that automatically runs an account on behalf of Zutha. All a user would need to do is set up an account and give this open source software control over it.
  + Users will get to keep a large part, or maybe all of the interest they earn by holding money for Zutha, so there will be an incentive to maintain the relationship
  + Of course, only high reputation users would be given the privilege to manage money for Zutha and so they would have their Zutha reputation and the accompanying income and influence associated with it, to lose if they failed to send money when asked to by Zutha.
  + If I can get big banks and companies like Google to use the Zutha client software to receive transaction commands from Zutha – great. But I don’t want to have to count on that happening too often.
* Ownership of zutha should be divided among development contributors, infrastructure providers, philanthropic causes, and the zutha community.
  + The relative proportions are decided by the community.
  + Within each category the breakdown of ownership is further determined by reputation.
  + Zutha's income from commercial advertising subscriptions, taxes on investment, and direct donations will be divided among these owners.
* A strategy to create better incentives to win for sporting competitions
  + In a lot of sporting competitions, clubs will earn money even if they perform mediocre
  + If only the one winning team (or individual) benefits from the competition, the incentive to invest a lot of resources in a sporting competition with a lot of competitors would be low
  + What if competitors gained a proportion of investments in prediction markets in the outcome of them winning?
  + There would be an incentive to make investors think you have a high chance of winning even if you don't actually win.
  + This would be done by simply making a team the owner of the candidate in a prediction market which represents the case of them winning
  + The current plan for how remuneration passes to owners of popular items is that they chose a proportion of “donations” to give back to investors, and keep the rest in the form of an investment in the item owned by them
  + For the case of sporting competitions, this would mean that a competitor may only benefit from making people think he will win without actually winning, if he sells the investments he has earned in the item representing the outcome of him winning, before the outcome of the competition is determined. This would be in effect a public admission of defeat and would probably lead to a sudden reduction in the amount of investment in that outcome.
  + In think this system could have some very interesting psychological effects
  + My guess is that the apparent incentives associated with making investors think you will win, will actually work in encouraging more competitors to try harder even if they don’t think they can win, yet I don’t think it would be common for competitors to actually cash these incentives because of the social and psychological costs of publically predicting one’s own failure.
* Extreme public sentiment should be able to compel a landowner to sell their land at the current market price i.e. through an auction.
  + This might be done if someone disregarded social expectations and started building a huge and ugly building in an area treasured for its beauty.
  + This empowerment of the majority to discipline extremely distasteful development in an area would be a substitute for zoning regulations.
* Some potential outlets of Zutha resources will require a minimum allocation of resources before they can go ahead
  + For example, a proposal to build a public park might cost a minimum amount
  + People will invest in that proposal to the degree that they support Zutha’s investment in building that park
  + The current worth of this Proposal will be translated into a monetary amount by dividing Zutha’s current budget across all resource outlets in proportion to their relative worth.
  + Until a proposal reaches a sufficient worth that the resources that would be allocated to it meet the minimum requirements for the project to go ahead, that proposal will not drain any resources from Zutha’s budget
* How can a user’s physical location be secret, even when they are able to communicate with any other Zutha user?
  + All messages sent via Zutha are propagated across the whole Zutha network
  + Hosts only need to know how to connect to their neighbours
  + Hosts also know which clients are connected to them and push messages to connected clients at any time until the client breaks the persistent connection
  + A client may be connected to multiple hosts simultaneously, but it need not be. Just because every communication occurs with multiple hosts at once doesn’t mean the client has to be directly connected to all of those hosts at once. Those hosts just have to digitally sign their messages to the client and propagate them across the network.
  + The need to propagate every view request and response across the whole network seems needlessly inefficient. It would be cool for data collection’s sake, though.
  + Shortest paths between nodes could be found dynamically by keeping track of the number of hops since the source host for every message sent. Hosts would accumulate a real-time database of which of its neighbouring hosts to send a message to so that it will most quickly reach a target node, either a host or a client.
  + Hosts should not share their own databases with other hosts: a host should only know which of its neighbours to send a message to to fast track it to a particular host or client; it should not know the entire route that will be taken.
  + If someone wanted to find out where a user is physically located he would have to look through the private databases of one host after another, manually following the trail of neighbours that would be selected to find a particular client. Since hacking just one host would be very difficult, this process should be impractical.
* It would be cool to have every host know the full history of view requests from every client, but spreading this data during busy times would be wasteful
  + Such data should be propagated between hosts during times of low traffic
  + It may actually be that sending every view request to every host would be practical
  + Almost no processing would be required to receive a view request or response, store it, and transmit it on to other hosts probabilistically
  + Already, web servers are capable of handling extraordinary amounts of traffic, and this capacity is going to increase rapidly. Since view requests are only small amounts of text data, and responses are only 1 page worth of text data, the actual amount of data should not be overwhelming
  + Then again, the number of requests is probably more important than the amount of data transmitted per request
  + I guess I just don’t know yet whether requiring every host to store and relay all request and response data will be practical.
  + Another benefit of sending every view request to everyone would be that the whole community of hosts could verify view requests to identify errant responses, instead of relying on clients to report inconsistencies.

# Sun 23-Jan-11

* Provide an auto-sell investment feature that automatically sells a user’s existing investments – selecting based on rules the users chooses – whenever the user wants to make a new investment that he doesn’t currently have enough cash for.
  + The user should be able to “lock” investments to explicitly exclude them from auto-sell
  + A rule for selecting investments to sell might be something like: chose the oldest investment excluding locked investments
  + Another rule could be: sell the smallest investments first
  + Another possible rule: choose the investments that have appreciated most in value
* It would be useful if the Zutha community could verify that messages being sent between military commanders had actually been encrypted with the right key.
  + The public key corresponding to a temporarily private key will be used to encrypt a randomized symmetric key for every message.
  + Since the symmetric key is cryptographically random there would be no way to know anything about it after any arbitrary transformations such as encryption with a public key
  + I can’t see a way for the public to be able to check that the symmetric key has been encrypted with the public key it should have been encrypted with, because they do not know the symmetric key so they cannot encrypt it with the public key themselves and compare the result with the given encrypted symmetric key.
  + Thus I don’t think this convenience will be possible
  + At any rate, it might not really be that useful. If commanders wanted to keep information from the public they could easily just encrypt their messages twice, once with the temporarily private key and again with another private key which is actually fully private to them. I’m sure there are all sorts of other ways to keep information from the Zutha public if all the commanders agreed to. The most potent defence against this happening is the independence of the commanders and their attachment to their Zutha reputation.
* It seem unfair to me for an existing powerful international corporation to come into a poor society and buy mineral rich land cheaply off them (because it isn’t actually worth much to them without the resources to extract to minerals) and then carry the huge profits from mining the minerals internationally
  + Such an operation would of course benefit the local society because it would provide jobs and a lot of new wealth to them (though not necessarily)
  + It does seem appropriate to describe such a scenario as “taking advantage of” the poor, though, because the mining company is not contributing to the society that owned the mineral rich land in even the same order of magnitude as it benefits from buying that land from them.
  + A good parallel to make here would be to consider what the mining company would be able to get by mutual agreement from a much wealthier and more educated society. I think it likely that an educated society would make a more mutually beneficial deal like granting the company the right to mine the minerals and gain half the profits if it gives the other half back to the landowners – the local community.
  + A poor and uneducated society is much more likely to fall for selling a goldmine for a practical penny because the penny seems like so much to them
  + The biggest danger to the community is if a powerful company was able to get an unfair deal out of them before competition arrives to offer a better deal for the landowners.
  + This is where Zutha could, and I think should, help: by ensuring that deals involving huge stakes, particularly deals involving poor uneducated people, should be made public enough for long enough to allow sufficient competition to drive up the price to its value under an approximation of perfect competition. The market price of rights to mine a goldmine under high competition is likely to be at the very least in the same order of magnitude as the actual value of those rights.
  + It would be better for the local community to allow market forces to find the best deal for them, rather than have a paternalistic authority – even one like Zutha – enforce specific restrictions, supposedly with the best interests of the poor landowners at heart. This is because any restrictions on the deal making process will only reduce the fineness (it could even be described as resolution as in the pixel resolution of a graph) with which the optimal mutually beneficial deal can be found. If Zutha tried to force mining companies to conform to a minimum level of contribution to local communities then either the minimum will be less than the maximum a mining company could afford in which case competition would make the minimum redundant anyway, or the minimum will be too high in which case no mining company will be willing to enter a deal, no goldmine will be constructed, and everyone will be worse off.
  + The same laws apply to any kinds of restrictions a governing authority could impose. Another example: if it was mandated that no international companies were allowed to develop mining operations in a given region, but only ventures owned by people in that region, then the immense intellectual and physical capital available internationally to produce the most possible wealth from the mine, would be excluded. A free market would not prevent the local community from refusing an international deal and choosing instead to develop the mine themselves; if that was the most beneficial thing for the local community to do, then they should and could do that, but mandating that they aren’t allowed to invite in an international company to help them develop the mine only reduces their capacity to produce wealth for themselves using their assets (the goldmine in this case).
  + Of course, choosing the best deal from among an arbitrary variety of potentially very complex possibilities will be extremely daunting at best for a poor, uneducated community. This, again, is something Zutha could and should help with.
  + Zutha should encourage and perhaps fund volunteers to advise local communities on what is in their best interests. Such a system could probably be kept honest by reputation mechanisms, but there are still risks that interest groups without the community’s best interests at heart might find their ear. If the Zutha community is involved in picking people to recommend as advisors to a community, they will have incentives to pick people who will be friendly to the interests of the Zutha community. I think there would be a large enough proportion among those willing to be outspoken on such issues, who would be authentic proponents of values like fairness and equality, that the Zutha community would pretty reliably support advisors who really do have the best interests of a local community at heart.
  + There would be many resources available for finding the best deal to accept that an advisor could recommend, including prediction markets, professionals and entrepreneurs willing to invest effort in finding the most profitable deal in exchange for a percentage of those profits – thus giving the entrepreneur the right incentives.
  + This last source of advice would need to be taken with a grain of salt, because an advisor whose primary incentive is to find the deal which would bring in the highest monetary value, may not sufficiently consider other factors which are important to the local community such as the impact on community cohesion and culture and the environment.
* Zutha’s policy regarding hostiles
  + Disable enemies with anaesthetized bullets.
  + Capture and disarm them.
  + Provide them one night of comfortable accommodation.
  + Allow entrepreneurs to offer them deals in which the entrepreneur trains them in exchange for a percentage of their salary for a period of time - maybe life.
  + Allow captured attackers to find work immediately if the can and want to, or leave if they want.
  + The greatest challenge with implementing such a policy will, I think, be convincing the people attacked to accept it. If this would be too great a challenge, though, Zutha could take the captives to a different Zutha community to offer them freedom.
  + It would be important to punish attackers for the actual harm they do if they do any (the intention would be to disable them before they can actually harm innocents) because otherwise an incentive would be created to actually attack Zutha communities. The incentive this policy is intended to create is for soldiers under command of a hostile ruler to surrender themselves willingly to Zutha hands because they get a better deal there than under their military masters.
* Zutha should invest in extensive surveillance technology like spy UAVs and satellites and make this information publically available.
  + On the whole I think this would make the world a much safer place.
  + People aren't as mean when the public is watching - particularly if they are literally watching in real time as opposed to watching second hand through news stories.
* Law enforcement
  + The harmed party in a crime should specify via their protection agency, multiple punishments which would be acceptable to them.
  + The defendant’s protection agency negotiates an acceptable punishment for their client.
  + A criminal may be given the choice between a whipping and a sentence in jail at the expense of the prosecuting agency. Some protection agencies may not accept whipping as a punishment under any circumstances because their clients consider it unethical. If such an agency came up against a prosecutor that would only accept what it considers unethical punishments, it might negotiate a punishment that it would pay the cost of, such as a jail sentence.
  + The worst case scenario would be a battle between protection agencies that come to a fundamental impasse. This would be necessary for such cases where criminals set up a protection agency that refuses to accept substantial punishments for terrible crimes. In this case many agencies, representing the majority of society, would gang up on them and destroy them. They may only need to threaten for the criminal protection agency to back down though.
  + It may be wise for people to buy primary protection from one agency as well as paying a smaller premium for secondary protection against their own agency in the case of them breaking their contract.
  + Secondary protection could also be provided to all citizens by zutha. Zutha would only operate under very simple rules though. If a protection agency started a protection racket against its clients by blackmailing them into continuing to pay it and not buying protection from another agency, it would be zutha's responsibility to shut down this agency when the predatory behaviour is discovered. Since the victims would not be paying any other protection agencies, none of them would have an incentive to intercede on their behalf. If people paid for secondary protection though, perhaps from a group of agencies, these agencies would have an incentive to intercede against a predatory primary protection agency because the secondary protection agency group would lose customers if the customers saw it not fulfilling its contract with other customers.
  + There would also likely be many charity protection agencies that protect the poor who can't afford private protection. Zutha should fund such charity agencies. People who can afford protection might also be satisfied by the more basic blanket protection provided to all by charity. The free protection would likely only cover the main dangers faced by the poor like rape, murder and theft though. Charities would be unlikely to be able to afford the legal costs of protecting against more subtle crimes like inappropriate signage leading to injury - the kinds of things rich people sue over. And there are unlikely to be charities for crimes against businesses, like leaking trade secrets.
* Sqrt before or after aggregation for reputation calculation?
  + Let’s go back to the fundamentals of why I want to calculate reputation as the square root of net investments in owned assets
  + One consideration is I don’t think someone who benefits from increasing returns to reputation – what might be called the “Harry Potter effect” – should receive the same level of influence as someone who gains a significant amount of reputation from a huge number of relatively popular contributions.
  + Another goal is to make it very difficult to pass reputation and wealth from one user directly to another; gaining reputation should require the affirmations of many users.
  + I recently realised that while the creator of a Harry Potter doesn’t deserve as much influence in Zutha as a super popular forum contributor, I think the Harry Potter creator deserves more monetary reward.
    - The main purpose of giving money to users according to reputation is to encourage people to freely distribute creative works that required a large investment of resources to create. A case in point would be to encourage open source development of software.
    - On the other hand, for the prolific and popular forum hound, reputation would be its own reward. Indeed, I want the accumulation of reputation to be the primary incentive for such activities, rather than monetary reward, because I want to give such people reputation-based influence and be able to trust that reputation is important enough to them to keep them honest with the influence.
  + If the distribution of Zutha’s real money wealth was portioned according to absolute net value of owned items, then this would reward “Harry Potter creators” more than forum hounds because it will be easier to attract a large amount of investment in a few super popular items that have a strong brand name – like Harry Potter – than by creating a large number of obscure, domain specific contributions.
  + Influence, on the other hand, could be based on the sum of the square root of the worth of every owned item, in which case the forum hound would have the advantage, because he can produce a much larger number of items which attract some investment.
  + So reputation should be calculated as the net worth of all owned items, which will be weighted towards creators of a small number of super-popular items, and should determine the amount of real money received form Zutha
  + Influence should be calculated as the sum of the square root of the worth of each owned item, which will be weighted towards creators of large numbers of moderately popular items, and should determine the maximum quota a user can invest in a given item and the ability of a user do certain actions. Influence will be domain specific.
  + That leaves how to calculate a user’s salary in zuth
  + Let’s first check how well the above definitions of reputation and influence satisfy the goals
  + Case study: user wants to give reputation to a specific user
    - User A has influence of 100 and cash of 1000
    - A invests 100 zuth in 10 different items owned by B
    - B gets reputation of 1000 and influence of 100
    - A could also invest 4 zuth in 250 items owned by B, yielding 500 influence for B
    - Or 1 zuth in 1000 items yielding 1000 influence for B
    - Or 0.01 zuth in 100,000 items, yielding 10,000 influence for B
    - Ah… there’s a rather nasty flaw in the square root idea: the sqrt of a number less than 1 is larger than the number itself.
  + The real problem here is that a user can give an arbitrary amount of influence to another user. This ability is sort of limited by how much cash the influence donor has, but given the flaw I just discovered with investments of less than 1 zuth, the donor can donate more influence than he has in cash, without his own influence even being a factor.
  + For starters, I need a function with a negative 2nd derivative which is strictly less than y=x.
    - This seem to be a very difficult problem – I’ll get back to it later
    - I found it: y=sqrt(x+1)-1
  + I think the biggest source of protection against the exploit of investing a tiny amount in a huge number of someone else’s items, is the difficulty of making that many investments look authentic.
  + The problem is an exploiter could invest 0.0001 zuth in 10,000,000 items, granting 100,000 influence to a user just before the moment salaries are paid. If salaries are based on influence this would instantly multiply the cash controlled by the exploiter by 100. It would be difficult for the zutha community to get this much cash back off him.
  + Using the function Influence=sqrt(worth+1)-1, the exploiter would only be able to transfer at most about the same amount of influence as he has in cash, resulting in a mere doubling of the cash he controls if he does so just before payday. This is much better, but still far from ideal.
  + This system is looking fundamentally vulnerable. Let’s look at the problems associated with sqrt-after-aggregation
  + Case study: possible sqrt-after-aggregation exploit
    - An item representing an open source software library is worth 100 zuth
    - A single person created the library, but instead of putting it under a single owner, he creates 100 users and claims all of them own an equal share of the item
    - The 100 zuth will be divided equally among the 100 owners adding 1 to each of their reputations (let’s define reputation for now as total worth of all owned items). If each users started with 0 reputation, their total reputation is now 1, and sqrt(rep) is 1
    - If salary equals reputation, then the owner of these 100 users will receive a net salary of 100 across all of them, instead of 10 if he uses a single user.
    - What happens if the exploiter does the same thing again?
    - If he divides an item worth 300 among the same 100 owners, their total reputations go to 4, yielding salary of 2 each and 200 in total, as opposed to reputation of 400, and a salary of 20. This is no different from the case of having just one item worth 400.
  + So a user can accumulate more total cash from salary this way, but he gives up having any high reputation users. Thus he will not be able to use that cash to do anything destructive because none of them will have enough reputation.
  + This is not really such a bad flaw, because there is a major trade off required to gain that extra cash. Since building reputation is really what the game is all about, not accumulating cash, this exploit is unlikely to be all that attractive.
  + So the system I’m running with at this stage is reputation = total worth of all owned items and salary=sqrt(rep+1)-1.
    - Reputation is the measure used to determine permissions and portion real money salary from Zutha
    - I’ve forgotten why I decided to use reputation (the old sqrt version) as a cap on investment in one item. I’ll get back to this later.
  + The only goal I lose with this system is that a forum hound no longer has an advantage against a “Harry Potter creator” in influence.
    - Is this really so bad?
    - I think the example of Harry Potter is distorting my perspective on this matter. If someone creates something of great value in a field, they deserve great influence in that field.
    - People that create super-popular things or become super popular themselves are natural leaders of opinion in society; why should that not be reflected in Zutha?
  + The reputation based investment cap is the mechanism I thought of to assign influence based on reputation
  + If the cap was calculated as the sqrt(rep+1)-1 as with salary, then the problem arises that those 100 users with 1 reputation each (100 total influence) would have far more total influence than the single user with 100 reputation (10 influence). This is a much more severe problem than the ability to get more cash distributed across many weak users.
  + The original intention I had when I decided to use sqrt(rep) to determine influence, was to forced influence to be more distributed across more users. What I realise now, having thought about the exploits associated with both the sqrt-before-aggregation and sqrt-after-aggregation systems, is that, due to these exploits, influence will not reliably be distributed across actual people, only across users. A single person will be able to use the exploits to pretend to be multiple people and thus gain the additional influence that is supposed to be reserved for a group of independent people.
  + If I was going to adopt either sqrt-before-aggregation of sqrt-after-aggregation, I would chose sqrt-after-aggregation because it will be harder to create a huge number of users than a huge number of items, especially once stronger identity begins to be required for users.
  + I’m beginning to come to the conclusion that using the sqrt function to try to increase the degree of democracy is fundamentally unworkable. It is simply not practical to distinguish reliably between an individual and a large group and trying to give the large group an advantage in terms of power only provides opportunities for exploitation by resourceful entities like big companies.
* Salary per unit time should be calculated as a fraction of reputation. For example, a salary might be something like 1/30\*reputation per day.
* Now that salary is linearly related to reputation, and reputation is linearly proportional to all currency in the Zutha economy, a salary of 1\*reputation per month will lead to a doubling of the money supply every month.
* Let’s say someone has a lot of cash and they want to convert it to reputation. How would they do it?
  + A person X owns users A, B, C and D
  + A has 10 reputation and 90 zuth
  + A invests 10 zuth in 3 items owned by each of B, C and D for a total cost of 90 zuth
  + B, C and D now have 30 reputation each, but no zuth
  + B, C and D will accumulate zuth very slowly. As they do, they invest it back into items owned by A, adding to A’s reputation
  + If other users decide that some of the investments made by X in this process are not justified, they will invest negatively in the reputation of the users involved, reversing X’s efforts to increase his own reputation.
  + It will be important that the salary is very small in comparison to reputation so that it is impractical to rapidly accumulate reputation by transferring cash into each other’s items.
  + I’m thinking the salary should be more along the lines of 1/365 \* reputation per day – this would result in inflation of 100% excluding the effects of taxes
  + Even if salary is not large enough to allow rapid conversion of reputation into cash, two already high reputation, cashed up users can invest in each other and immediately convert their cash into reputation.
  + I want it to be possible for two users with mutual respect for each other to boost each other’s reputation, but they should be punished for doing so to a blatantly unjustified degree.
  + Given how temporary the benefits would be of having two users invest unjustified amounts of zuth into each other, and given the associated loss of reputation (in real terms), and the tax costs associated with undoing the stunt, I don’t think it would be a very attractive exploit.
* I think the investment cap should be some small fraction of reputation
  + A user’s reputation is likely to be far higher than they would realistically want to invest in one item
  + Reputation represents something like a year’s worth of salary; this is not an amount one would invest in a single item
  + For extremely high reputation users, making the cap equal to reputation would be effectively no cap at all, whereas the cap would have real effect for users with moderate reputation but a lot of cash from successful investing.
  + I have no idea what would be a good fraction of reputation to use for the investment cap – perhaps it should even vary between item types, domains, and even specific items.
  + The range of values that seems realistic to me would be something like 1/1000 \* reputation to 1/10 \* reputation, but perhaps the useful range is even more arbitrary than this.
  + The more influence wealth (cash) should have in determining influence, the higher the fraction should be. That is because if the fraction is extremely small, everyone will be able to easily afford to max out their quota, leaving reputation to determine the relative influence.
  + The quota should also be a higher fraction of reputation if it is desirable for investors to have more choice about how much an item or decision means to them. If the quota is extremely small, there wouldn’t be much point not maxing out one’s quota and so the voting process would become one of yes or no, rather than how much.
  + As with every other arbitrary value, the fraction of reputation that determines investment quota should be selected by a continuous investment market – perhaps even many; one for each type of item or specific item that should have its own value.
* Having abandoned the square root function on reputation, the Black Swan problem is back
  + Someone like J.K. Rowling could have practically monopoly control over her domain – books and writing or something in her case
  + I can’t see a way around this now that I understand the inherent limitations to trying to force greater democracy on reality.
  + It is possible, however, that the kinds of entities liable to attract disproportionate attention – like works of entertainment – would not fit into the domains that are actually applicable to Zutha decision making. I can’t think of why there would be any decisions to make in Zutha that would be specific to the domain of books, for example.
  + The kinds of domains that will need to be created for isolated decision making will be such things as programming, military expertise, legal expertise etc.
  + Sigh… yes, those are all fields subject to the Black Swan, to some degree at least. Paul Graham comes to mind. I’m sure there are military commanders and lawyers who receive disproportionate attention as well.
  + I’m starting to reconsider this conclusion.
    - J.K. Rowling won’t really get more reputation than all other authors put together. Even in terms of revenue Harry Potter doesn’t trump everything else put together, though it would possibly be a non-trivial portion. I also think it highly likely that reputation would be spread far more evenly than income is.
    - Paul Graham even less so – he might be relatively popular as a hacker, but not in comparison to all other hackers put together by any means.
  + I’m a bit more optimistic now. Perhaps raw reputation will work quite well to determine influence after all.
  + As another safety net though, it should not be possible to have reputation generation occur in complete isolation – or even any isolation.
    - Any item that contributes to someone’s reputation should be open to investment by any user
    - Domain specific reputation will only be relevant for decisions relating to that domain.
    - Decision items will not be owned by any user, so domain-specific investment will not contribute to any domain-specific reputation
    - The reason this rule is important is so that an agent cannot sustain monopoly control over a domain
    - If a user gains a monopoly over a domain, it is essential that all items contributing to his reputation are open to attack by the whole zutha community in the case of the monopolist misusing his power.

# Mon 24-Jan-11

* Real money wage should be portioned by domain
  + E.g. 10% of Zutha income is divided among users by global reputation; 10% is divided among Zutha Contributors by reputation in that domain
  + This means that someone who has reputation in the domain Zutha Contributor, earns money for it twice: once via the 10% divided among all kinds of reputation, and once via the 10% divided among Zutha Contributors
* Zuth salary could be a fixed global total which is portioned by domain also
  + I think I’ve really struck something good here
  + The zutha community should determine a total rate of currency input
  + This fixed rate of currency input is divided among various kinds of reputation (or “reputation domains”).
  + E.g. The community might determine a total input of currency of 1 million Zuth per month
    - 50% of this might be divided among users by global reputation
    - 10% might be divided among Zutha Contributors
    - 1% might be divided among users by reputation in the domain Law
    - 2% might be divided among users by reputation in the domain Programming
    - Small additional amounts could be given to all sorts of domains like Law and Programming
    - The breakdown would not be selected by giving specific percentages to each domain, but by dividing the total input of currency among candidates, representing domains, weighted by the candidates’ worth
* Mercenary Military
  + Zutha could pay multiple independent mercenary military forces to serve it
  + A lot of the problems usually associated with the concept of a mercenary military don’t apply to Zutha
  + Perhaps the most significant difference is that it is impossible to attack Zutha itself – there is not central entity to attack
  + Even if there was only a single mercenary force, if they decided they wanted to stop serving Zutha there would be no way for them to force Zutha to continue giving it money; it would not be able to start a protection racket. This is because Zutha’s finances will be distributed across the whole planet and will be managed electronically.
  + If a mercenary force turned bad and started trying to extort money from people, Zutha could just hire one or more additional mercenary forces to take out the first one.
  + Mercenary military forces would be used to both protect Zutha communities from invaders, and police protection agencies and each other.
  + If Zutha needed to wage a large scale war, requiring the cooperation of all its mercenary forces, it would appoint temporary military commanders and give them temporary control of all the mercenary forces.
  + One of the problems often associated with mercenary forces is the difficulty of convincing them to keep fighting when the going gets tough
  + The thing is, though, modern western military forces are paid to fight. What distinguishes them from mercenaries?
  + A distinction has to be made between professional and mercenary forces. Professional forces are paid by a single entity, like their own government, and enter a contract. Mercenary forces are freelancers who accept individual contracts from different payers at different times.
  + What I really want is independent professional forces, rather than mercenary forces.
  + Zutha should pay ongoing salaries to its military forces. There should be many independent parts which receive varying resources depending on their reputation, as with everything in Zutha.
  + An overarching commander should perhaps be present continuously as well, in order to manage large scale organisation. The overarching commander should hold his position only so long as he retains sufficient support from the Zutha community.
  + It would be important to foster an attitude of loyalty to Zutha, rather than loyalty to one’s commander. If salary is dependent on individual reputation, I think this scenario would arise naturally. Soldiers and officers would compete alike for reputation.
  + If the top commander decided to try to lead the military away from Zutha and turn it into a conquering force, Zutha would stop funding that commander and anyone who supports him, and would continue paying those who continue to support Zutha – probably against the rebel commander for a while, if he behaves in a way unacceptable to the Zutha community.
  + Individual members of the military would be paid directly by Zutha. This would be another crucial difference from a standard military system. Normally, payment of individual soldiers is managed internally to the military establishment, so if a military force turned against the state that funds it, the state would cut funding to the whole military, including all the individual soldiers. Thus the soldiers would have no choice but to retain loyalty to the military establishment if they hope to get their salary back, whether this is achieved by the military reaching an understanding with the state, or by extortion, a coup, or military conquest.
  + Zutha might still control different military establishments in different regions.
  + Actually a likely setup might be many small military teams all over the world, with higher level commanders assigned from among the Zutha community and located potentially anywhere in the world. What I’m beginning to envision is a distributed, but intimately connected military system and one which is fundamentally integrated with the Zutha community.
* While a long military operation is in progress, many temporarily secret keys should be used for different levels of security, and they should be changed regularly so that blocks of military data in time and category can be released to the public as soon as they no longer need to be kept secret.
* Group selection of defaults
  + E.g. Zutha community selects a default retirement plan by standard investment market
  + Zutha would not be in the business of forcing people to do anything, though, including using a retirement plan
  + The way such default selections are likely to be manifested is as recommendations to individual employers and providers of services about how to select their defaults to best satisfy consumers.
* Private communication could be extended to beyond the case of Zutha military commanders managing a war
  + It could be useful for groups to be able to have private conversations on Zutha, using Zutha infrastructure of semantic organisation and reputation
  + All that is needed is for each message to be encrypted with a random symmetric key, which is then encrypted with each of the public keys of the members of the private group
  + Any of the group members could release to the public an arbitrary subset of these messages
  + Any group member could be excluded from future discussion by the other members agreeing to stop encrypting future messages with the excluded user’s public key
  + The fact that the messages are stored on Zutha as they are created would act as a strong form of digital timestamp. It would be impossible for a group to modify what they said in the past over the Zutha network. They could reveal things they’ve said in the past on Zutha though, and everyone will be able to trust that they actually did say that then.
* Example applications of secret group discussions on Zutha
  + Authors collaborate on writing a novel.
    - They don’t want the plot to be leaked to the public, but they want to be able to invite new members in to contribute once they’ve established they can trust them.
  + Terrorist communication, unfortunately
    - I wonder if it would be possible to force all secret communication over Zutha to be temporarily secret
    - I don’t think so. It would always be possible to encrypt messages with public keys whose counterparts are fully secret.
    - Zutha wouldn’t really be adding a significant service to terrorists since they would be able to communicate directly using encryption anyway.
    - Then again, it is possible that some terrorists at least would not be able to provide the infrastructure Zutha provides for making users truly anonymous
    - The anonymity provided by the distributed architecture of independent hosts would be the real service Zutha would be providing to terrorists. Whether Zutha conveniently integrates the secret conversation feature, terrorists could easily encrypt messages themselves. Mind you, the fact that the messages are encrypted would only be concealed if they were allowed to add legitimate encryption over the top.
    - Anonymity could conceivably be granted only conditionally, and could be revokable. A single agent (like a government) may not be able to hack enough hosts to find the trail to a terrorist’s location, but the distributed system as a whole has that information. If Zutha commanded all the hosts to release the relevant information… All that would be required to expose the physical location of an anonymous user (or at least the history of locations from which he communicated with Zutha) would be a sufficient consensus from the Zutha community.
    - If encryption were not allowed at all over the Zutha network, then it could be required that anyone using encrypted messages would have their location data exposed. If secret communication was allowed, though, even if there was a way to force all encrypted messages to be temporarily secret, it would be easy for terrorists to encrypt their messages using their own encryption key, and then add the temporary encryption over the top.
    - I don’t think there is any way around this. If I allow any encrypted data at all, it will always be possible to conceal encrypted data in it which Zutha can never decrypt.
    - I’m not optimistic about this, but it might be possible to design a temporary encryption system in which it is not only possible to verify that a message is encrypted with a particular temporary encryption key without decrypting the message, but possible to check that, after decrypting the message, the result will be plain text.
  + Companies could set up entire private intranets on Zutha
    - They would be able to tap into the full semantic architecture of Zutha including its powerful querying capabilities and distributed processing power
    - Private company items could be arranged in semantic relationships with public items
    - Some sections of the company’s intranet could be made public, such as discussion forums about products they provide
    - Employees would have company-specific reputation, but would be able to accumulate cash anywhere on Zutha
    - For items in which company-specific reputation is very important for making a decision, the decision item could have its investment cap set very low, such that cash is not a big factor.
    - For most cases, though, the ability of an employee to accumulate a lot of zuth (cash) across the whole Zutha ecosystem would be a good indication of his value as an employee, and thus he would deserve the additional influence he would have by being able to invest more in more company items.
* Intelligent decentralized real-time prioritization of information for military commanders.
  + The Zutha community would be extremely effective at filtering out the most important information from the massive stream of data coming in during a war
  + Individual users would have incentives to find pieces of important information and invest in them with the hope of making a profit when others discover the same piece of information, agree that it is important, and invest in it also
  + These incentives would lead to swarms of people scouring the inflow of data for useful information, writing comments, summaries and reports of the important stuff, making suggestions, investing in the best comments, summaries, reports and suggestions, and the most interesting satellite images etc.
  + This same decentralized data mining process would also be incredibly effective during natural disasters and any other scenario where important information needs to be extracted from a huge amount of data.
* Why won't zutha society result in ubiquitous permanent virtual reality?
  + This issue came up when I was thinking about ideas for my book The War of Mortazael (which has become a collection of books now). I considered that a system of government (like the ones in the current richest societies) which has incentives to provide welfare to the least productive citizens, would be prone to leading to a society in which most citizens live their lives in virtual reality with the small (due to super advanced technology) costs of powering their virtual lives paid for by a welfare government.
  + Absent competition from another more effective society, such a welfare state would be able to create enough wealth to support their population of virtual citizens with dedicated artificial intelligence that has no desire (or at least no power) to pursue its own self-interest (perhaps because it has none), but instead exists only to produce energy and advance the technology underlying the virtual reality.
  + I hypothesize that if a more “effective” society like a Zutha society came onto the scene, it would outcompete the welfare state and probably lead to its annihilation or something close. This is because
    - a) The Zutha society would be competing for the same limited resources as the welfare state and would be far more competent at getting them
      * If it came to war the Zutha society would win),
    - b) The welfare state would be abhorrent to the Zutha society for the same reason the prospect of the whole human race living permanently in virtual reality appals many people today.
      * We have evolved through a competitive mechanism (evolution) and we have inbuilt alarm bells that alert us to scenarios that are evolutionarily uncompetitive.
      * Our abhorrence of murder, extortion, breaching contracts, slavery, dictatorship, torture etc. are all such alarm bells. These things reduce the competitiveness of the society they occur in for various reasons, one of the common ones being that they cause a net loss of wealth by taking more wealth from one entity than they give to another.
      * Just as people today are disturbed by behaviours which make a society uncompetitive because having that reaction lead to an improvement in mankind’s competitiveness in evolution, a competitive advanced society like a Zutha society would have members with similar negative reactions to uncompetitive social behaviours – precisely because having members with such reactions would make the competitive society more competitive.
      * Just as people in democratic capitalist societies were horrified by the Soviet Union and wanted to destroy it (not the people but the system – though I’m sure many people did apply their abhorrence misguidedly to the people in the Soviet Union), a Zutha society would want to destroy an uncompetitive technologically advanced welfare state.
  + The reason a Zutha society would be far more competitive than a welfare state is because it produces the right incentives for independent intelligent beings to innovate in pursuit of their own self-interest.
    - In order for an intelligent entity to achieve its own self-interest in a free society, it is likely to want things that other intelligent beings have, and in order to get these things, it will need to trade something valuable, which it will need to produce.
    - To put this in less abstract terms, members of a Zutha society have to give in order to get.
    - The process of unimpeded competition is extremely effective at producing what its members want because it rewards the members who are most effective at producing what other members want.
    - Now for a society to be competitive in an evolutionary context, one of things its members are going to need to want is survival. So one of the things this distributed innovate engine is going to be working hard on is how to defeat other societies who will try to destroy its members. The welfare state will see the Zutha society as a threat, just as the Soviet Union saw the United States as a threat, so it will try to destroy the Zutha society. Thus the Zutha society will fight back and win.
  + So why won't zutha society result in ubiquitous permanent virtual reality?
    - Because intelligent beings living permanently in an isolated virtual reality would not have their energy focussed on producing things other beings want in the real world
    - Sure beings could produce virtual wealth and trade it for any virtual things they value. They could also trade wealth in the virtual world for resources in the real world, but crucially, for the latter to happen, there needs to be beings producing the wealth in the real world which can be traded with beings in virtual worlds for virtual wealth.
    - If there is trade going on between beings in virtual worlds and beings producing wealth in the real world, then this is not ubiquitous permanent virtual reality. I’m not trying to make an argument that there would not be some, or even many, beings that would opt for permanent virtual reality in a Zutha society.
    - My hypothesis is that a Zutha society would not produce a scenario in which all the interesting stuff and all the most complex and intelligent beings would reside in virtual reality, leaving boring robots in the real world to maintain the servers.
    - I’ve realised I need to add a qualification to this claim: it doesn’t apply to just any Zutha society but only to a competitive Zutha society – that is, one which has survived, or would survive, in a competition for survival between societies. Actually I really need to adjust my claim further: there isn’t anything specific about Zutha societies in this concept; the fact that the society is competitive is the key.
    - So, in a competitive society, a significant proportion of its resources – crucially including its most potent assets: the independent intelligent beings that compete with each other to most effectively satisfy each other’s desires in exchange for things of value – will be invested in the real world. That implies that a large part of the complex, independent intelligent beings with diverse wants and drives, will be investing their energy in producing wealth in the real world and will not exclusively dwell in virtual reality.
    - Let’s illustrate by a counterexample.
      * Suppose that in a competitive society, A, all the most intelligent beings live in isolated virtual reality.
      * In a different society, B, most of the intelligent beings live and invest their energy in the real world.
      * One of the primary things of value to beings in society B is survival. Survival can be thought of as an economic “good” which is highly valued by members of society B.
      * The “good” of survival can only be satisfied by innovation in defensive technologies in the real world
      * Since survival is a highly valuable good, a lot of resources will be invested by the intelligent beings in the real world to provide that good.
      * Since society B has more intelligent beings investing more resources in defence than society A, society B will defeat society A in a competitive context. Thus society A is not a competitive society.
    - One might argue that society A could have extremely intelligent philanthropic robots working very hard to develop powerful technologies of defence and that if it starts as a larger and more powerful society than B, it may still be able to defeat it.
      * My rebuttal to this would be that intelligence is far more potent than existing raw resources and society B would be orders of magnitude more intelligent than society A
      * Society A is wasting (with regard to real world defence) the most potent source of intelligence available: the emergent intelligence of independent beings with independent desires competing in a competitive market for their own self-interest.
      * The philanthropic beings society A uses to provide for its virtual reality dwellers could be individually uber intelligent, but without the incentives to innovate provided by competition, their collective intelligence would be insignificant compared to a whole marketplace of intelligent beings with those incentives to innovate.
* The next singularity will not be caused by the creation of human level AI, but the unleashing of the first distributed super intelligence i.e. Zutha.
  + Extremely intelligent AI will not in and of itself be a potent creator of wealth without effective incentives to create wealth
  + The previous singularities, agriculture and industrialization, are distinct from other significant technological developments because they were not innovations, but paradigm shifts in terms of the architecture of competition.
  + Artificial Intelligence will continue to be a technology, and improvements in it will be innovations. The point at which AI reaches human level intelligence will not be a paradigm shift in the architecture of competition, but merely a continuation of the progress of innovation within the existing paradigm.
  + The effectiveness of AI in producing wealth will be dependent on how well we produce the incentives for it to choose to invest its energy in producing wealth
  + These incentives would be best found by a competition for survival between many intelligent beings, including AIs. The most effective way to survive would be to innovate to accumulate resources. This would involve making allies, avoiding the animosity of powerful beings, providing value in return for physical resources etc.
  + An AI that behaved undesirably by the judgement of the collective society would be destroyed.
    - For example, if an AI ended up with the desire to wipe out the human race, this would conflict with the desires of other powerful AIs who have developed a survival strategy of giving beings (including people) what they want in return for resources, as well as, of course, all people.
    - Initially at least, mankind would be powerful enough to destroy any beings that threatened their existence, so the AIs that survive will be the ones that have a psychological makeup which causes them to please humans.
    - Also, human societies that manage to create AIs that are friendly to them and hostile to AIs that are hostile to them, will be the societies which will survive.
  + Thus no central planning would be required to figure out how to inject incentives into AI that would be friendly to human welfare – AI with such incentives would emerge through competition in a free market.
  + A super-intelligent AI residing in a society without competition, like a socialist state, would likely be very bad at producing wealth
    - In such a society the powers that be are likely to stifle competition against their precious super AI, and try to develop, through central planning, an incentive system for the AI which will lead it to do what’s in their interests.
    - This will likely take the form of a perfectly obedient AI which coldly calculates how to most effectively carry out the orders it is given.
    - In this case, the super AI would magnify the destruction of wealth effect of the power monopolists. Independent thought would be more effectively suppressed and thus innovation would be more effectively stifled.
  + This same effect is true of many other innovations/technologies: when used in a wealth creating context like free competition, they magnify the creation of wealth, but when used in a wealth destroying context such as a suppressive dictatorship, they magnify the destruction of wealth.
  + The point is that it will not be an improvement in the power of the tools used to generate wealth that will trigger another singularity; the development of the tools is endogenous to the wealth generating system. Rather, it will be an upgrade of the wealth generating system that will result in the next singularity.
  + The industrial revolution was an upgrade of society’s wealth generation paradigm from feudalism and monarchy to capitalism. This upgrade represented a drastic increase in the freedom of individuals to pursue their self-interest by mutually beneficial trade of goods.
  + The conversion of society from a system of representative democracy – with all the rent seeking and arbitrary restrictive legislation that goes with that – to a decentralized, libertarian, reputation oriented organisational paradigm, will be the next great paradigm shift

# Wed 26-Jan-11

* You should be able to buy reputation by donating to popular causes because people will invest zuth in the action if they believe it deserves praise
* How to stop slavery being possible in a Zutha community?
  + Slaves would not have paid for protection
  + Philanthropic protection agencies could provide blanket protection for some basic human rights, and Zutha could provide some funds to such philanthropic protection agencies
  + As with other public goods, Zutha should contribute some resources according to reputation, with additional funding coming through direct donations

# Thu 27-Jan-11

* It could work to have crowd decision making be a prediction market
  + Consider a decision about what Zutha’s tax rate on selling investments should be
  + The decision market for this decision could be set to expire on a particular date
  + On the expiry date, the decision/prediction market is frozen, the Zutha tax rate is selected as the median rate as determined by the prediction/decision market, and the winnings are divided up among those who invested in the selected rate
  + This is clearly a self-referential market: investors influence the outcome by predicting it
  + I think such a system will still output valuable results though
  + An individual investor should have insignificant influence compared to all other investors in the decision market combined – this is crucial, otherwise a Bill Gates investor could overwhelm other investors and force the selection of the tax rate to be within the range he invests massively in.
  + Provided no individual investor has statistically significant influence, any individual investor is forced to predict what the majority will predict. Since investors are independent in the sense that they do not communicate en masse (and are thus unable to operate as a single entity), this entails predicting based on the status quo criteria.
  + The status quo criterion will be what decision is in the best interests of Zutha. The import of this status quo criterion will be strengthened by the fact that a lot of people will invest with that goal in mind, rather than purely to make profit by predicting other people’s guesses.
  + Thus as an isolated individual trying to make money on the decision/prediction market about the Zutha tax rate, the best way to achieve that goal would be to use all information at your disposal to predict the best rate by the criteria of maximising benefit to the Zutha community.
  + I predict, therefore, that this mechanism will have the same merit as any other prediction market in achieving its objective. As with any prediction market, if there is a systematic bias among investors, the aggregate prediction/decision will be biased in that direction. The only way to reduce this flaw is by trying to make investors as independent as possible.
* A prediction/decision market about the Zutha tax rate (or other things of course) could be complemented by a prediction market about its affects
  + The prediction market would be 2-dimensional, with selected tax rate on one axis, and resulting Zutha wellbeing (perhaps measured in income) on the other axis
  + The tax rate axis would be a conditional variable in that only the currency invested in the chosen tax rate will be divided among winners, whereas currency invested in sections of the tax rate axis which do not represent the chosen tax rate will be liquidated
  + The tax rate axis will need to have a resolution, rather than being perfectly continuous, because if the chosen tax rate is an infinitesimally small point on the axis, then there will be no currency in this range to divide up among winners. If the chosen tax rate is represented as a range on the tax rate axis, however, then there will be a positive amount of currency invested in that range.
  + If we calculate the weighted median (i.e. the predicted value) of Zutha wellbeing at each possible increment of tax rate, we will have a relationship of chosen tax rate with Zutha wellbeing. A graph of such a relationship will clearly reveal a maximum point representing the best possible tax rate for maximising Zutha wellbeing.
  + The recommended tax rate from this prediction market will be a major piece of information in the decision/prediction market responsible for actually selecting the tax rate. This information will be known to all, and it will probably be such a swaying factor in people’s investment decisions that the chosen tax rate will inevitably be exactly the same as the one that is predicted to maximise Zutha wellbeing.
  + If someone with a lot of resources wanted to skew the tax rate to their preference, say towards zero, they would invest a huge amount at zero
    1. The weighted median of investments would shift slightly towards zero
    2. Investors, wanting to ensure they have an invest covering the winning selection, might shift their investments ever so slightly lower, to make sure the range they have invested in still covers the slightly lower weighted median
    3. Thus the impact of the powerful investor will be positive, but probably extremely small because his market power will be insignificant compared to the collective.
    4. Most investors will want to keep their prediction as a narrow range around the value recommended by the Zutha welfare prediction market, because that will be the most likely value to win, and the less they spread their bet, the more profit they make if the winning value falls within that range.
* Why not have the Zutha tax be selected automatically as the maximum of the graph of tax versus Zutha wellbeing as specified by the prediction market?
  + One disadvantage to doing this would be that it would reduce the range of tax values that would be invested in, because the wellbeing prediction market would be purely self-referential. This would be undesirable because less information would be extracted from investors about the relationship between tax rate and wellbeing. The graph is likely to look more like a blog at the predicted tax value and corresponding predicted wellbeing value, as opposed to a curve with a maximum at the best tax value and highest wellbeing value.
  + Another factor is that other prediction markets may not have such a clear cut way to determine the best course of action from the information provided
    1. For example prediction markets could be used to predict the number of deaths over various time ranges conditional on the decision to invade a country to topple a foul regime.
    2. It cannot be determined programmatically from this information whether the best thing to do is invade the country
    3. Thus a separate decision/prediction market would be used to make this choice
    4. Many other prediction markets may be helpful for providing relevant information about the decision to invade, but it would not be obvious (especially programmatically) what choice should be made based on all this information.
  + There may be some cases (such as selecting the Zutha tax rate based on predicted Zutha wellbeing) where the best decision is clear enough from the prediction market to determine programmatically, but given that such cases are likely to be the exception, I think the better approach is to consistently require an independent decision to be made based on all available information, including the relevant prediction markets
* If a powerful agent got in early in a decision/prediction market and skewed the weighted median to his personal preference, this could have a significant effect on the final outcome, because later investors would see the existing apparent “consensus” and want to jump on the band wagon rather than risk investing in what they think is a more sensible choice when they don’t think there is much chance of the consensus shifting to their view
  + I think it will be enough in most cases that a significant enough proportion of investors will be investing to influence the decision rather than make profit
  + Advantages to making decision markets operate as prediction markets and distribute winnings
    1. If you are a powerful investor and want to skew the results, you pay a cost if you fail, whereas if investments were liquidated after the decision is made, it would not cost anything to invest a huge amount to sway the decision
  + Disadvantages
    1. The powerful investor will have more effect per unit of currency invested because of the psychological effect on other investors
  + There is an incentive to invest in a decision outcome that you believe deserves to be more greatly represented, because if other people agree with you and do the same, you can profit by selling your earlier investment, even if the consensus never actually moves to that outcome
  + I think this effect might be enough to push the apparent consensus towards the real consensus, because if enough people invest in an undervalued outcome because they think they can make profit by selling out before the decision market expires, then eventually that once undervalued outcome will be valued highly enough that people will feel more confident investing in it with the hope of actually picking the winning outcome.
  + There are about 4 forces working against someone who wants to distort the outcome of a decision market
    1. Other users wanting to distort the market in an opposing direction
    2. A large bulk of users wanting to actually pick the best outcome for Zutha
    3. The incentive to invest in an undervalued outcome (by the criteria of what you think others will think is the best outcome for Zutha) because you expect to be able to sell the investment later for a profit once other people act on their own believe that the outcome is undervalued
    4. The incentive to invest in what you think others will think is the best outcome for Zutha because you predict that the above 3 forces will eventually drive the weighted median to that outcome, and you will win a proportion of the winnings if you hold an investment in that outcome
  + I think the combination of these forces will be sufficient to prevent significant distortion of the market by powerful agents
* Ideas to integrate game concepts into Zutha (as per Freedom TM)
  + Certain realms of influence in Zutha could require the completion of quests
    - The quest would be created by a team of users with the relevant kind of reputation
    - They will design the quest as a good test of relevant skills to hold a particular responsibility
  + People could be tested in simulations (ranging from computer games to virtual reality to real world tests) for various mental and physical competencies
    - Judges and facilitators could be selected randomly from a limited pool of available users with the relevant reputation for those roles
    - Judges would rate competitors in a test on factors that a computer could not measure
    - Some attributes would benefit from a combination of statistics processed programmatically and human opinion
    - For example, to measure a person’s ability for creativity under stress, relevant information would be simplistic measures of stress via sensors, fuzzy measures of stress from human judgement, statistical rate of success in achieving objectives under stress, and the judgement of human observers regarding how creative a competitor was in solving problems he/she faced while trying to achieve those objectives
    - Facilitators would control various factors in the test environment to make it more effective
    - For example they may permanently or temporarily control Non Player Characters, modify weather conditions, cause equipment to fail, create objects in convenient or convenient times and places etc.
    - This paradigm would be very powerful for guiding events toward greater entertainment value in MMORPG games
    - Facilitators would be rated on their performance by other users so they would have an incentive to do their job as specified and a disincentive to be destructively playful, arbitrary or unfair
  + Zutha could fund and use its decentralized resources to manage MMORPGs
    - The underlying software of the MMORPG would be developed OpenSource
    - A subset of high reputation hosts would volunteer to host the game, and pledge secrecy about secrets of entertainment value
    - Groups of users would be selected (or volunteer) to develop certain aspects of the game content in secret
      * Some aspects of content creation would need a degree of top-down management to prevent contradictions and clashes of various kinds
      * Development of the background history and personalities of existing characters would need such central management to maintain consistency
      * Some teams could work away independently and create content that is secret to everyone else, including the top down management
      * Any content that does not need to be accounted for in the background history could potentially be generated this way
      * For example, items and characters could be created that have no history outside the real-time progress of the game.
      * A team of contributors might collectively invent a cool character that they develop from an early age and lead through adventures which they manipulate to lead to desired character personality and history.
      * A team of contributors might take an existing talented blacksmith and have him create some awesome weaponry which they then secretly arrange to have taken to places and times that would add entertainment value to players
      * A team of contributors may take control of some NPCs in a battle to make it more interesting
      * After such teams of contributors reveal their involvement in content creation which has been revealed to the public, other Zutha users may grade their performance, providing an incentive to produce good content
    - Players would be charged some rate to play. I would prefer a very low per-hour rate to discourage extreme addiction, but there might be more optimal payment systems
    - Profits would go to Zutha and be divided up among its various owners like any other income
    - If an MMORPG increases in popularity and profitability, there would be an incentive for users to vote to provide its various components (like hosts and contributing users) with more of Zutha’s resources to do more of what it is doing, because that would lead to greater income for the whole Zutha community
* It will be essential to start Zutha off with a large amount of diversity of world views and opinions
  + I’ve been listening to an EconTalk podcast with Cass Sunstein which reminded me of the vulnerabilities of prediction markets and group discussions to polarizing towards an existing bias
  + I think the most crucial time will be the very start – the birth of Zutha, as it were – because the initial high reputation users will be the ones to choose the subsequent high reputation users
  + If Zutha starts with predominantly people of libertarian sympathies (for example), then the community is likely to remain biased toward that paradigm because the bulk of power in the community will lie with libertarians who will of course invest more reputation in libertarians than in socialists
* There needs to be a distinction between the kind of ownership that earns reputation, and the kind that earns money
  + For example having an ownership stake in Zutha would imply that you receive a proportion of Zutha’s income. There might be a set of charities that receive money from Zutha, for example. These charities do not deserve reputation from the worth of the Zutha item, though, as they have not contributed to Zutha, they are merely beneficiaries of its generosity
  + I think the key here is that Zutha’s revenue should not be divided up based on a concept of ownership, or at least not the normal kind of ownership
  + There are some entities (like myself and other contributors to Zutha’s development) that would deserve actual ownership of Zutha in the way that earns both income and reputation through the Zutha item
  + Entities which Zutha donates to would not be partial owners, though, but something else… “beneficiary” ?
  + Every item should be able to have 0 or more owners as well as zero or more beneficiaries
  + Reputation is divided among owners only
  + The owners of the item should collectively determine the percentage of their real money income that they donate to beneficiaries
  + This percentage is divided proportionally among beneficiaries by their relative stake as beneficiaries in this item
  + The remaining percentage is divided proportionally among owners
* How should owners and beneficiaries be assigned for the Zutha item?
  + Zutha should be owned by users who have contributed significantly to it
  + Some of the owners will be individuals who stand out as major contributors, including myself
  + Certain groups of users will also have a partial ownership stake
  + For example, there will be reputation domains like Zutha Host, Zutha Contributor, Zutha Code Contributor, Zutha MMORPG Contributor; the groups of users with reputation in these domains will have partial ownership of Zutha
  + The owners of Zutha will collectively choose a percentage of income to donate to beneficiaries
  + They will also collectively specify the beneficiaries and their relative stakes
  + Beneficiaries will probably include the groups Universal (representing any user with reputation in Zutha), Philanthropic, Zutha Military, and probably many individual projects the Zutha owners want to provide additional support to because they believe their work will contribute significantly to Zutha’s interests
* Zutha owners could make use of prediction markets to make decisions in their own best interests
  + For example, they could make a prediction market to predict their own long term income for various levels of giving to beneficiaries
  + The prediction market would presumably take into account the effects of giving on Zutha’s reputation and attractiveness to new users, and its ability to spread as a paradigm and thus earn more mindshare across the world.
* Owners of Wiki articles should be the Zutha items corresponding to the Wiki site’s users that contributed to the page
  + Once a Zutha user claims ownership of a Wiki site user, the Zutha user can acquire the reputation and money associated with the Wiki site user.
* It would be very useful to be able to know who leaked a document from a private, encrypted repository
  + There would need to be a way to embed information in the document itself which is unique to the decrypter and which is difficult to identify as the tag so that it cannot easily be removed. This way, if the information is leaked and Zutha users find the leaked document, they can identify the traitor
  + I don’t think this will be possible. A large part of the information that is to be kept secret will be plain text. Even if all messages were encoded as images with hidden tags embedded, it would be relatively easy to extract the plain text and share that.
  + It will of course be possible to narrow the possible traitors to those in the private group.
  + If a group is compromised, the group should be disbanded and a new one formed to take on their role
  + The members of the compromised group would then be under suspicion, but not automatically condemned. They would be allowed to join new secret groups, but no two members of a compromised group would be allowed into the same group again.
  + The more compromised groups a user has been a part of, the more suspicion would mount against him. Provided traitors are relatively rare, being part of just 2 or 3 compromised groups where all other members are clean would be very compelling evidence that you are a traitor.
  + I’m thinking here in terms of a Zutha intelligence service. This process would not really work for a company since they cannot easily disband and start again with a new group of employees.
* There need be no central command of a Zutha intelligence service
  + Intelligence should be reported in small pieces to many different groups
  + These groups would discuss their segments of intelligence in secret and perhaps share insights or specific documents or facts with other groups they think might benefit from them
  + The main aggregation of information would happen through prediction markets
  + There will be a balance to find between allowing information to spread enough that it is known by enough users to have its influence on the outcome of relevant prediction markets, and restricting the spread of information enough that individual powerful ideas do not dominate people’s opinions and cause a distortive bias.
  + Additional users could be brought into a secret discussion or provided access to specific secret documents simply by encrypting the relevant documents in his public key
    - Secret documents should always be shared in this way so that it is known who has had access to that information
    - This will be crucial for the identification of traitors (and sabotaged or careless people)
* One of the main pieces of information that needs to be kept secret in an intelligence service is the identities of agents. How would this issue be dealt with by Zutha?
  + People that want to be Zutha agents would create an anonymous user and apply in a generic way to be a source of secret information
  + These volunteer agents would be automatically assigned to a group of automatically selected information managers
  + The volunteer agent should be able to veto any of the selected information managers if he is concerned by his safety and doesn’t trust the user in question
  + The agent would then report his secret information to the secret channel that is only accessible to himself and the selected information managers
  + Because information managers are selected randomly, even those who feel traitorous are unlikely to know who to sell the information they receive to
  + If an enemy was looking to infiltrate Zutha’s intelligence network they would find that there is no one to target; no individual or even small group knows a significant amount about the entire spy network or the information it knows.
  + Of course, an enemy can look at the information coming out of the prediction markets like anyone else can, but prediction markets are likely to generate advice to Zutha that no individual person can understand well enough to counter.
  + The output of the decision markets would be like the moves on the chessboard; they wouldn’t necessarily give insight into the workings of the mind behind the moves.
  + A powerful entity might be able to plant a few high reputation agents in Zutha who manage to get invited into lots of secret information management groups, but given the randomness of the selection, these agents will only randomly stumble across information of particular interest to their sponsor.
  + A lot of information will enter Zutha’s distributed mind that does not come explicitly through declared agents.
    - Many loyal agents of intelligence services around the world are likely to be tempted to profit from their knowledge by investing in relevant prediction markets on Zutha with anonymous users. They would be very safe from being caught doing this, since it would not even be clear what information they had used to make their predictions, but their knowledge would be encapsulated in the prediction and would add to Zutha’s knowledge
    - A huge amount of useful information will be aggregated just from ordinary people on the ground where interesting events happen
* In some continuous variable prediction markets the winning point should be the highest point, while in others it should be the weighted median point
* Zutha need not try to explicitly hide any information from its enemies since none of its enemies will be capable of processing all that information
  + I’m assuming that none of Zutha’s enemies would be distributed entities like Zutha itself. For the short term, I think, this assumption will be born out; Zutha will face mostly bureaucracies and dictators during its early years. Eventually, though, malevolent distributed entities might arise that do not value the wealth of humanity. I don’t think such entities are likely to arise until we start producing serious AI and manipulating human nature, though, which will be quite a while away, I think.
  + All the discussions of Zutha users about how and why Zutha should do this or that will be laid bare for enemies to read, but there will be so much information that no individual or bureaucracy will be able to understand what it all implies about what the aggregate mind is thinking, and they will not be able to predict long in advance, the decisions that aggregate mind makes.
  + It might be that the outcome of a critical decision market is looking very clear cut, and the knowledge of this very likely decision by Zutha leads one of its enemies to make a particular decision in response. That enemy will probably find that Zutha can change its mind very rapidly in anticipation of the decision the enemy makes in response to the current apparent state of Zutha’s mind.
* When people create accounts on Zutha, they should be encouraged to create as many secret questions as they can be bothered making
  + In the case of a dispute over ownership of an account, the person who can answer more of the secret questions (provided the margin is significant) should be granted access to the account
  + Most of the time, you would use existing OpenID accounts or security devices to instantly authenticate yourself, but if some of those were stolen and another user took over your account, you would regain control by answering lots of security questions and then link the account to a new set of OpenID accounts and security devices.
  + I think secret questions, provided they are numerous enough and of good enough quality, will eventually be a more powerful security mechanism than biological identifiers. It will likely be relatively easy eventually to steal information (with or without their knowledge or consent) about people’s DNA, retina, fingerprints etc. but to extract knowledge from their mind will remain considerably more difficult for a long time, I think.
* I think my investment model will be able to produce useful results in prediction markets that have no expiry and no way to measure the correct outcome
  + Consider an ongoing prediction market to read someone’s mind
  + The person could be being interviewed on video with the video streamed out to the Zutha community
  + There could be items representing such things as whether the person is currently lying, feeling guilt, sadness, anger etc. or aiming to deceive or thinking of a particular person or place… the list of possibilities is endless
  + The candidates in this prediction market would not have expiry triggers; they would never end with the winnings distributed among correct predictors. Instead, they would be like any other item with no tangible underlying value, only the aggregate perceptions of investors guided by a fuzzy, status quo metric for the value of the item.
  + Consider an investment market for whether an interviewed person is lying or not
  + When a user A first becomes convinced that the person is lying, A would invest as soon as possible in the item that represents that state. If A’s perceptions are better than others’ then others will catch on after A and invest after him. As soon as A believes the person has stopped lying, he will want to sell as soon as possible to beat the stampede of sellers triggered by the realisation that the person has stopped lying.
  + There would also be an investment pool representing the case that the person is not lying. This pool would presumably behave in an inverse manner to the positive pool. As the person stops lying, investors would sell as quickly as possible from the positive pool and invest as quickly as possible in the negative pool.
* Users who own items that are earning money should be able to have some or all of that income confiscated if they have a negative reputation
* All view requests should eventually be propagated to all hosts, but the path that leads between the client and the hosts responsible for processing the request should be prioritized.
  + Hosts will have a private database linking host and client IDs to the neighbouring host which appears to be closest to that target
  + This database is formed by taking the neighbouring host with the smallest average number of hops from a given target/source host/client.
  + Every message carries with it the number of hops it has taken so far. This is the information used by each host to profile its neighbours about their relative distances from a given client.
  + The term distance as used here does not mean geographical distance. Hosts probabilistically find an optimal connectivity pattern with each other which will be only slightly related to geography.
  + When a host receives a view request it will send the request on to a small number hosts immediately with high priority. The hosts selected here will be the ones with the smallest distances from the target hosts of the view request. The receiving host may itself be a target host in which case it sends the messages on toward the other target hosts.
  + During times of low load, hosts should send out all their view requests to all their (probabilistic) neighbours and process the unprocessed view requests they have received that didn’t belong to them.
* I predict that, if I manage to get people from enough different paradigms devoted to Zutha early on, eventually people with all sorts of different world views will start arguing with each other about why Zutha works. Ultimately I hope they’ll discover that they’ve all been striving for roughly the same thing all this time but just had very different ideas about why we aren’t there yet and what we need to change to get there.
  + Libertarians will argue that Zutha works because it allows the distributed intelligence of the market to work unconstrained by arbitrary regulation
  + Anti-consumerists will argue that Zutha is good because it recreates the sense of community that brought people together in humanity’s past. They will like the way it encourages people to be generous and they’ll probably call it “giving something back” to the community
  + Marxists will praise the way Zutha gives power to the producers of wealth – the workers – and takes it away from big exploitative companies
  + Democracy advocates will approve of the way Zutha puts power directly in the hands of the people, rather than through representatives who get to power through selling political favours to powerful rent seeking companies. Hopefully such democracy advocates will realise that the weighted nature of influence in Zutha is actually like a more subtle representative system, though it is a recursive representative system in which the base unit is the amount of value an entity offers to society rather than individual people. The superiority of this system will become more apparent, I think, as the definition of a citizen, and eventually of a person, becomes more and more blurred.
  + Capitalists will argue that Zutha is effective at generating wealth because it allows the creators of wealth to profit as much as they deserve to (as judged by the free market) from the generation of that wealth.

# Fri 28-Jan-11

* Distributed electricity.
  + The electricity grid is literally a grid, probably underground which has many inputs and outputs in many places in its 2-dimensional expanse
  + The potential difference is formed between the grid and earth, as with most current electricity distribution systems
  + Individual nodes in the grid may be either net consumers or net producers of energy at any time
  + Metering is done at each node
  + Multiple independent companies should have meters at every node
  + This redundancy will make it impossible for a metering company to report incorrect results either deliberately or accidentally.
  + Who pays the metering agencies?
    - There would be multiple competing distribution companies
    - A distribution company would find a balance of customers such that the net inputs and outputs of energy average to zero
    - The distribution company would have an incentive to get accurate metering so that they can manage their books correctly to maintain the right balance of customers and charge and pay them correctly. Therefore they would have an incentive to pay multiple metering agencies to report the flow of electricity at each node in order to ensure that what they report is correct.
    - A metering company would have an incentive to take a bribe from an electricity producer to exaggerate their production and vice versa with electricity consumers. If a distribution company gets incorrect data about how much electricity is being produced or consumed at a particular node, it must pay for the error itself.
      * Let’s say a net electricity consumer gets their meter modified to report less electricity consumption than they actually consume
      * This will mean more electricity is being consumed in aggregate than is being paid for from the perspective of the distribution company
      * The electricity still has to be produced and the distribution company has to pay the producers for it, but some of what they buy is being stolen by the consumer with the modified meter.
    - Why wouldn’t the distribution company do the metering itself?
      * The distribution company’s customers would change over time. They wouldn’t want to go to the trouble of installing a meter at a customer’s address if that meter might be useless to them in a few years’ time when the customer changes distribution companies.
      * Specialisation improves efficiency. Metering companies would be better at metering and would be able to do it more cheaply.
      * The same risk of bribery exists when the distribution company does its own metering. A customer can bribe the installers of the meter to modify its calibration slightly in their benefit.
      * It would be extremely difficult for a customer to simultaneously bribe multiple different metering companies (or their relevant employees) to change the settings on their meters at exactly the same time so that no discrepancy is every detected.
  + A case study
    - At a particular node, 4 metering companies have already set up metering hardware
    - A distribution company wants to pay for the services of 3 of them for redundancy
    - The metering agencies will compete to outbid each other until they reach prices close to their costs
    - The distribution company will select 3 of the metering companies based on their perceived value/cost. He will probably take their reputation into account to estimate their value.
    - The metering company with the worst perceived value for money will not be paid for its investment in metering infrastructure at the node in question. If this happens to it consistently in enough areas, it will go out of business.
    - Another company may come along and purchase some or all of the failed metering company’s infrastructure and make profit out of it with better management techniques or some other improvements in efficiency.
  + Distribution companies would purchase electricity from net producers and sell it for a slightly higher price to net consumers. Some of the difference would pay for the distribution company’s management expenses and some would account for the energy loss between producers and consumers.
  + A customer who is a net purchaser of electricity might select a distribution company because it buys a certain percentage of renewable energy.
  + Distribution companies could buy and sell electricity from each other to balance their books
  + A node from the perspective of a distribution company might be a whole office building of even a small community of houses. An office building would be metered as a whole by the distribution company. The office building owners would be responsible for metering the electricity usage of their residents individually (if they choose to).
  + Who builds the distribution infrastructure?
    - A company that builds a section of electricity grid could charge distribution companies for electricity they buy or sell on that part of the grid
    - The infrastructure company would pay the metering companies for their metering information so they can agree with the distribution companies about how much money they are owed.
    - This is another reason why metering companies should be independent. The infrastructure companies would not want the metering to be managed by the distribution companies because then the distribution companies could pretend there is less electricity traffic across the grid than there actually is. The same would apply from the perspective of the distribution companies about the infrastructure companies managing the metering.
    - What would stop an infrastructure company from charging an unreasonable price for the use of their grid?
      * If they charged a very high price for electricity transmission across their grid, distribution companies would have to make the difference between their buy and sell price very large in order to pay the infrastructure companies this high premium.
      * The price of electricity would be very high for consumers, making demand low.
      * The distribution company would only buy from the cheapest suppliers of electricity, probably meaning the biggest companies because of scale-related efficiency.
      * So the poor consumers and small suppliers would lose out big time, but those that remain would pay a huge amount and the infrastructure company would earn monopoly profits.
      * The super high electricity prices would drive people away from the area, but the barrier to entry and exit would be so high that the infrastructure company could make a lot of profit for a long time.
      * So it looks like nothing is naturally stopping the distribution company from charging unreasonable (far above cost) prices
    - I wonder if it would be possible to design a system in which residents in some sense collectively own, and have to pay for, the electricity infrastructure in their area.
      * Purchasing a property could include purchasing a proportionate share in the collective ownership of the electricity infrastructure servicing that area
      * When a property developer purchases a large area of land, they would pay in full for the installation of electricity infrastructure in that area. When they sell the land in lots, the cost of the lots would incorporate the cost of installing the electricity infrastructure.
      * I think something along these lines probably already happens; the key issue with electricity infrastructure is all the infrastructure that has to be built between developed areas, such as between power plants and cities.
      * Zutha could pay for the construction costs of power lines between a power plant and a city, and then impose those costs somehow on all the property owners in the city that benefit from connectivity to the power plant. The community affected should be the group that decides, using Zutha, to build the power line; Zutha would simply enforce the even distribution of the cost of the public good.
      * The cost of the power line should not be incurred immediately and directly on the people currently living in the city. Instead, it should be represented as some kind of bond that is associated with the property in the city.
      * Zutha would basically announce that every property in the city now includes part ownership of the electricity infrastructure, as well as a loan owed to Zutha. Actually it need not be Zutha that the loan is owed to, but any financial institution that the community selects, through Zutha, to borrow from to build the power line.
      * Ultimately, the cost of the power line should be incurred by the users of the power it provides
      * The owners of the property in the city – the owners of the power line – should collectively determine what rate they want to charge for use of the electricity infrastructure, in order to pay off their collective loan.
      * If they don’t charge enough, the owners of properties will have to pay their part of the minimum monthly payment on the loan, which would be at least the interest on the loan.
      * If the community defaulted on the loan, perhaps because too many people left the area and those that remained couldn’t afford the minimum repayment, then the loaning institution would acquire ownership of the electricity infrastructure in the area.
      * The property owners of the city may choose to continue charging for use of their electricity system after they have paid off the loan. This revenue would then be distributed among property owners in the city. This might be done if the main users of electricity in the area were a small percentage of the property owners. In this case, there would be a net flow of money from high electricity users to the low electricity users.
  + Who maintains the electricity distribution infrastructure?
    - The community that owns the infrastructure could collectively hire a management company to manage and maintain the infrastructure
    - If the management company started charging too much or doing a poor job, the community would vote to change management companies
  + Who would arrange and pay for the construction of power lines between two cities?
    - Someone proposing the project would create a reputation domain on Zutha which includes the reputation domains of both cities’ property owners
    - People owning property in the two cities would vote on whether they want the project to go ahead – with the understanding that all property owners in both cities would be collective owners of the power line and would be collectively liable for it.
    - The same group would decide what rate to charge for electricity transmission across their power line
    - They would employ a management company to deal with all the details like hiring metering agencies and making deals with distribution companies
  + This paradigm of community ownership of infrastructure would be appropriate for lots of other infrastructure public goods like water supply, roads etc.
* Distributed government of a community via a Zutha reputation domain
  + An example would be the management of a city with respect to such things as building electricity infrastructure
  + How would such a system be made fair in the sense that every member of a community deserves an equal stake in decisions that impact them
  + When a new reputation domain representing citizens of a city is created, all citizen deserve an equal amount of influence (or reputation) in that domain
  + All citizens could be given an equal share of ownership in the item representing the city itself
  + Any Zutha user, but most likely users living in the city itself, can then invest in the city.
  + The worth of the city will be divided up evenly among citizens, as reputation.
  + Investments in specific parts of the city, such as parks or monuments, would aggregate up as reputation in the owners of the city. This is because the city item would be the owner of the park and monument (or at least one of the owners), so investments in those would add to the total worth of the city item.
  + Investments in many things will contribute reputation directly to some users as well as to the city as a whole. For example, the architect of a monument would probably be a partial owner of it. The architect may or may not be one of the residents of the city.
  + It may not be desirable for contributors to the city (such as the architect) to be given influence over decisions affecting its citizens.
  + The city item could be in the domain encompassing decisions about that city, but the monument may not be in this domain. Thus the monument contributes some reputation to this domain, but only through one of its owners: the city itself.
  + Alternatively, or in addition, gaining reputation in this domain may require more than just partial ownership of an item in the domain. It might also require the user to be a citizen of the city, for example.
  + Over time, certain citizens will show a greater interest in contributing to decisions concerning the city, and different citizens will contribute more than others to community projects. These citizens will accumulate greater reputation in the reputation domain associated with management of the city.
  + Initially, all citizens will have equal weight in making decisions. Some may be more active in using their influence than others, but essentially the community begins 100% democratic. This perfectly democratic citizenry will vote continuously on which of its members it values most, through investment in good deeds by fellow citizens.
  + Over time, the weighting of influence will become skewed towards a democratically selected set of representatives. There will be no explicit representatives, though; the degree of “representativeness” of citizens will vary on a continuum from citizen to citizen.
  + This process is effectively a continuous delegation of responsibility. People who aren’t that interested in thinking about the management decisions affecting their city may opt to just invest in the people they think are doing a good job at such management decisions.
  + Some citizens would probably be apathetic enough to never invest in anything and this wouldn’t be a problem. They would just be completely ceding any responsibility for decisions affecting their city.
  + The domain specific reputation granted to every citizen through their partial ownership of the city item would provide them with a small allowance of zuth. Citizens would be able to acquire more zuth via any means, including profit from investing in items completely unrelated to the city. This means that some additional influence will be given to active Zutha users over citizens who aren’t wealthy in zuth.
  + If a particular decision affecting the city should not be influenced too much by wealth potentially gained outside of the city’s reputation domain, the investment cap on that item would be set very low so that reputation is a much more significant factor in influencing that decision than wealth (in zuth).
  + It might be desirable for many non-decision items to be restricted to investment by users with reputation in a particular domain
    - For example, perhaps only citizens of a city should be able to invest in an action item by another citizen that affects the city
    - Then again, it will be important for the whole Zutha community to be able to praise and criticize individual communities.
    - The risk, though, is that a powerful user outside a community will try to infiltrate it by giving a huge amount of reputation to a selected citizen.
    - This behaviour could be pretty effectively policed both from within the community and with help from outside if the powerful user is powerful compared to the whole community.
    - In order to give reputation to a particular citizen, the powerful user has to invest in specific items or actions owned by that user. There’s only a certain point to which it would be possible to realistically exaggerate the value of a citizen’s contributions before it would become obviously absurd to everyone.
    - Where many people are going to agree that the value of an item is exaggerated, there is an incentive to invest negatively in that item as soon as possible to benefit from others doing the same later.
    - I think this effect will be sufficient to make it fair to allow anyone to invest in any items except decision items.
    - After all, if a community starts behaving barbarically as a group, it would be important for the wider community to be able to step in and criticize the barbaric decisions and praise the moderates in the group.
    - If enough people in a community feel that the power in their community is being warped undesirably by the outside Zutha community, they can choose to start their own independent Zutha network. If their behaviour deviates too far from what is acceptable to the wider Zutha community though, they might face military censorship from that wider community.
    - For example, I can imagine that a Muslim Zutha community might be pushed by the wider Zutha community to be nicer to women.
      * The wider community would influence the Muslim community by investing in actions that add to the reputation of moderates who support the community moving towards less discrimination towards women.
      * The extremists in the community are likely to be very upset about the power in the community moving to these moderates and may somewhat justifiably feel they are being oppressed or manipulated by the global Zutha community.
      * The extremists may rise up against the Zutha forces and volunteer institutions that protect human rights and try to push them out.
      * If they succeed, they may set up their own Zutha network in which the extremists acquire enough control to vote for such laws as bans against women paying for their own protection agencies, or more likely a ban against women working, which would achieve the same thing.
      * More likely again, the extremists will be uninterested in the concept of a Zutha community and will set up a theocracy.
      * In a scenario like this, I think the global Zutha community is likely to respond by not allowing the extremists to defeat the Zutha forces and philanthropic human rights protection agencies.
      * In this particular scenario I feel pretty comfortable about what is ethically appropriate, but considering the way Zutha is likely to respond to such an uprising forces me to consider whether it would do so in more ambiguous moral circumstances.
* The collective mind of the wider Zutha community could be very effective at curbing behaviour that is widely considered an abuse of human rights, but would it also sensor minority groups because of discriminatory attitudes?
  + There is also a philosophical question that is relevant here about whether it is ethical to forcibly prevent a society from perpetrating actions that are “widely considered an abuse of human rights” when the society itself, including the “victims” of the human rights abuses, may not consider them abuses of human rights.
  + In the case of Islam’s abuse of women’s rights, there is a way to check whether all members of their society truly believe in the doctrines that oppress women. Zutha can take over a Muslim community and enforce its global consensus on human rights, and allow anyone who wants to to leave and continue to live traditional Islam style.
  + I predict many of the extremists would leave, but many of the women would stay behind. Probably not all though. It would be a tricky business to create the atmosphere of security quickly enough that Muslims who are sceptical will feel secure enough in staying. A lot more than fear of harm would be involved, of course; there would definitely be a deeply ingrained cultural instinct that the Islamic way is the truth. That is why many Muslims would probably still leave.
  + I wonder whether the global Zutha community would feel justified in forcing Muslim people to remain under the Zutha umbrella for a minimum amount of time before being allowed to leave, just to ensure they have enough exposure to a new way of life to make an educated choice about what they really want. The moral landscape is already getting muddier.
  + I personally don’t think it would be ethical to prevent anyone from leaving the Zutha umbrella under any circumstances. That is a line that should not be crossed.
    - This would imply that a terrorist leader convicted of crimes against Zutha citizens would be allowed to leave the Zutha umbrella and go anywhere he wished.
    - Perhaps the policy should be that only people who have not been locked up legitimately by a Zutha protection agency are able to leave whenever they want
    - If a husband of a Muslim women severely beat her for some kind of religious offence and a philanthropic protection agency locked him up for human rights abuses, wouldn’t that imply Zutha restricting the right of that family to leave Zutha’s umbrella?
    - No. Only the right of the abusive husband to leave. Still, it might amount to the same thing. The abused wife may be hysterically upset about her husband being locked up and demand his release so they can all emigrate to Iran. For some reason I think this scenario will be relatively uncommon.
    - Perhaps the Zutha community will support a law which gives the victim of a crime absolute control over the punishment, or absence of punishment. Actually this is already the case for crimes policed by standard commercial protection agencies – the rule just has to be extended to philanthropic protection agencies. Should (or will) Zutha mandate that philanthropic protection agencies must operate this way?
  + Of the issues I have considered so far, here, I am not really concerned much by what the Zutha community decides, because they are legitimate moral dilemmas which the Zutha community has as much or more merit of choosing wisely about than I or any other individual does.
  + What I am most concerned about is whether the system I am designing would be prone to voting for discrimination which would ultimately lead to harm to society. In theory, the Zutha community would make all its decisions based on what would ultimately be most beneficial for all its members. Yet democracies demonstrably don’t make choices that are in their own best interests.
  + Actually I don’t think this is the real worry. Zutha will be far better at doing what is in its long term best interests than a democracy. The worry is that what is in the long term best interests of the (weighted) average Zutha citizen may involve harm to people outside – or minorities inside – the Zutha “club”.
  + Let’s consider a simple example: homosexuality.
    - There will be a lot of people in a Zutha community who feel repulsed (or other negative emotions) by homosexuality
    - They will likely act on this discriminatory attitude by investing negatively in gay people, actions by gay people, and actions that do good things for gay people
    - Consider a popular figure in a relatively gay-friendly community who is pushing for more rights to gays in his area
    - The local community might have a distribution of reputation that would be conducive to allowing the changes in gay rights
    - A sect of powerful users on the other side of the world could hate homosexuality with a passion and decide to try to warp the distribution of reputation in the relatively gay-friendly community such that the proposed changes in policy won’t pass. They would invest negatively in actions and people they perceive to be “gay” and positively in actions and people that are homophobic.
    - There could be a large and powerful category of people across the global Zutha community with an agenda to oppress homosexuality (the Muslim community?). In aggregate, they would be far more powerful than the small community trying to privately determine its own policies
    - There would clearly be a large category of users across the Zutha community who would oppose the homophobic group. Whether they are likely to be more powerful than the homophobic group or not, it would be exceedingly undesirable for a small community’s private decisions to be put on a global stage and made into an epic scale power struggle between global factions.
    - If there was a strong protocol that it is inappropriate to interfere with the private affairs of a community then there might be a lot of users willing to neutralize the effects of the homophobic group and criticize them for breaching Zutha protocol and interfering with the decisions of communities they don’t belong to. This is still a messy scenario though.
  + There is no reason the amount of discrimination in (or from) a Zutha community will be any more than the amount in the average democracy, so why am I concerned about this?
    - Because I want the Zutha umbrella to be able to extend over a much more diverse range of cultures than exists within one democracy
    - As discussed in Guns, Wars and Votes, democracy would not work for a population with too much diversity of culture.
    - It will be very important to fragment influence so that individual communities can largely manage themselves for issues that only concern themselves, yet still benefit from the protection and disincentives for conflict provided by the Zutha umbrella.
  + I think there needs to be some sort of barrier preventing any user at any time influencing the power structure in a local community they don’t belong to
    - What would be the implications if only users with reputation in a domain could invest in items of that domain?
    - What about the case of a pretty fountain that falls under the domain of the city; shouldn’t visitors to the city be able to invest in the fountain if they like it?
    - Perhaps some items could be opened to public investment, including the city itself and the things it owns. Actions by citizens directly related to management of the city might not be open to investment by outsiders.
    - So an outsider would be allowed to invest in a pretty fountain and the reputation would just be divided evenly among the citizens of the city
    - An outsider could invest in a kind deed he observes of a citizen of the city, such as a boy picking up something dropped by an old lady. This action would not fall under the domain of city management, but would only contribute to the reputation of the boy in the universal domain.
    - An outsider would not be allowed to invest in a comment by a citizen about the merits of a proposed power plant construction project. This comment would fall under the domain of city management and could only be invested in by citizens of the city in proportion to their City Management reputation.
    - An outsider would be allowed to write a reply to the comment about the merits of the power plant construction project. Since this reply would not have been written by a member of the City Management domain, it would not fall under the City Management domain. Thus any Zutha user would be allowed to invest in this reply to indicate their opinion about the construction project, but their input would not have any effect over the distribution of reputation in the City Management domain.
    - There may be cases where a small number of members of a restricted domain gain disproportionate control over that domain and lead it in a direction that is unacceptable to the wider Zutha community. In this situation, it might be justifiable to allow the Zutha community to vote to open the restricted domain temporarily to the public so that the distribution of reputation can be re-allocated according to the aggregate opinion of the global Zutha community (or perhaps a restricted subset of it). The threshold on the decision item to open the restricted domain to public interference in this way, should be very high.
* Threshold (item worth required to pass a decision) = percentage of total domain rep
  + The total amount of reputation in a domain represents the maximum possible worth of a decision item in that domain
  + If the decision item restricts the maximum investment size to a fraction of one’s reputation, the maximum worth will be lower by that same fraction, and the threshold for passing the decision should be adjusted accordingly.
  + So users select an initial worth that a decision item of a particular type must reach in order to pass. This is the “threshold” of the decision item
  + They also select the reputation domain which restricts investment in the item as well as the fraction of one’s reputation in that domain that they can maximally invest in this decision item.
  + The threshold is indexed against the total reputation in the domain of the decision item. This means that the absolute value of the threshold varies continuously – probably upwards – as the total amount of reputation in the domain increases.
  + If the fraction of reputation that limits the maximum investment size changes, the threshold must be adjusted accordingly.
* Fund research into adaptive frequency finding for devices.
  + Once technology in this area is sufficiently advanced, I think the most optimal policy for maximising public utility with regard to wireless spectrum would be to have almost all spectrum open for anyone to use, with perhaps some segments reserved (perhaps merely by status quo) for devices to find each other.
  + All devices would always listen for incoming requests on the reserved frequencies
  + Once two devices find each other, they agree on a frequency to communicate on and both switch to that frequency.
  + If a frequency begins to get crowded, devices communicating on that frequency will agree on a new random frequency to switch to which is less crowded. They may use their current frequency to do this – the one that is getting crowded – provided it is not yet too crowded to prevent reliable communication, or, if necessary, they can switch back to one of the reserved discovery frequencies to agree on a new communication frequency.
* Messaging system: send messages to individual users or categories of users. Bar at top of web page scrolls through highest ranked messages in inbox.
* Private garbage collection.
  + Consumers pay per weight for waste collected.
  + They choose their garbage collection company.
  + Companies reduce costs by finding optimal mix of landfill and recycling.
  + Make landfill more expensive than other uses of land to account for cost on environment. This might arise automatically by people being less willing to sell land for landfill due to reputational costs.
  + How to prevent people dumping garbage in vacant lots or in the forest or something to avoid costs?
    - Doing this would be a crime against the whole community
    - Policing crimes like this would be the sort of thing Zutha should fund, along with philanthropic organisations
    - What if people pay their protection agencies to protect them against being prosecuted for dumping garbage?
    - Firstly, not many people are likely to support a protection agency with such a policy, because they will recognise that it is allowing others that pay for that service to do harm to the public, which includes them.
    - Many people are likely to want to be a part of a protection agency that considers it a crime against its customers for someone to dump garbage and harm the environment. It would be easy to get protection agencies to police dumping garbage on property owned by their customers; it may be more difficult to get them to police dumping garbage on unclaimed land.
    - Zutha could take implicit ownership, or at least responsibility, for any land not explicitly claimed by people. Zutha would pay protection agencies or use forces it owns directly to police against people dumping garbage on such land. If a protection agency defended one if its customers for dumping garbage on unclaimed land, Zutha may have to, at worst, fight it militarily to impose Zutha’s rule that dumping garbage on unclaimed land is unacceptable. More likely the probably small protection agency will chicken out at the threat and under pressure from public animosity towards it.
* How could costs be imposed for actions with negative externalities such as emitting pollution?
  + The emission of pollution could be taxed to incur a cost on the polluter indicative of the cost the pollution has on society
  + Who should receive the tax?
  + The only appropriate candidate I can think of is Zutha itself. Zutha represents the community and so it could justifiably receive payments to offset costs incurred on the community, which it would then spend at the direction of the community.
  + The problem with this solution is that Zutha represents the global community, whereas a lot of pollution really only affects a local community.
  + There is a lot of pollution that potentially affects the whole world to some degree. Greenhouse gases or any air pollutants that may potentially circle the globe would come under this category. It seems sensible for Zutha to take responsibility for taxing these forms of pollution and taking the proceeds of the tax.
  + Some pollution only (or mostly) has a local effect, such as cigarette smoke, littering and larger scale dumping of waste, poisoning a local habitat with agricultural chemicals etc. These sorts of issues could be dealt with on a case by case basis by the local community. The local community might for example impose a fine on dumping waste, the proceeds of which would be divided up among members of that community.
  + Many forms of pollution would be very difficult to police because it is very difficult to measure their environmental effect. There’s nothing I can do about this. Technology will get better and communities (including the global Zutha community) will do the best they can to design policies and taxes that create the right incentives with the measurement tools available.
* Private companies would not have an economic incentive to service rural areas. Could and should Zutha do something to correct this “market failure”.
  + The inferiority of access to technology and services in rural areas is an intrinsic reality. Governments often ameliorate the effects somewhat by expending resources to provide them with more services than they would otherwise get.
  + Government provided postal service and education, and regulated telecommunications are examples of government interventions that somewhat narrow the gap between the wealth accessible to rural inhabitants versus city dwellers.
  + Should Zutha expend resources in a similar way to somewhat offset the disadvantages of living in rural communities?
  + I think it would be a good thing to do. I don’t think city dwellers have any moral obligation to give money to rural dwellers, but I think many people who live in cities would feel generous enough to do so anyway.
  + Some people may feel strongly enough about helping people in rural areas to donate to the cause directly. Others may feel the whole community should subsidise living in rural areas and can use their political influence with Zutha (their reputation) to vote for more of Zutha’s resources to be invested in that project. The latter would be a similar mechanism as the one that allows both rural and city dwellers to vote to increase government expenditure to support rural areas.
  + Actually there is a major difference between the two: governments forcibly take resources from most members of society and give some of it to rural communities, whereas Zutha earns money by providing a service. The influence a person has in determining where those earned resources are invested is proportional to the amount that person contributes to providing that service, whereas in a democracy the influence is skewed towards those who do not contribute the resources.
  + I wonder whether this will cause a Zutha community to contribute fewer resources to the disadvantaged than a democracy does.
  + I think that a Zutha community will do more good for the disadvantaged than the average democracy does both because of the focus on one’s role in the community that the a reputation-based economy would create, and the enormously greater efficiency with which a Zutha community would use resources. I don’t know whether the absolute proportion of resources that flow from wealth producers to the disadvantaged would be higher or lower. I think it is possible that the proportion would be higher, but even if it were significantly lower, the disadvantaged would still be better off in a Zutha community than in a democracy.
  + I’ve specifically been talking about the “disadvantaged” as opposed to the poor. I think the amount of resources that would flow from the rich to the poor in general would be far less than in the average democracy. This is not because people in a Zutha community would care less about the plight of the poor, or even that the system would lead to less expenditure on the poor than the community would like. The reason less money will be *spent* on the poor is because it will take a smaller investment of resources in a poor person or community before they stop being considered poor and start producing their own wealth.
  + In the average democracy, expenditure of resources on the poor takes the form of throwing money at poor people (a lot of which goes down the drain) because society feels guilty about seeing them miserable. It is a classic case of giving a man a (lot of) fish instead of teaching him how to fish. Obviously there are plenty of attempts to “teach poor people how to fish” as it were, through schooling and other means, but the success rate is demonstrably low.
  + Now let’s consider how a Zutha community might go about reducing poverty
    - Entrepreneurial teachers will go into poor villages and offer to take kids on as apprentices. They will charge no upfront fee, but instead negotiate a percentage of the child’s earnings for a particular number of years into the future, which will be owed to the teacher. The child faces no liability and has an incentive to listen intensely to what the teacher has to say because they will know that the teacher’s interests match their own: to make them capable of generating as much wealth as possible.
    - Some teachers may not make any kind of contract with students, but may suggest that the student voluntarily offer a proportion of their income, which they may vary over time, to the teacher, in order to create effective incentives for both parties.
    - With no minimum wage, companies will have an incentive to find people in a desperate state of poverty, give them an optimal amount of training (optimal being greatest earning potential increase to cost ratio), and pay them their worth. If another company thinks they can optimally train them to a higher earning capacity or use their existing skills more efficiently, they will outbid the first company with a higher offered wage.
    - Philanthropic organisations may offer to employ workers at a loss purely with the goal of training them to lift themselves out of poverty. Since philanthropic organisations will receive funding based on their actual success at achieving their goal of reducing poverty, and will be in competition with other philanthropic organisations to most optimally achieve this goal, they will have an incentive to innovate to find the best way to do this. One of the things the survivors of such a system are likely to discover early is that giving money to poor people does not alleviate poverty.
* There should be multiple available mechanisms for allocating ownership among multiple owners
  + Ownership could be equally spread across a set of users, where the set of users can be specified explicitly or by some criteria.
    - E.g. the ownership of a city item is divided evenly among citizens
  + Relative ownership is calculated based on some function, such as amount of property owned or reputation in some domain
    - E.g. the ownership of a city is allocated evenly by square meter of land such that owners of more land get proportionally more ownership of the city
  + Relative ownership is specified by the relative worths of candidate items created specifically for the purpose of assigning relative ownership
  + Relative ownership is specified as explicit percentages by the item creator. Other users (with sufficient reputation) can modify these values later. If a dispute arises, it can be resolved by a period of time of investment in candidate items to select a new distribution of ownership, after which a new fixed set of percentages are assigned. Alternatively, the method of determining relative ownership could be permanently switched to the investment in candidates option.
  + Distribution of money among beneficiaries should also have these options
* Users could create containers in which to pour anything from twitter-like short messages through blog-length messages, to entire essays
  + You could “follow” another user’s container and ask to be notified when a new item appears, or when an item exceeds a threshold worth, or some combination of the two

# Sat 29-Jan-11

* Someone designs a city. Many companies buy blocks of land, ownership of which includes partial ownership (and associated debt) of the common infrastructure (like electricity and road systems). The companies finance the common infrastructure by their purchase of the combination of land and associated stake in infrastructure. They can then sell both onto consumers. The additional cost of the land will be offset by income from the associated capital: a stake in the common infrastructure.
* What stops an electricity distribution company from selling electricity it has not bought?
  + If a distribution company’s suppliers and consumers are public knowledge, than anyone can police the distribution company’s books to ensure they balance.
  + Someone would also need to ensure that distribution companies purchase enough electricity to account for what they sell given that some electrical power will be lost during transmission.
  + I think all these problems are already faced by existing electricity distribution systems, so I’m sure it’s possible. There are companies in Australia that are only resellers of energy. Somehow, someone needs to check that they don’t sell energy to consumers that they have not purchased.
* What stops companies from hiring protection agencies that discriminate against consumers suing them effectively? Would the protection agencies of individuals be powerful enough to hold their own?
  + This is a specific case of a more general concern that factions within society will come to unresolvable disagreements which lead to conflict between groups of protection agencies.
  + In the case of companies versus consumers, the big question is whether it is worth more to consumers as a group to be able to sue effectively than it is for companies as a group to be able to avoid being sued.
  + If it is worth more to consumers to be able to sue, they will vote to have their protection agencies use force to extract punishment from offending companies. The cost to the companies of fighting back will be huge as it will probably start a large scale conflict. The cost of such a conflict is very likely going to be enormously larger than the expected cost associated with being suable, since most companies will only have a small chance of being sued. Thus most companies will probably switch to protection agencies that are not facing conflict with the united forces of all consumers.
  + Another factor in consumers’ favour is that they can discriminate against companies who hire protection agencies that don’t accept suing claims. Companies will have an incentive to switch to protection agencies that consumers consider to be fairer, because it will give them a reputational edge against their competitors that stick with protection agencies that protect them aggressively against being sued.
  + I haven’t yet taken the role of independent courts into this equation
  + The way companies’ protection agencies would resist suing would be to only accept courts/arbitrators that don’t pass suing claims against companies. Consumers’ protection agencies would not accept any of these courts so the deadlock would take the form of no court being acceptable to both parties.
  + There might be a role for Zutha, here, to force parties to accept a neutral court if they can’t agree on a court on their own. This might take the form of Zutha enforcing a law that requires a set of courts (selected by vote of the whole community) to be on the accepted list of every protection agency. Protection agencies can place these compulsory courts as far down their priority list as they like.
  + The court for a case could be automatically selected by calculating which one maximises the combined acceptability score across all protection agencies involved.
    - Protection agencies would assign a non-zero percentage acceptability score to all the compulsory courts as well as any others that are acceptable to them.
    - The court which maximises the product of acceptability scores across all protection agencies involved would be selected
  + If a protection agency refused to go to the court selected, or refused to accept its decision, other protection agencies, with the support of Zutha, would have an incentive to force the rebellious protection agency to submit to the court’s decision. If they did not intervene, and a protection agency was allowed to get away with disobeying the accepted practice of law, this would undermine the rule of law for everyone, which would mean everyone would have an incentive to pay to prevent this happening.
  + A protection agency that merely balked at a court’s decision would lose a lot of reputation, and any people or companies that remained as its clients would receive some shock to their reputation as well. The reputational attack could be even more effective than the threat of military confrontation for some protection agencies.
* Should IP addresses be recorded?
  + It would be useful to be able to reverse all the actions of a vandal who uses many user accounts but acts from the same physical address
  + But in order to have this information available, all Zutha hosts would have to know every request’s IP address so they can reverse all actions by that IP address. This would eliminate the capacity for privacy and anonymous users, yet these features will be very important to allow people unhindered free speech.
  + A vandal would not be able to do much except create massive quantities of junk items, because he would not be able to get hold of large numbers of high reputation users.
  + If a vandal had the will and resources to do Zutha a lot of damage, he could relatively easily acquire many IP addresses. Still, the number of IP addresses available to someone would be substantially less than the number of user accounts he could create automatically, let alone the number of items he could create automatically. So it would still be useful to be able to delete items based on creation IP address, even if many IP addresses had to be identified for a single bout of vandalism.
  + It would also be useful to be able to identify MAC addresses and IP addresses used by terrorists or other criminals, but only once the Zutha community votes to release them.
  + This could be done in the same way that a user’s location is kept private until the information is asked to be released by the Zutha community.
    - The host that a client connects directly to would record details about the client in its private logs, including the client’s IP address and MAC address.
    - When the Zutha community wants to release such private information about a user, a message is sent to all hosts to release that information.
    - The hosts that have private information about that user announce themselves and send out the IP address data to the whole network.
    - In this way, the Zutha community can get information about every connection a user made with the Zutha network, including which host he connected to (indicating rough location), his IP address and MAC address, and any other identifying information which might be recordable now or in the future.
    - Unless the Zutha community requests this information, it is effectively secret to any individual attacker, because the information could be stored on any of a very large number of hosts. If an attacker owns some of the hosts, he will have access to the connections data for those hosts, but he is hardly likely to come across the data of a specific targeted user in that data.
    - Hosts do not know anything about the private connections data of other hosts, except for having a probabilistic knowledge of which of its neighbours are closest in the network to a client. In order for an attacker to locate the connection data for a particular target user, the attacker would have to hack into a long chain of independent hosts to trace the path taken by a message from the target user. The last host in the chain would have some connection data about the user.
    - It will be extremely difficult to hack into many independent hosts and will likely take a very long time even for a powerful entity. By the time this powerful entity has located the location from which its target sent a recent message, the target will likely have managed to be long gone - if he is aware he is being pursued.
* Terrorists could use Zutha for money laundering by creating many fake “charity” items owned by many different users and using many different users to pay real money to those charities.
  + They need not even create convincing charities – merely a huge number of barely valuable items to which tiny amounts are donated to, each, with the total across all these items adding up to a large amount.
  + There would be no common theme across any of the senders or receivers of money which could be used to block these transactions
  + It would only be possible to do this on large scale with zero or very near zero reputation users
  + If the terrorists used some users with reputation to give some reputation to each of their mules, these high reputation users would be the common thread which could be used to block the transactions.
  + Thus the way to block this money laundering opportunity is to require a minimum amount of reputation to send and/or receive real money
  + I think both sending and receiving money should be blocked to very low reputation users. Then again, this would mean users who aren’t active on Zutha would not be able to receive donations for their work.
  + Perhaps just sending money should be banned to low reputation users. But this would cut out a lot of potential revenue for content creators and Zutha because many people would like the Zutha platform for donating and investing, but may not be able or interested in earning a high reputation on the site.
  + I was wrong about content creators not being able to receive money without being active users of Zutha. They have reputation by the very fact that they have created something worth donating to. This looks like the best solution then. You cannot donate to an item that has less than a minimum threshold of worth in zuth.
  + Now if terrorists want to send money across Zutha they have to have a high reputation user with a lot of cash to invest in all the items that receive the money. They may have quite a few of these high reputation users and may manage to keep them all independent, but their ability to create independent high reputation users will be severely limited because they will actually have to contribute significant value to accrue reputation for each one.
  + If a particular high reputation user is found to be investing in a large number of items that subsequently receive suspicious donations (in the sense that they clearly don’t deserve donations), then suspicion will fall on that user and all other items invested in by that user that are receiving donations.
  + All this might not be necessary because users’ bank details are not secret. If a user wants to receive money, he has to specify the account for Zutha to send it to, and this is public knowledge. It is unlikely that a terrorist group will be able to get hold of thousands (or even dozens) of completely independent bank accounts so the pattern that will expose them will be many suspicious donations all going to the same account. That account can then be investigated and its owner identified – if the bank is cooperative. The terrorists still have all the existing avenues to conceal their connection to this “front” account, but those avenues already exist and governments are constantly working to close them. Zutha is not providing any additional security for money launderers, except that, unless the Zutha community agrees to reveal this information, it protects information about users’ location.
  + This means that only users without access to Zutha’s real money services can be truly anonymous – even from governments.
    - This is not a problem. The purpose of anonymity is for sharing sensitive information. There is no need for a whistle blower to be able to send or receive money.
* Many actions will be doable instantly by a single user with a certain minimum reputation. What will this minimum reputation value be indexed against so that it scales with the growth of Zutha?
  + Minimum reputation values could scale with the reputation of the highest reputation user in that domain
  + This would be an elegent way of preventing some actions from becoming unavailable to anyone because the highest reputation user in the domain falls below the minimum reputation required for that action.
  + If a single user was a long way ahead as the highest reputation user in a domain, he could set the minimum reputation requirement on some actions so that only he could do them. If this is a problem it would already have been a problem even if the minimum reputation value didn’t scale with that highest reputation user’s reputation, because he would always have been able to continue lifting the minimum reputation value of an action as his reputation increased.
  + I don’t think it is really a problem if the highest reputation user can make the minimum reputation requirement for an action too high for any other user to access. If he abuses this privilege, the other users in the domain simply have to bring down his reputation until he is no longer the highest reputation user.
  + There should also be a mechanism to open a dispute about an action, or request an action be performed that you don’t have permission to do. In both cases, a decision item would be created which would allow all users with reputation in the relevant domain to participate in.
  + The only real danger is if the highest reputation user in a restricted domain is more powerful than all other users in that domain combined. This would be a situation where the wider Zutha community would have to step in to rebalance things. For domains that are not restricted, the wider Zutha community would be constantly involved in influencing the reputation of users in the domain.
* Wealthy entities will be able to buy votes on decisions from high reputation uses
  + Does this pose a risk that a scenario will arise like the one seen in modern democracies with rent seeking and trading of political favours?
  + There are several major differences about Zutha’s political system which will make it very resistant to rent seeking and effective trading of political favours
  + Firstly, control is separated out over a great deal more people in Zutha, many of whom will not be accessible for secret communication because their only known identity is through the public Zutha interface. This will make it exceedingly difficult for rent seekers to find and convince a significant proportion of powerful users to vote for policies that benefit them.
  + Secondly, users have to justify their votes and if they don’t do so acceptably (especially if they are powerful users) they are liable to be punished for the bad decision by the rest of the community. In democratic politics (even in its idealistic form), politicians only receive feedback on the whole bundle of all their deeds over a period of time. Zutha users receive feedback very directly for every decision, so they have a strong incentive to make every one of them a good one.
  + Thirdly, influence in decision domains is concentrated among users who actually know and care about that domain, because that is how they acquired their high reputation in that domain. In order to pass a decision in a domain, these interested parties will have to be convinced so it will be very difficult to pass a decision that only benefits a minority of those interested parties. In a democracy however, influence is spread across the whole, mostly disinterested, population evenly, so it is relatively easy to pass laws that benefit only a minority of voters because that minority will be the interested proportion of the voters and will be the only ones watching.
* Some organisations may want to use Zutha’s semantic database and software tools with a completely isolated community.
  + If they only want to be isolated in terms of reputation, but are happy to allow zuth to flow between their community and the wider Zutha community, then they can just use a restricted domain
  + If they want to be completely isolated from the wider Zutha community, including from Zutha’s economy of zuth, then they would have to set up an independent network.
  + An independent network would still be able to tap into the wider Zutha network to retrieve data, but it would not be able to modify Zutha’s data
  + The independent network could synchronize its private semantic database with the Zutha database, either just one way – from Zutha’s database to their own – or two ways if they have access to users on Zutha with sufficient reputation for all their changes.
  + Changes made by users who only have reputation in the private network would only be able to be mirrored in the public Zutha network if the user used to synchronize has sufficient reputation in the Zutha network.
  + In the case of a restricted domain, the zuth that flows within the restricted domain has been earned legitimately in the wider Zutha community, and so reputation created with it contributes to global Zutha reputation.
  + In the case of an independent network, where the economy of zuth is controlled privately, reputation created in that network would not be able to contribute to Zutha reputation because the controller of that network would then have power to create arbitrary amounts of zuth and reputation.
* Who is likely to want to destroy Zutha and when will it be most vulnerable?
  + If Zutha went into an African country and claimed some of its land, the government would be quite likely to fight to maintain its sovereignty.
  + I think Zutha would be able to accrue enough resources even before it creates its first local community to take on a small African dictatorship (or fake democracy).
  + I don’t think there would be that much reason to worry about multiple African dictators teaming up to drive out Zutha because it is unlikely that neighbours of a country Zutha has invaded would recognise Zutha as a threat, especially while the first Zutha community is still small, which will be when it is most vulnerable.
  + There are a lot of powerful companies that may feel threatened by Zutha because they would recognise Zutha’s potential to undermine their government granted privileges. Daniel Suarez paints a picture of all the shadowy industrial powers teaming up to take over the world when they recognise that the Daemon threatens their position of power in the world unless they can take control of it. I think Daniel Suarez indulged himself too much in popular conspiracy theories. There might be some invisible ties between powerful companies, and there are definitely powerful (partially) invisible ties between big business and government, but I don’t think there is any organised “power behind the scenes” that would be capable of marshelling as a unified force against a threat to their collective “financial empire”. Competition between powerful companies is real, and I think there would be as many companies that would seek to take advantage of Zutha and/or adapt to it in order to get an edge over their competition, as their would be companies that would seek to destroy Zutha because they see it as a threat. The latter are likely to attack Zutha primarily though government regulation rather than military force, though, I think, and this leads to possibly the most important group of entities in this equation: democratic governments.
  + There are many aspects of the Zutha phenomenon that are liable to evoke strong negative reactions in people. Here are some examples, described in the most unfavourable light I can think of.
    - Zutha profits from people gambling
    - Zutha conquers democratic states like a new age colonial power
    - Zutha takes advantage of the poor by making them work for less than subsistance wages
    - Zutha takes advantage of children by placing them in lifelong bondage that requires them to pay a percentage of their income for the rest of their life as some kind of commission to their bond holder.
    - Zutha gives absolute power to an intellectual elite who rule over the poor and unfortunate and extract wealth from them
  + I’m sure a skilled propaganda expert could come up with even more effective distorted portrayals of the nature of a Zutha community. There is a risk that some such negative impressions may arise naturally through people’s existing preconceptions.
  + There are special interests that would have a strong incentive to flare up negative sentiment against Zutha and try to get their government to act to stifle it. Here are some examples of actions governments could potentially be induced to take against Zutha
    - Zutha could be declared a terrorist organisation because it invades democratic countries and assists terrorists in communicating secretly
    - Trade with Zutha communities could be banned or severely restricted, perhaps with the justification that they are terrorist organisations, or simply that they aren’t legitimate states
    - Use of the Zutha site could be banned with the justification that it is an online gambling site, or a terrorist site
    - One or more governments could declare war on Zutha for invading a democratic government
  + Theses are worst case scenarios. I am optimistic that Zutha will be able to become popular enough before it starts doing potentially controversial things like allowing “gambling” with real money or invading African countries, that when it does start doing such things, a lot of people will understand its nature and purpose well enough to defend it. If Zutha were able to become as popular as Facebook, say, then practically every second person would be aware that Zutha is not some evil conquering being, but the collective opinions and intentions of many good people, including themselves. If a government banned trade with a Zutha community then, then they would be directly opposing the will and hindering the profits of a large number of its citizens.
  + Is there any chance that any of the western countries would try to drive Zutha out militarily if it took over a piece of land in Africa and declared its own sovereignty?
    - Would they be able to justify this if Zutha took zero lives even after many invasion attempts by the local despot, and was already substantially increasing the prosperity of local people living under its umbrella?
    - Even though land regularly changes hands through bloody conflict and the powers that be in the world turn a blind eye, Zutha might be treated differently because it might be perceived as “the west” invading Africa and the western powers might feel they have a responsibility to police their own even if they don’t have a responsibility to prevent Africans from massacring each other.
    - The big question is whether any democracies’ constituents would accept military assault against a peaceful community that is obviously doing a lot of good to reduce poverty in Africa. It seems unlikely to me that any would. My guess is that the soldiers assigned to such a task would be so repulsed by it that it wouldn’t take much to convince them to lay down arms and join the Zutha community.
  + Would a powerful non-democratic country feel any need to destroy Zutha? China maybe?
    - I can’t see any reason why.
    - China hasn’t gotten involved much in the world so far and I can’t see why they’d feel a need to put down a tiny, but perhaps quite interesting, takeover of a small piece of land in an obscure African country.
  + Zutha will grow quickly from its most vulnerable state as a tiny community in outback Africa. Is there a point during its growth that is likely to be threatening enough to powerful entities to make them attack, before Zutha becomes too powerful to be vulnerable to any existing powers?
    - Zutha will quickly become too strong to be threatened by African despots because it will be so astronomically more productive than the typical African country that it will quickly be able to design and construct enough advanced non-lethal defensive weaponry to outmanoeuvre any sized army of teenagers with AK47s.
    - I don’t think Zutha is ever likely to be that vulnerable to democracies either, because Zutha will not be a tangible military power.
    - Let’s imagine that the United Sates decides to “liberate” Africa from Zutha’s inexorable advance. So they bring in the tanks and fighter jets and… what? Start killing African civilians living in Zutha communities?
      * If American soldiers walked into a Zutha community they would be unharmed unless they harmed someone. There would be soldiers from many independent protection agencies scattered around, armed with non-lethal weapons. Would the US soliders attack them? For what purpose?
      * There would be no enemy to attack since the community would have no central command.
      * There would in fact be no one oppressing anyone
      * Unless the Americans were willing to exterminate every human being in the Zutha community, the soldiers would have nothing to do but go back home… or change their address to Zuthaville, Africa.
      * I don’t think the Americans would come in the first place… at least not armed anyway.
    - I think Zutha could encompass the whole of Africa without any powerful country intervening.
    - The question is what would it do next?
    - I predict that long before Africa is a thriving array of Zutha communities, people will try to start their own Zutha communities in their own countries. The problem is it would be a far messier scenario to defend a block of land against the US military than against a backward African despot.
    - Democratic governments can be quite aggressive about enforcing their right to tax their citizens. If people defend themselves with force against the police trying to jail them for tax evasion, they will face real force.
    - But how would the dynamic be changed if the public was aware that this group of tax evaders was trying to start a Zutha community like the ones that were right now exploding African countries out of poverty at a miraculous rate? And how would the public react if their government responded to non-fatal defensive weapons with lethal firepower?
    - Then again, if Zutha communities have developed potent non-lethal weapons, it is likely that many western governments will have figured out how to make them as well – the design will be open source after all. But maybe I’m being too optimistic about the capacity of bureaucracy to adopt new technology in a timely manner. The US military might manage to keep up. If so, there would be a battle between innovative robotic military devices and non-lethal weapons. The US military would be extremely reluctant to lose, but if they lost their temper and got careless enough to kill unarmed people in a massively public spectacle they might feel even more embarrassed. Things could get very tense.

# Sun 30-Jan-11

* Require robotic devices to identify their owners with a digitally signed signal.
  + Unidentified devices will be captured or destroyed.
  + This is to prevent untraceable remote assassinations and other crimes using advanced robotic devices
  + Would it be possible for a device to have a private key for signing that couldn't be extracted from the device?
  + I don’t think it’s that difficult to make devices that store information that is extremely difficult to extract. DRM technology relies on being able to conceal decryption algorithms enough that they cannot be extracted from DRM protected devices.
  + Every device beyond a certain level of complexity should be required to be registered as owned by a publically known identity.
  + In fact, every device should have its own unique identifier which is registered in a public database.
  + The owner should be able to remotely change the key used by the device to identify itself. At the same time, they would add the new key to the public database under their name, and remove the old key from the registry. This would increase the security of the identity system because someone who managed to steal a device’s identity would only be able to use it until the owner blacklisted the old key and updated the authentic device with a new one. The owner might do this regularly, or just when they recognised that a counterfeit device was masquerading under their name.
  + Devices that do not identify themselves with a valid registered key in the public database would be impounded and sold by Zutha or the protection agency that confiscates them.
  + An owner would be able to regularly query a device they own, to ensure it is still under their control. If they discovered tampering, they could blacklist the device and it would be confiscated by device police or anyone wanting to make a buck by selling unclaimed or inauthentic devices. There would of course be an investigation into the device’s history as recorded by security cameras and people. If the tamperer was identified he would be punished accordingly. If the authentic owner of the device could be identified the device would be returned to them. If not, the entity that confiscates the device would get to sell it as a reward for helping to police the streets against the threat of untraceable crimes by anonymous devices.
  + The rule that Zutha would impose would simply be that all devices must be cryptographically identifiable and traceable to an owner who has a public identity. The technology used to achieve this goal would be allowed to vary and improve over time. As new identification standards and paradigms emerged, they would be added to the known pool of acceptable identification mechanisms and protection agencies and Zutha device policing agencies would be expected to keep up. There would need to be a constant dialogue between identity technology inventors, adopters and regulators to ensure they all agree on a common set of acceptable technologies.
  + Only devices that have the capacity to harm people or property should be required to regularly respond to requests to identify themselves.
  + A person should not be required to have his phone regularly respond to identification requests as this would effectively broadcast every person’s location at all times for the whole world to see. Unfortunately, this is not far from the state of affairs right now and the trend is only likely to continue towards less and less privacy in the real world. Is there any point resisting the trend or should society just get used to it?
  + I need to read that book about Privacy to get some more perspectives on what the real risks are with respect to loss of privacy.
  + I can see that loss of privacy would be (and is) very dangerous in a world where there are entities with a lot of power (like governments) who can oppress individuals who displease them more effectively the more information they have about that individual.
  + In a Zutha society, though, even if you say or do something to displease a powerful entity, that powerful entity will not have the power to harm you without severe consequences despite the fact that it knows where you are practically every moment of your life. There is no government in a Zutha society that can act without full accountability to the law. Instead there is a lot of redundancy with many policing agencies watching everyone and each other.
  + Now and historically, humans have not relied on privacy to keep themselves safe. It’s always been the case that, if someone wants to hurt you, it is rarely their lack of knowledge about your whereabouts that stops them, but instead the consequences of hurting you. Those consequences might be the wrath of your family or tribe, punishment by the rule of law, or nothing (as in the case of a dictator or other unaccountable government), in which case you aren’t protected very well even if you can hide for a while.
  + I guess a lot of people do like their secrets.
    - Husbands might want to cheat on their wives without getting caught, for example, and this would be made difficult if the path of their cell phone was public real-time information.
    - Competing CEOs might want to meet in secret to arrange a collusive agreement and this would be made more difficult if everyone knew they were meeting.
    - Interestingly, neither of these are traditionaly considered “good” behaviours in society. What’s more, the second activity will actually be made far easier (the communicating in secret part, not the success of the cartel) in a technologically advanced society with ubiquitous encrypted communication.
  + I can think of plenty of sinister things that would be made more possible by lack of privacy in a society with a powerful centralized authority (think 1984), but I’m struggling to come up with anything that would be truly harmful to liberty or quality of life in a libertarian society like a Zutha society.
* The concept of strong identification of devices could become an increasingly important paradigm as AI advances toward and eventually beyond human capabilities
  + One of the things people often fear about the future is that super-intelligent AI will take over the world and relegate human beings to amusing pets at best, and ashes at worst.
  + Initially it would be expected that the human makers of advanced AI would attempt to make it friendly by nature to human beings
  + In order for malicious intent towards humans to arise in advanced AI it would have to either evolve from (or be created by) AI which is friendly to humans, or be an unintentional feature of AI designed by humans, or be a deliberate feature of AI designed by humans.
  + For the first two cases, all humans including the makers of the AI would want to eliminate this attribute of malicious intent towards humans. For the 3rd case, everyone except the maker would want to eliminate its animosity towards humans.
  + Now consider that there will never be a single super-intelligent AI entity that is far superior to humans and all other AI. Evolution of AI complexity will advance smoothly and there will be a lot of AI software and devices taking many different forms, controlled by many different human agencies.
  + The first AI that deviates from accepted behaviour by say, killing a person without justification (such as because it was commanded to defend its owners against enemies), will be opposed by humans and other AI alike, and will easily be defeated. The mutant AI will than be either destroyed or modified so that it no longer has a desire to harm people.
  + Now imagine that some power-hungry fool builds or trains many AI beings to be loyal to him, but to obey their official masters until the time comes. He plants an army of these undercover AI beings in civilization and then turns all of them at once into killing machines with the goal of asserting power over the civilisation.
  + It’s very difficult to imagine what form the minds of extra-human AI would take and what it would for them to be loyal to their human creator. If AI were really to be more intelligent than humans, presumably they would have to have at least as much autonomy and independent thought as intelligent humans to really be a menace.
  + There are pretty hardwired attributes of human psychology, such as our aversion to death and enjoyment of sex, which remain potent in driving our behaviour despite our free will and independence. It is precisely through means such as these that I presume advanced AI would be “programmed” to be averse to harming humans. I also presume this is how a crackpot would “program” his “loyal” AI to be covert killing machines.
  + I think the power hungry creator of these destructive AI would be likely to regret his genius eventually, because it would be very difficult to program something as subtle as “kill some people but not me” as a hardwired attribute. Releasing those destructive AI into the wild is likely to eventually lead to a super-intelligent artificial species with the driving goal of their existence being “kill humans”.
  + I think the key to preventing this scenario is accountability.
  + If the rest of the world knew that all those undercover killing machines were associated with the ambitious maniac, they could do something to stop the maniac before he had enough Destroyer AI in place to launch his attack.
  + Let’s imagine a world in which every intelligent entity from surveillance drones to people to super-advanced AI has a cryptographically strong public identity.
    - When an identity is first created, it is associated with a creator who already has a strong identity
    - An entity will require a minimum reputation to be allowed to register some kinds of identities; particularly identities for dangerous devices or very intelligent beings
    - When an entity creates an identity, it has to stake its reputation on that identity, so if you create a device which goes out and starts slaughtering people, you will be held responsible and likely lose a lot of reputation (a milder example might have been more appropriate here)
    - Creating a biological human would require much less reputation than creating a new kind of advanced AI.
      * Perhaps creating a human would require only as much reputation as every human automatically has
      * Then again, it could be that a society decides that creating a child is a great enough responsibility that you need a non-trivial amount of reputation to be allowed to, or at least to be allowed to take responsibility for it.
  + Now let’s consider how the power hungry maniac’s plan would play out in a world of strong identity
    - If, as is likely, the maniac doesn’t have a very high reputation, he will not be able to get new AIs registered and so any he creates will be destroyed or impounded
    - Let’s presume that the maniac does have a lot of reputation, as well as a lot of resources
    - So he creates a large quantity of his covert killing machines and sells them to various people and organisations in society
    - When someone else purchases one of his AIs they will gain some influence over it, and be given some amount of responsibility over it, but since the maniac was the creator, he will retain some responsibility also.
    - The maniac will have to have a very large reputation to be allowed to create so many powerful AIs. He will probably have to be in control of large company with a very good reputation for manufacturing AIs.
    - When the maniac launches his attack, it will be immediately obvious that the maniac (or his company) is the common owner of all the murderous AIs.
    - All AI created by the company will come under suspicion and be destroyed or neutralized.
    - It will be easier to defeat the army of murderous undercover AI when they are all instantly identifiable than when they can act covertly as in the scenario without strong identity
    - The maniac and his company will lose all their reputation and be punished severely.
    - Since companies in a Zutha civilization are likely to be more cautious of permitting a single point of control – and thus a single point of failure – there are likely to be others in the maniac’s company with a stake in its reputation that would oppose him.
    - Crucially, if the maniac’s minions were to create AIs of their own, they would be identified as created by an AI created by the maniac (or his company).
  + There is no need to distinguish between human and AI in this model; it will hold up fine in a world where the distinction between human and AI begin to blur
    - Consider a being, X, created by a reputable creator who starts off with a friendly attitude towards humans, but over time begins to develop a disdain for them
    - X may have accrued a reasonable reputation so far, enough to register some new intelligent beings of his own
    - X decides not to program (or train) his creations to be respectful of humans, but decides to have them be disdainful of them also
    - Eventually the disdainfulness of these beings will reveal itself in the form of actions which show disrespect to humans. For example, they might kill or torture a human at the extreme end of the continuum or steal a human’s property or just ignore or insult a human. Such actions will incur punishment or criticism in proportion to the crime or slight which will be costly to these beings’ reputation.
    - These beings are unlikely to gain a high enough reputation to be allowed to produce intelligent beings of their own, and their reputation may fall enough to harm the reputation of their creator X.
    - If X loses enough reputation, he will not be allowed to create more intelligent beings of his own and thus the spread of his anti-human meme will be halted
  + Effectively what this system is doing is imposing an evolutionary force anchored to reputation instead of just survivability
  + Provided the pro-human meme remain strong across civilisation, humanity would not face the risk of extermination by AI, or even relegation to the status of pets
  + Unenhanced biological humans might fall out of positions of power and may cease to be the primary innovators in the world, but they would be better compared to the disadvantaged and undeducated humans of today than the dogs of today, because they would still have the same rights as enhanced humans and advanced AI.
* How would the strong identification model extend into cyberspace?
  + In order to affect the real world, a cyberspace entity has to be given access to a real world device
  + This means that either the real world entity that created the device or currently controls the device has to grant access to the cyberspace entity, or the cyberspace entity has to hack its way into the device to control it.
  + There will always be an ongoing battle to defend against hacking; the success of that battle will determine the capacity of cyberspace entities to affect the real world through that avenue.
  + As for the other avenue, if all devices are identified and associated with strongly identified creators and possibly owners, then it is the responsibility of the creator and owner to ensure only trustworthy cyberspace entities are granted access to their devices. The reputation of the creator and owner is at stake.
  + Cyberspace entities can be given strong identities as well. A reputable agency can vouch for the trustworthiness of a cyberspace entity by digitally signing it or providing it with a private key for digitally signing its actions. If the reputable agency can trust the cyberspace entity, it can trust it to keep its private key to itself.
  + If a cyberspace entity shares its private key or lets it be stolen, and another entity uses that stolen identity to commit a crime, the reputation of the authentic cyberspace entity will be placed in jeopardy and all enities with that key may stop receiving access to devices and services that affect the real world.
* How to defend against bombing from despot governments?
  + It might be easy to design non-fatal weapons for use against foot soldiers, but neutralizing a fighter jet without killing the pilot would be very challenging.
  + Many military aircraft have eject capability, so it might be possible to do enough damage to fighter jets to force the pilots to eject without doing too much damage that it kills the pilot before they can escape.
  + It would be very useful to be able to communicate with the soldiers fighting for the opposition to explain to them that they can quite easily avoid death and actually gain access to a far better life than they’d had, if they surrender.
  + Zutha might have access to enough intelligence resources to be able to discover enemy communicaton frequencies and maybe crack their encryption codes. Who knows how sophisticated the communication systems are of the average African Air Force.
  + It might also be possible to launch intelligent weapons at enemy craft which attach themselves to the hull and communicate verbally with the pilots in their native language by sending sound vibrations through the frame of the aircraft. It could say something like: “Please tell your comrades that you are all welcome to settle peacefully in our community or go wherever you desire provided you don’t harm anyone. We will provide you with free food and accommodation for a week if you want it, or free transport to anywhere in the world. Eject now and you will not be harmed, but hesitate too long and you will die because there is a bomb attached to the bottom of your hull which will explode in 30 seconds.” It might be challenging to devise a message that would be believed.
  + The hardest part about implementing the expensive rewards for enemy soldiers that surrender would probably be convincing enough locals to support it.
    - It would easily be worth the cost since it would potentially prevent many casualties and a lot of property damage that would have been done by hostile pilots before they were taken down. The greatest benefits of the policy would be difficult to measure or even perceive. The pshychological effect on the opposition’s soldiers, including those back at home that have not yet been deployed, would be enormous, especially once word started getting back that the offers were authentic. It would be interesting to see how many soldiers would be willing to abandon their jobs and head off with their families by foot to find the nearest Zutha civilisation. More likely the message would cause immense confusion among the opposition’s troups, making them far less effective and liable to revolt.
    - The locals already living in Zutha communities may feel resentful of free services being provided to their “enemies”, especially if the attacking forces were of an ethnic group that isn’t liked by many of the people in the Zutha community.
  + Zutha would have a networked array of sensor devices surrounding its communities both in the air and on the ground which would alert everyone to the approach of hostile forces. It should be possible to neutralize enemy forces before they get close enough to Zutha communities to do much harm.
  + Zutha will be vulnerable early on when it doesn’t have as many defensive resources. I think it would be important to save up a great deal of resources and defensive military technology before risking serious antagonization of a despotic government.
* Security system
  + HD video streams from many cameras analysed by many different computers and people in real time.
  + This would be effective for finding snipers, threats and blackmail (via body language cues), as well as all obvious crimes.
* Send swarms of distributedly controlled UAVs with knock-out weapons into North Korea and simply prevent all violence and deactivate all weapons. Then start giving out internet devices to people and let the world get to work enlightening them.
  + I wonder what would happen.
  + The government would have no one to attack except the UAVs, which would be very difficult to defeat.
  + Until they did destroy the UAV swarm the government would have a hard time oppressing its citizens.
* Zutha chooses a required interest rate it demands from its money managers. Some of this goes to owners of money, some to zutha. Any excess return money manager can get is theirs.
  + If a money manager ever fails to pay interest or send money, it will immediately stop receiving money.
  + It will be have to successfully pay interest and transfer any money requested by zutha, for a period of time, before it will start receiving money again.
  + The money manager will also lose a lot of reputation on zutha. They will have a lot to lose because only high level users will be allowed to manage money.
* How to deal with terrorism?
  + Car bombs? Suicide bombs? Anthrax? Hostage taking?
  + Many threats will be dealt with by advances in detection technology and surveillance. New technological threats like nanobots and armed UAVs will be handled by decentralization and strong identification of devices.
  + Would there be a need for preemptive attacks on terrorists or secret interrogation? I hope the latter would not be necessary. Preemptive attacks could be effective and I can't see that they would pose a threat to long term global freedom (provided they were done with full openness and accountability after the fact) the way secret interrogation could.
  + How would a terrorist be tried?
    - A philanthropic protection agency would represent them as part of their role of protecting human rights.
    - The prosecutor might be a coalition of protection agencies from the community threatened by the terrorist.
    - The facts used to identify the suspect as a terrorist would need to be made public. A secret military trial like those sometimes done in the US for terrorism cases would not work in a zutha society.
    - This might mean exposing informants to danger.
    - Protection agencies would just have to find ways to get evidence against terrorists that doesn't put human beings in danger. This will become increasingly possible with advancement of surveillance technology.
* Some decisions need not require zuth.
  + Votes could simply be given, weighted on reputation, without the ability to sell.
  + In this way decisions could be weighted only with respect to domain specific reputation and zuth or reputation earned outside that domain would be irrelevant.
  + For a restricted domain this could mean a true representative democracy, where influence is assigned democratically over a continuum.
  + I guess wealthier users would still be able to vote more times to increase other users' reputation, though.
  + Perhaps the only way to achieve a truly isolated system with a democratic base would be to create a separate Zutha network.
    - Independent Zutha communities might want to have such isolated systems
    - They could still tap into the main Zutha knowledge base
    - User accounts could be linked across multiple independent Zutha networks
    - Reputation in an independent network would not be able to contribute to reputation in the main Zutha network though
* There are some ironic parallels between my Zutha and the Zutha from PathFinder. For example:
  + "He was described as being mightier with the pen than the sword"
  + He's a gluttonous ruler over a rich kingdom
  + He entrusted his essence to three volumes of a book which were "scattered to the farthest corners of Avistan."
  + It's a pity that the PathFinder Zutha is an evil undead monster.
* Zutha should save a percentage of income for use on one-time large expenditures like buying a batch of military or surveillance devices, bribing a government to leave it alone, buying a solar power satellite, lending money to a community which is just getting started etc.
* Zutha money managers should be partial owners of Zutha so that they have a lucrative and powerful privilege to lose if they fail to return the money they hold on Zutha's behalf when demanded. Anyone who adds some metadata to Zutha or contributes some code, and gets praised for it, is a partial Zutha owner so there will have to be a minimum degree of ownership to be allowed to hold Zutha's money.
* It would be useful if there were an online service that implemented the Zutha client logic for money managers. Money managers could then provide their identification key and bank authentication details to one of these services and let it manage zutha transactions.

# Mon 31-Jan-11

* Protection agencies can make a policy of reimbursing each other for stopping crimes against their clients or catching the perpetrators. This would make the protection system more efficient while retaining large redundancy.
* ID device
  + Mobile app with private key connected to pc or other internet device (including cash register, security door etc.) via wifi.
  + App shows list of devices available to authenticate with and the actions each can perform (such as log in, open door etc.)
  + User selects a device and action and enters a password or fingerprint into the phone to authenticate the action with the selected device via the phone.
  + The phone simply encrypts a message containing the device id, the requested action and the current time with its private key. It also sends its public key unencrypted.
  + The device unencrypts the message with the public key and looks up the public key in its database or a public database. It then knows that the identity associated with that public key has given permission to do the requested action.
* How can environment be protected from harm by landowners?
  + Initial owner can impose restrictions on harming the environment as part of the land rights.
  + Community can pay an existing owner to impose new restrictions on his land.
  + How would a fair price be found?
  + Should the community be able to force the owner to accept the new restrictions?
  + A strong enough community sentiment should be allowed to force the existing owner to sell at market price.
  + The environmentalist philanthropic agency could purchase the property at market price, impose the environmentmental restrictions and then sell at the new market price, potentially back to the original owner. The environmental agency's losses would be the cost of the environmental restrictions to the value of the land.
  + The environmental agency would then be the owner of restrictions on the land and would have the right to police their imposed restrictions and extract fines for breaching them.
  + Problem: the owner of the land could get a friend to bid an enormous amount, forcing the environmental agency to pay more than the property is worth. The owner could promise to buy the land back from the friend at the same price if he ended up winning the auction.
    - There must be some way this is dealt with generally with auctions. I'll have to find out what it is.
  + A landowner could offer to buy the restrictions back from the environmental agency and abolish them. A large enough public sentiment should be allowed to force the environmental agency to sell their restriction privileges at market price.
  + Environmental damage which affects people outside the property where it occurs should be taxable by the affected people.
* A land developer can impose restrictions on the land he sells, such as that all roofs in the area must have the same tile pattern for aesthetic appeal. He may sell the rights to these restrictions to the community that occupies the properties he sells. The rights would be a bulk agreement so anyone wanting to breach the agreement would have to get permission from the whole community that collectively owns the restriction privileges.
* Ocean overfishing market failure
  + Zutha could claim ownership over areas of ocean and sell areas to the highest bidder.
  + The owner of an area of ocean would have an incentive to sell fishing rights and manage the fishing rate so that his resource remains valuable.
  + The price he rents his water for would be his reimbursement for his service to society of policing the fishing of his waters to prevent overfishing.
  + It would be important that there be a lot of competition for ocean ownership to push the price of fishing an area close to the marginal cost of policing that area for overfishing.
  + The "owner" of an area of water would not have absolute rights over the water, only specific rights relating to fishing regulation. They would not be allowed to stop people entering their waters for example.
  + Zutha would specify the rights it claims over the water and sell these rights on.
  + Zutha could distribute the proceeds of selling the oceans among all countries of the world.
    - This might be perceived as fairer than zutha distributing the money as its community chooses because that community does not represent the rightful owners of the currently unclaimed oceans - which arguably is the whole world.
    - Whether governments are just representatives of the people they rule over is of course questionable.
    - I think it would be fairer to give every human being an equal share of the proceeds of selling the oceans, but this would be logistically difficult.
* It will be important that property owners be able to prevent invasion of their privacy on their property.
  + There will likely be a battle to achieve this as technology makes spying ever easier.
  + Reputation will have a roll to play in punishing those who publicize people's private lives.
  + Privacy is important for people to be able to feel secure and free to be playful, creative, rebellious and innovative.
  + Privacy will be able to be found in one's home or away from public property such as in nature. I don't think privacy needs to be defended in public spaces like streets and other people's property in a libertarian society.
  + The restrictions required to impose privacy in these contexts where it does not really belong would do more harm than good I think.

# Tue 1-Feb-11

* A decision item can be restricted to a weighted combination of reputation domains.
* It will be important to encourage a culture of expecting openness from companies and charities so that it is not a disadvantage to be open and have good practices rather than just closed and unknowable.
* River ownership for pollution, fishing and traffic congestion policing.
  + Owner of downstream segment of river would tax adjacent upstream owner for pollution that crosses the boarder.
* Private Health Care with charities, which will compete for donations and zutha funding, supporting the poor.
* No drug patents. Drug inventors and to some degree manufacturers have an incentive to pay reputable testing agencies to put their stamp of approval on drugs, giving more people the confidence to use them, which will increase the drug's popularity (if it deserves it) and thus the drug company's profits via proceeds from investment in the zutha item representing the drug.
* A culture should be encouraged to push people, via reputation based incentives, to invest in the products and services that benefit them - especially with things like drugs and software which will very often be available free or practically free in a zutha society.
* Protection agencies could bundle life insurance and severe injury insurance into their service to provide them with an incentive to actually protect their clients from harm rather than just pay for the legal costs of prosecution.
  + Clearly this would be an attractive deal for consumers.
  + Protection agencies could promise to pay other protection agencies if the other can prove that they defended one of their clients from harm.
  + How would protection agencies be stopped from staging a protection racket by setting up an apparent attack on someone else's client and then heroically saving them?
    - The reputation and business they would have to lose if caught would be a huge disincentive.
    - It would also be hard to convince anyone to play the skapegoat considering how likely it would be that they are caught.
    - Given the level of redundancy in the system, there will be many independent watchers of public spaces, so it would be very hard for the duplicitous protection agency to arrange to be the only one at the crime scene.
* Commercial Postal Service
  + Zutha pays a proportion of postal costs in rural areas.
  + Postal companies still have an incentive to reduce costs for rural citizens because they will still be shopping for the cheapest service since their costs scale with the real cost.
  + Society would do this because it values the ability of rural citizens to access more services than would be available to them with a strictly efficient allocation of resources.
* Provide free access to the Zuthanet to everyone with a Zutha client and a connection to a host
  + Many clients should also act as low level hosts, facilitating a distributed network requiring no fixed hardware
  + Some Zutha items will represent web pages. These should store a regularly updated cache of the pages they represent so that these can be served across the Zuthanet to users without internet access
  + It might be possible to serve some kinds of web resources over the Zuthanet by relaying communications between Internet servers and Zutha clients.
  + Provide free global access to Zutha via satellite and permanent airborn hosts. Priority should be given to high reputation users when all traffic cannot be handled.
* Clients should have reputation that is only available for praising hosts for good service
  + A client will receive reputation for reporting inconsistencies between hosts’ responses
  + Most – perhaps all – clients should also act as hosts
  + Every node (host or client) on the Zutha network will have a unique ID listed in a dynamically changing public directory
  + A client may select from any of the currently known nodes when it requests data
  + Nodes with a higher reputation should be given a higher probability of being selected, but only by protocol since there would be no means of forcing clients to observe this rule
  + A new node can only initially acquire reputation by contributing useful information it discovers from making requests. Useful information would include:
    - Discrepancies between data received from different hosts
    - Reporting that a host does not have the latest data for a particular request
    - Statistics about the time taken to receive requests from different hosts. It might not be possible to check the validity of such stats so it might not work to have this information rewardable by reputation.
  + Once a node has some reputation it will start receiving a salary of zuth
  + When a node A requests information from another node B and B replies successfully and correctly, A should automatically invest a certain amount of Zuth in this action, boosting B’s reputation
  + The calculation of how much zuth A invests in B for a correct response will be done automatically (not within the control of A) and be based on A’s current pool of zuth and reputation
  + If B fails to respond in time to a request, or responds incorrectly, A invests negatively in this action, reducing B’s reputation
  + A can produce proof of either a correct or incorrect response from B but it may choose not to. There is no way to force A to give up this information.
  + A can be prejudiced for a host by neglecting to report its failings, and prejudiced against a host by neglecting to report its successes. There should be very little incentive to bother doing this. If A is friends with B and B makes an error, A may choose not to report this, but A should still tell B about its mistake since other nodes will report B’s errors if it continues making them.
  + A only has a limited amount of zuth to invest, so it cannot significantly reward another node by making a huge number of requests from it simply for the purpose of giving it some reputation for each successful response. A will soon run out of zuth – though this should happen asymptotically.
  + It would be really good if there was a way for a node A to prove that another node B did not respond within a particularly timeframe as this would allow B to have its reputation (and thus its traffic) reduced automatically when it begins to be overloaded.
    - Actually the only thing under B’s control is how quickly it processes incoming messages. The time taken for the message to get from B to A once B has created the message is out of B’s control.
    - When B processes A’s message, B includes the hash of A’s message and the key for the current timestamp in its response.
    - If A doesn’t receive B’s message within the timeout period, A should report this to the network.
    - If B cannot produce a signed message that was created an acceptable period after it received A’s message, then it will have some reputation deducted.
    - But how does B prove when it received A’s message?
    - What happens if B never receives A’s message and how does B prove it if this happens?
    - The whole path of a message could be traced between the sender and receiver to establish which links slowed down and stopped a message. This seems unworkable, particularly if it is considered that some links between nodes might be managed by entities outside Zutha’s ability to measure or control. For example, if two Zutha nodes are linked via the internet.
    - Also, if the path of messages needed to be traced, this would prevent Zutha from protecting the privacy of users’ location, which is an important feature.
  + Returning an incorrect response or not responding in time should incur far greater penalities in reputation than the gains in reputation incurred by returning a correct response
  + A node should be allowed to refuse a positive investment or invest negatively in itself in order to self-manage the amount of traffic it receives.
  + A higher reputation node should have far more to lose by sending an incorrect response than a low reputation node. In fact, the punishment for sending an incorrect response could be a percentage of current reputation. This mechanism would not be compatible with zuth-based investment though.
  + Perhaps increases in node reputation should require expenditure of zuth on the part of the praising node, but false responses would be punished by simply taking away reputation from the offending node.
  + Reporting a false response by a high reputation node should be rewarded by a small increase in reputation.
  + It would not work to reward reporting correct responses from nodes, because this would allow gaming the system by making an enormous number of requests which you know will be successful.
  + Two nodes could reserve their zuth to only invest in each other by making an enormous number of requests to each other
    - This is only practical if nodes can select which nodes they make requests to
    - A node will always be allowed to make requests to any node it wants, but the selection algorithm could be used to assign an official set of nodes for each request, where only official nodes are allowed to receive reputation.
  + In order for the selection algorithm to be used for view requests, the current set of available nodes must be fixed for the current time period. This is because other nodes can only verify that a request is being made to the correct set of nodes if it uses the same set of available nodes in the selection algorithm as was used by the requester.
  + If the set of hosts for a given time period must be fixed, then a new node will have to wait a while after it requests connection to the network before it is allowed to serve requests. This is not a problem. That node will still be able to make requests, so it can still connect and act as a client very quickly.
* The Zutha.com website will eventually provide a single Client page written in javascript, in addition to the standard interface
  + The standard interface will communicate only with the one host: Zutha.com
  + The Client page will be a javascript Zutha client that provides access to all Zutha features without ever changing urls
  + The Client page will use the selection algorithm to select a randomized set of hosts to connect to for each request
  + It should be possible to save the Client page on one’s desktop and run it from there. Internet access would still be required since that is the only portal through which a browser would be able to access the Zuthanet, but it would not matter if the Zutha.com domain was unavailable.
* Zutha hosts might be connected to their neighbours via the internet, or via direct wireless communication
* Calculation of the TimeFrame Key
  + I’m defining a TimeFrame as a fixed interval of time which corresponds to a fixed set of available nodes
  + Previously I’ve called the TimeFrame the period
  + When a message is created, it is associated with the current TimeFrame. The list of available nodes associated with that timeframe are the set of nodes from which the recipients of the message will be selected using the selection algorithm
  + The TimeFrame key is an unpredictable key which identifies every TimeFrame uniquely
  + If a node receives a message that was sent using an expired TimeFrame key, the message will not be accepted
  + Candidate 1 for calculating the TimeFrame key
    - During each TimeFrame, the selection algorithm will use the current TimeFrame key and the current set of known nodes to select a node to calculate the next TimeFrame Key
    - The problem with this is that the node selected to determine the next key has all the information required to predict which node will be selected to determine the next key. It would thus be theoretically possible for the selected node to engineer its key selection so that the next node selected would be one that is under its control. In this way, a single agent could maintain control of the key selection and it would not be unpredictable for him.
    - It could probably be made computationally infeasible to do this provided the selection algorithm was cryptographically strong enough and the TimeFrame sufficiently short.
    - Still, I would rather it be intrinsically impossible for anyone to predict future keys
  + Candidate 2
    - The unpredictable component of the key is provided by some external unpredictable variable, such as some approximation of the Cosmic Background Radiation
    - This just seems to introduce unnecessary complexity and difficulties and I don’t like it much
  + Candidate 3
    - The entering and exiting of nodes from the network would be an unpredictable variable
    - Using this as the unpredictable input would only work if the set of known nodes changed regularly enough
    - If all clients are going to be a part of this Known Nodes list, then the rate of change of the list would be very high
    - Let’s say that a node that requests entry to the network in the current timeframe will be included in the list of known nodes in the timeframe after next
    - So in any given timeframe, the set of known nodes is known for the next timeframe but no further
    - The unpredictable key for the next timeframe could simply be the hash of the guids of all the nodes that will be included in the next timeframe.
* How does a new node connect to the network?
  + A node A selects a guid to identify itself and sends a connection request out which will be associated with the current timeframe t1
  + This request will immediately be propagated across the network
  + If the A’s selected guid clashes with an existing guid in the Known Nodes list, the first nodes to receive A’s message will reject it and not pass the message on. This is simply to close a possible security vulnerability, since it is practically impossible for two guids to collide accidentally.
  + The length of the timeframe would be chosen to ensure that there is a very high probability that a message could propagate to the vast majority of nodes in the network in that time
  + About halfway through the next timeframe t2, nodes would start asking each other for their calculated timeframe keys (the hash of their Known Nodes list) for t3
  + If a node B has not received A’s join request by halfway through t2, then it will find that its calculated timeframe key for t3 will differ from the majority of other random nodes it asks. In this situation, B should ask some of the nodes whose timeframe key matches the consensus to send it all the join and exit requests applicable to t3 – that is, join and exit requests that occurred in t1.
  + If A only sent its join request successfully to one node and that one node only successfully sent it on to a small number of nodes before the propagation stopped, then there would be a small group of nodes with a different Known Nodes list than the majority. This minority should reject A’s request when they discover that it was not received by the majority of nodes by halfway through t2. A should be notified so that it can resend its join request.
  + It is extremely unlikely that a request would propagate to a significant proportion of nodes without reaching practically all of them, because once the message has reached a large number of nodes, the likelihood that it will have reached hub nodes with very strong connectivity will be very high. Thus it is safe to trust that if a node discovers that its Known Nodes list differs from the majority of a random selection of other nodes at a time halfway through t2, then that node must be of a significant minority and should conform to the consensus.
  + In timeframe t3, A is added to the list of Known Nodes
* If there is going to be no change in the Known Nodes list in the next timeframe, the timeframe key will remain the same for two periods in a row
  + Theoretically, the timeframe key could remain the same for many timeframes
  + If this was a concern, the Known Nodes list could consist of a set of pairs (guid,rand), where guid is persistent id for a particular node, and rand is an arbitrary random number selected by each node.
  + If a significant portion of a period passes without any entry or exit requests, some nodes should voluntarily change their random number to ensure that the timeframe key is not the same for two timeframes in a row.
* How can nodes be punished for being slow to process requests?
  + This is important so that nodes don’t slow down the network by being so greedy for high volume (and the associated reputation) that they cannot process all the requests in time
  + The problem is that it is very difficult to prove who is responsible for a message being delayed or blocked
  + Maybe it isn’t important to concretely isolate the node responsible for a delay. It could actually be useful to reduce the reputation of nodes that are hard to get to. From the perspective of the rest of the network, this is just as bad as the node being slow at processing its requests.
  + A client that fails to receive a reply from a host in time can send a complaint out to the network containing the signed replies from the hosts that did reply to the message.
  + There should be an algorithm to decide which of the target hosts should forward the message to each other, in order to ensure that the client is unable to deliberately refrain from sending a message to a particular one of the selected target hosts.
  + There should also be an algorithm to specify which other target hosts a host should forward its reply to.
  + If a host is accused of failing to respond in time by a client, the hosts it was expected to forward its reply to should report on whether they received the reply. If none of them (or too few of them) report receiving the reply, then the accused host should lose some reputation.
  + The accused host may not have been responsible for the delay or blockage of the pipeline between it and the other target hosts and the client, but it is punished for being in a part of the network with low connectivity. The purpose of this reputation system is to allocate requests efficiently to the hosts who will respond most rapidly; the cause of the slowness of a host to respond is irrelevant.
* When a request to a particular node is delayed or fails to get through at all, the probability that messages take that path in the future should be reduced.
  + When a node learns that a message it relayed failed to be delivered in time, it should reduce the connectivity weighting on all the neighbours it sent that message to.
  + A node or connection (including an internet connection) which is commonly the cause of network delays will often have its connectivity weighting reduced by its neighbours because messages that go through it will often be delayed.
  + Neighbours with a low connectivity weighting should be used probabilistically less frequently
  + Connectivity weightings are private to each node.
  + In this way, paths with low connectivity will be used less often, without anyone knowing what those paths are.
  + The physical location of a node in the network will still be secret to everyone except its neighbours.
  + Connectivity weighting is combined with node-specific connectivity weighting to determine the probability of using a given neighbour to relay a message
  + Node-specific connectivity weighting is determined by the distance (in terms of number of hops) between a neighbour and a given node
  + This distance is found by averaging the number of hops reported by each incoming message from a given node through a given neighbour.
* What is node reputation used for?
  + A node’s connectivity reputation determines its relative likelihood of being one the targets of a request.
  + Connectivity reputation also indicates how much load a node is taking
  + Nodes that are taking high loads would incur greater costs and should be paid more for their services. This will happen naturally by dividing pay in the normal way: proportionally to reputation.
  + I had also planned that hosts would be name servers and even holders of money or owners of devices
  + I now think it more appropriate if users be the holders of money and the partial owners of devices, since their reputation is based more on merit and loyalty to Zutha. It is also harder to get user reputation because it requires contribution of real value as assessed by the Zutha community, so users with high reputation have more to lose than nodes with high reputation achieved through pure expenditure of computational resources.
  + Node connectivity reputation is only a measure of resources, so it is not a reliable metric for trustworthiness. An untrustworthy resourceful entity who wants to infiltrate Zutha in some way can easily contribute very powerful hosts if that will gain them privileges – thus it should not… apart from the privilege of handling more requests and being paid more for the service.
  + What about name servers?
    - These will need to be nodes with high connectivity reputation since they will need to handle a great deal of requests and be reliably available
    - A name server does not really have much opportunity for disrupting or infiltrating the system.
    - The main purpose of a name server is to timestamp each item (and assign its ID of course)
    - The name server cannot manipulate the request itself because it is signed by the requesting user
    - The name server cannot significantly manipulate the timestamp it assigns because it must distribute the timstamped request immediately to an auto-selected set of request manager nodes. If these manager nodes receive the request too long after the specified timestamp, the request will be rejected.
    - The average time between a request’s timestamp and the time the request is received by the request managers for a given name server should affect that name server’s reputation.
    - Thus if a name server consistently reports a timestamp that is too early, it will lose reputation and be less likely to receive requests to timestamp items in the future.
    - If a name server is ever caught reporting a timestamp that is after the moment it processing the message it should be severely punished. It would only be caught by one of the request managers receiving a message before its purported timestamp.
    - A name server could deliberately delay sending a message to bully a client.
    - The request managers should receive the initial request from the client as well. They could then measure how long it takes the name server to process the message.
    - The request managers should give the name server the go-ahead when they receive the raw request
    - The timestamp must lie between the moment the first request manager received the raw request and the moment the first request manager receives the timestamped request.
* It would be good if being in a relatively isolated part of the network lead to the selection algorithm prioritizing nodes in that part of the network.
  + Data would still be shared with the main network by hosts pushing messages to each other, but the bulk of client messages would be sent among the local hosts, reducing the load on the bottleneck between the isolated network and the main network.
  + It wouldn’t work to have the selection algorithm work differently for different parts of the network. I can’t think how this would work, but even if I could, it would allow the selection algorithm to be manipulated by orchestrating a controlled isolated section of the network
  + Ultimately every request has to get to every superior node in the network, so any bottleneck is just going to have to handle this. If it doesn’t, traffic to that isolated part of the network will be slow
  + The only way things can be sped up for the isolated part of the network is through caching. This would only help for view requests of course.
  + Nearby nodes can respond to requests even if they aren’t one of the selected request handlers
  + I need to figure out an efficient way for nodes to decide whether they should respond voluntarily to requests.
  + It would not be desirable for every node involved in transferring a request to also provide their own response
  + It would be ideal if more than just the direct neighbours of the client were allowed to respond voluntarily, though.
  + A view request could carry with it the latest known data according to all nodes up to that point in the path.
  + When a node Y in the path has more data to add, it sends back this data along the path. All the nodes in the path from the client that made the request to the node before Y can then update their databases with the data sent from Y.
  + The first superior node in the path will probably be the last to send back data before the request handlers, because a superior node is likely to have the latest data.
  + The request handlers will still be responsible for confirming the validity of the response.
  + In most situations, the data returned by a client’s nearest superior node will be correct, but since the superior node nearest a client is knowable by an attacker, this data is theoretically falsifiable.
  + A node which volunteers data can still be punished for returning false results. The client just has to propagate the signed false results across the network to prove it.
  + It is unlikely that anyone will bother to try to falsify results with this much resilience in the system.
  + Some clients that want to conserve resources may choose to accept the results they get from the closest superior node and not bother processing the redundant data from the request handlers.
* Nodes with partial databases
  + Many nodes (such as tiny UAVs) will not have the storage capacity for the whole Zutha knowledgebase
  + Nodes with incomplete databases should still be able to respond to view requests
  + I’ll call nodes with incomplete databases inferior nodes from now on
  + Inferior nodes should not have all new data pushed to them
  + Instead, they should act like caches. When they process a request, they should cache the results – deleting any data that has not been needed in a long time – and return this cached data next time they receive the same request.
  + The handlers of view requests should be selected from the whole Known Nodes list, which will include inferior nodes
  + When an inferior node receives a view request it will immediately respond with whatever data it currently has that matches that request.
  + An algorithm should specify an efficient, but redundant way for the selected request handlers to pass their results to one another
  + Any inferior nodes among the request handlers will be able to update their databases once they receive the latest data from the superior nodes among the request handlers
  + Inferior nodes will also be able to update their databases when they act as conduits for messages they are not involved in processing.
  + Inferior nodes should accumulate a space-efficient database of the most common data requested
  + Inferior nodes that are neighbours of the requester (or even just very close to the requester in terms of number of hops) should voluntarily return a quick and likely out-of-date response to the requester even if they aren’t one of the selected request handlers.
  + Neighbouring inferior nodes should voluntarily return their cached response to the requester if it is less than a maximum age or contains a minimum amount of data.
* The request lifecycle
  + I’ve decided I want to unify all request types (view, edit, invest etc.) with the same mechanism
  + A client C (which may also be either an inferior or superior node) generates a request R during timeframe t1
  + C uses the selection algorithm with R and t1 as inputs to select the request handlers from the Known Hosts list and the Master Host from among the Name Servers (which I might make the same as superior nodes)
  + C sends R out to a random selection of his neighbours. The probability of sending to a given neighbour is weighted on the reputation of that node as well as its connectivity weighting with respect to C.
  + Let N1 be one of C’s neighbours that receives R
  + N1 adds 1 to the hop count of R and sends it on to one or (with a low probability) two of its neighbours. The neighbour(s) selected is the one with the smallest distance to any of the request handlers.
  + If R is a view request and N1 has some data matching that request which is more recent than the data C has (specified in the request), then N1 should return that data immediately to C
  + Let N2 be a neighbour that N1 forwards R to
  + According to this request, N1 is a hop distance of 1 from C, so N2 should update the average distance of C from N1 in its private connectivity database
    - Actually I don’t think the average hop distance of C from N1 should be recorded, but just the last distance
    - If C moves around, the fastest route to it should be updated quickly. This wouldn’t happen if the distance to C from N1 was measured as the average of all its past distances from N1
    - I need to think more about how to efficiently find short paths between nodes while also allowing these shortest paths to quickly change as nodes move around relative to each other
  + N2 forwards R on to more nodes in the same way as N1
  + Each of the request handlers are assigned (by the selection algorithm) a small number (maybe 3) of other request handlers to forward the message on to for redundancy. The arrangement of forwarding targets is such that even if only one request handler initially receives the request, all of the request handlers will eventually get it.
  + It will be unpredictable which of the request handlers get the message directly from C. The mechanism of having intermediate nodes forward to the neighbour which is closest to the closest of the request handlers will lead to, roughly speaking, the closest of the request handlers to C receiving C’s request first.
  + I’m having second thoughts about this mechanism.
    - I think a better way is for C to send a separate message to every request handler and the Master Host so that each message will follow the shortest path to that node.
    - C would then send each message (each addressed to a different target request handler) to the neighbour which is closest to that request handler.
    - If C wanted more location privacy, it could send its messages to completely random neighbours, potentially included any node accessible via the internet
  + As soon as the first request handler H1 receives the message it forwards it to 3 other request handlers and notifies the Master Host, that it can now assign a timestamp to the request
  + If R is a view request, H1 will return the results of the request to C immediately
  + The other request handlers will also return their results to C as soon as they receive R
  + The results are returned to C by backtracking the path taken from C to each request handler.
    - Actually there is nothing different about the approach to getting a message from H1 to C as from C to H1
    - Because a message has passed from C to H1, all the nodes in between have updated their connectivity databases to record which neighbour is closest to C (the one through which they received the message from C)
    - Actually I’m not certain how the connectivity weightings will be measured exactly yet; it may be that the path taken on the way back to C will not necessarily be the same as the path taken from C to H1
  + Even if R is not a view request, H1 (and the other request handlers) may return a preliminary response to C indicating whether the request would be successful or not according to H1.
  + If all the request handlers returned a fail response to C, the user should be alerted that the request failed
  + If all (or most) of the request handlers return a success response to C, the user should be alerted that the request is pending.
    - At this point, if the request is to create an item, the item will not have an ID or timestamp assigned yet
    - If the request is a modification request, the user’s client might show the modification as preliminarily successful, but without an associated even item to confirm its success
  + Once the Master Host receives the go-ahead from H1, it creates a timestamp for the request and assigns IDs to all the items that will be created in response
    - There will be at least one item created for any kind of request, including a View request
    - A View request simply generates a View Event item
    - All requests generate one Event item, which needs an ID
    - If the request is to create an item, then the new item will need an ID assigned as well
  + The Master Host, M, sends its completed items to all the request handlers
  + If M’s assigned timestamp is earlier than the time H1 gave M the go-ahead, then M will be severely punished
  + The request handlers again forward M’s message onto each other in a redundant, but not too inefficient way.
  + If a request handler Hx does not receive a message directly from M or M’s message arrives late (more than one timeframe after the timestamp), then Hx should report this
  + If the majority of request handlers report a failure or delay in receiving M’s message, then M should lose reputation
  + Regardless of whether M loses reputation, the paths (perhaps incomplete) between M and the request handlers that report a failure should be weakened.
    - Every node involved in relaying the message between M and Hx (or another node that reported a failure) should weaken the connectivity weighting with respect to Hx of the neighbour that they sent this message to
    - Hx does no real damage to M by falsely reporting a connection failure and there is no other incentive I can think of for Hx to report falsely. The only effect of a lone failure report is a slight rearrangement of the paths messages will take between M and Hx
    - Hx does have an incentive to report a failure if it is a true failure, because if the other request handlers report a failure as well, then M will lose reputation and that will mean slightly more income for all other hosts including Hx.
    - If M fails to send out any message then no paths will be weakened because there will be no nodes which received the message from M intended for one of the request handlers.
  + Provided no conflicting requests have arrived in the meantime, the request handlers will all send confirmation messages to C which include the ID and timestamp of the newly created items.
    - The first request handler to receive a response from M will be the first to dispatch a confirmation message to C
    - Even if a minority of request handlers receive a direct response from M (meaning M will lose reputation), all the request handlers will return a confirmation (or potentially failure) response to C, because they will pass M’s message amongst each other.
* Connectivity weighting
  + When a node X receives a message C via a neighbour Ni, it should increase the connectivity weighting of Ni with respect to C
  + When a node X relays a message M via neighbour Nj toward request handler H, and message M fails to reach H, X should reduce the connectivity weighting of Nj with respect to H
  + In this way successful connections – as between C and X – are strengthened, while failed connections – as between X and H – are weakened.
  + What happens if C connects successfully to X, moves to a completely new location, and then connects to X again?
    - There is nothing valuable about the old information regarding the shortest way to find C, because C is no longer connected to the same neighbours
    - For this scenario at least, it would make sense for X to simply throw out all its old data and maintain just a single closest neighbour to C at any one time – determined by the last neighbour through which X received a message from C
  + It would be important that hosts not move around because they need to be contactable by nodes that they have not directly contacted yet
  + I think the best way to think about this problem is to imagine how the shortest paths will be discovered after the network is first created and as it begins to grow
  + Let’s imagine there are 4 superior nodes already in the network: A, B, C, D
  + Pairs (A,B), (A,C) and (C,D) are neighbours
  + E joins the network and connects to A and C
  + E is asked by X to send a message to D
  + Since E does not know whether A or C is closer to D yet, E sends the message to both
  + C relays the message to D in one step
  + D will record that X is 2 steps away from C
  + D will reply to X via C
  + C will relay the reply to E, reporting that the reply has taken 1 hop from D to C
  + E relays the reply to X, reporting that the reply has taken 2 hops from D to E
  + Meanwhile, A has relayed the message to C
  + C has probably already received the message and knows there is no point sending it to D a second time.
  + C reports back to A that the message has already been passed to D and that there is 1 hop between C and D
  + A reports back to E that there is 2 hops between A and D
  + E now knows that C is the closest of its neighbours to D in terms of number of hops
  + Other metrics can be included in the connectivity weighting including the frequency with which the connection through a neighbour fails, and how often and how much messages through that neighbour are delayed
  + Now let’s suppose that E moves so that it is connected only to B
  + If C had been directly connected to E (as wireless devices within range of each other for example) then C should detect that E has disconnected and should report this to its neighbours A and D
  + Should D forget the fact that C is the closest of its neighbours to E?
  + This information could be useful again if E reconnects to C
  + Also, it is likely that E will reconnect to the network somewhere relatively close to where It was before
  + C’s message that E has disconnected from it should cause the connectivity weighting, from D’s point of view, of E with respect to C to diminish significantly.
  + If E is a high reputation node and it disconnected without notifying the network sufficiently in advance, then it would be liable to lose a lot of reputation if anyone tried to connect to it while it was down.
  + E will likely lose reputation for moving at all, because it will take a long time for E to be “found” again by all the other major nodes in the network
  + Just after E moves, the first few requests sent to E will have to be sent on wide and blind searches to rediscover the shortest connectivity paths to it.
  + When D now tries to send a message to E, the highest weighted neighbour may still be C (in this case D has no other neighbours, though), but this weighting will have been diminished by C’s report of E’s disconnection
  + When C gets D’s message to E, C will have no idea where E is, so it will send messages to E in all directions
  + It would be counterproductive for C’s blind searches to be propagated across the whole network with every intermediate node relaying to every one of its neighbours because it has no idea where E is
  + A node should only send out “seeker” messages (blindly sending out messages in all directions) if the number of hops taken for the message to reach it is less than a predefined maximum
  + If a node finds a known trail to E before this maximum “seeker” limit is reached, though, the message will be propagated all the way to the end of this trail, even if the total number of hops will end up exceeding the “seeker” limit
  + In our example, C will send a seeker message to A, which will send a seeker message to B, which will connect directly to E
  + When the reply is sent from E back to D, D will discover that E is now 3 hops away from C
  + The seeker mechanism will be effective when E has not moved very far
  + E should make it easier for the network to find it after it moves by sending out “locator” messages to be propagated across the network. Every node that receives one of these locator messages from E will be able to assign appropriate connectivity weightings for finding E.
  + If E’s locator message reaches D before D has tried to reach E by the old path, D will think it knows a shorter path to E. Should D use what it thinks is the shorter path but is actually now, unbeknownst to D, a longer path?
    - In the case that E has moved it seems clear that the best approach is for D to simply use the most recent successful path to E
    - E should thus report that it has moved so that other nodes can update their connectivity weightings in the appropriate way
  + An incoming message (say from E to D) should always have taken at least as good a path as any known paths between the same two nodes known by the destination (D). This is because when D discovered its best path to E, it also let E know about that path.
  + How will a possible shorter path between E and D – say by C and B connecting directly – be found by E and D?
    - Nodes should randomly relay messages onto a second neighbour
    - When a node discovers a new neighbour, it should assign a relatively high connectivity weighting to it by default
    - This will mean that it won’t take too long before a message going through C from D to E will be sent to A as well as B
    - C will then learn that B is a closer neighbour to E than A
    - If B had not known a path to E, it should have sent out seeker messages.
    - If B’s seeker messages had petered out and not found E, then C would know that B is a dead end for finding E
    - If B’s seeker messages had found E, but over a longer distance than the distance between A and E, then C would get a reply back indicating this distance and it would assign a far lower connectivity weighting of E to B compared to A
* Location privacy
  + The most ideal scenario for location privacy would be that a person’s client C can change its ID at any time and act like any other node in a message path, and even the C’s direct neighbours don’t know how far along the chain C is.
  + At the moment, the number of hops that have elapsed since the client is carried with the message
  + I don’t think it will be possible to abandon this generally because at some point the relative number of hops between two paths needs to be compared to determine which one is shorter
  + A message marked anonymous could be simply passed along the pre-discovered paths without contributing to the discovery of shorter paths and would therefore not need to carry the number of hops along with it.

# Wed 2-Feb-11

* Protection agencies have an incentive to publicize their surveillance data
  + Some protection agencies might even offer rewards for those who notify them of risks to their clients.
  + One of the main metrics that protection agencies will compete on is the statistical safety of their clients
  + If a protection agency allows the public to help them prevent crimes against their clients, then they will reduce the rate of successful crimes against their clients and improve their business’s reputation, which will bring them more clients and make them more profitable
  + I don’t think there is even a Free Riding issue here, because it will cost very little (in a technologically advanced society) for a protection agency to publicize their data, and since their surveillance data will be concentrated on their own clients, providing it publically will benefit them more then it benefits their competition.
  + There will of course be a lot of benefit to the competition of publicizing one’s surveillance data since the public will be able to use it to identify crimes against competitor’s clients.
  + The better way for protection agencies to compete would be in how much they pay members of the public for alerting them to dangers to their clients or identifying the perpetrators of crimes against their clients.
  + If a protection agency wants to maximize the public’s ability to help them fight crime against their own clients, then the best way would be to publicize their own data, which is targeted towards their own clients. For example, a protection agency is likely to have more surveillance in areas of a city where more of their clients live.
  + Crucially, not publicising one’s surveillance data would not benefit you (as a company) enough relative to the competition to justify the very small cost savings. In fact you would lose slightly relative to the competition because the public wouldn’t have access to as much surveillance data focussed on your clients.
  + If there is an added reputation benefit to publicizing surveillance data then the case for it would be stronger yet
* Open source Genetic Engineering of crops
  + Developing successful GM organisms is purportedly a very expensive business
  + If the cost of developing each successful GM organism did not fall, and assuming development were somehow able to continue at the same rate, who would incur the costs of their development in the absence of patent laws?
  + Income from the donation component of investments in successful crops would reward those directly involved in the invention of the successful crops
  + Investors might include farmers who are pleased by the crops, people who are interested in the GM field and want to contribute to its development, or professional Zutha investors with some knowledge of the field
  + In an open source market for genetic inventions, there are likely to be a lot more people involved in the discovery process.
  + Each independent entity will likely have a far smaller research budget than the typical Biotech corporation.
  + The market will take the form of many independent entrepreneurs seeking to make massive profits by coming making an extremely successful discovery. As with software entrepreneurs, each will probably overrate their chances of success.
  + So a large part of the costs of the discovery process will be incurred by entrepreneurs who fail – just the same as in the software and internet services discovery process
  + The rare biotech entrepreneur who invents something that becomes extremely popular must be able to profit immensely in order for this mechanism to work
  + The critical element will be how much revenue can be generated through Zutha’s investment mechanism.
  + The fact that biotech companies can charge such enormous premiums on seeds that cost very little to produce means that the invention benefits a lot of people a great deal
  + The problem is that if the price of GM seeds is driven to their marginal cost, farmers will have much lower marginal costs and be forced by competition to charge much lower prices for food. The people who benefit from these much lower prices are unlikely to attribute the benefit to the relevant biotech discoveries because of the degrees of separation and the (justifiable) disconnectedness of consumers from the processes involved in bringing them final goods and services.
  + The net benefit to society if the entrepreneurs can be convinced to invent the successful GM crops will be enormous, even if all the biotech entrepreneurs were to incur large net costs.
  + The key, I think will be to find a mechanism for (probably crude) measurements of the benefits of GM discoveries to society to be publicized enough that the discovery becomes worth investing a lot in.
  + The mechanism to incentivize investing would be the expectation that many others would consider the discovery worthy of high value on Zutha.
  + A culture needs to be created of valuing biotech discoveries highly on Zutha. This culture could be facilitated initially by philanthropists frequently investing a huge amount in successful biotech discoveries. This is likely to happen naturally because many philanthropists will recognise the huge benefit to society of successful biotech discoveries and will want to encourage biotech entrepreneurs to keep it up.
  + Once such a culture begins, investment in biotech discoveries should grow by itself for a while, through feedback. People investing for profit will recognise that a successful biotech discovery receives a lot of investment from philanthropists, thus they will try to get in early and invest before the philanthropists so they can profit from selling after the fervour about the discovery abates. These profit investors will be indistinguishable from philanthropists to other profit investors and so the appearance from outside will be that investing in biotech discoveries is generically becoming more popular, and thus more profitable for new profit investors.
  + This feedback cycle has to stop somewhere. I think it is likely to be tempered by people’s subjective opinions about the relative value of different concepts.
  + I predict that the total distribution of investment across items will approximate a scaled up version of the distribution of philanthropic investment across items. This would of course be very difficult to measure because profit investing would be indistinguishable from philanthropic investing. Indeed, there would not be a sharp line between them, because many people would invest through a combination of both motives.
  + So the goal would be to promote the value of biotech discoveries to society and encourage philanthropists to invest in these discoveries.
  + If the amount of entrepreneurship in biotechnology was to fall below an optimal amount according to individual philanthropists, this would encourage them to donate more to this cause. Their increased investment activity would be magnified by the response of profit investors. The overall effect would be a self-optimizing mechanism where the amount of money injected by society into a given field would approximate the relative value of that field to society according to the aggregate opinions of philanthropists.
* “philanthropically” offer to pay for the storage of nuclear waste, or even just offer a competitive, potentially below cost price for the service
  + Accumulate a stockpile of nuclear waste which can be burned profitably in new generation nuclear power plants
  + Invest in developing the new generation fission technology which burns the large part of current nuclear waste
  + Use and/or sell the nuclear material once the technology has made it worth a lot of money
  + There may not even be any need to create or buy a new company to run the infrastructure. A deal could simply be made with the current holders of nuclear waste that Zutha will pay for a large part, or possibly all, of the storage costs in exchange for ownership of the material.
  + The motivation for doing this need not be a secret, though a better deal might be achievable if less people were aware of the motives. It would simply be understood that Zutha is taking on the costs of nuclear storage in exchange for an uncertain probability of being able to profit from the gamble in the future if and when technology allows.
  + Perhaps the bigger hindrance to the potential for profit from this venture would be socio-political resistance to nuclear power
  + A venture funded by Zutha would be based on the confidence that Zutha will create a prosperous open society in Africa which will open doors for profitable nuclear entrepreneurship
* An investment of real money in an item should have properties that make it technically a donation
  + A user donates a chosen amount to an item
  + This amount becomes an asset; an “investment” in the item which is owned by the owner of the item
  + The owner has the right to keep the whole donation at any time, if she chooses
  + The owner can also donate a proportion of the “investment” (which is really a donation) back to the investor
  + The investor receives this return donation as an investment asset in the item, which can be sold for its market value at any time
  + When an asset is sold, either by the item owner or the recipient of the return donation, a percentage of the sale value is taxed by Zutha
  + In order to encourage more investment, it would benefit the item owner to publicize her return donation policy. This could take the form of a simple percentage of donations which are donated back, or something more complex.
  + There would also be an incentive for the owner to follow her publicised return donation policy in order to facilitate trust. If profit investors don’t trust that she will keep her word, they will be less likely to risk investing in her item.
  + It would be important that this return donation policy be automatable without being controlled by the Zutha network
    - This is so that the choice of whether and how much to donate back to any given investor is always under the owner’s control
    - This control might take the form of an owner publicizing a simple rule, but actually operating by a secret, somewhat more complex rule
    - For example, the owner might publicize that her rule is to donate back 70% of investments in one of her items. However, she might secretly define a rule that specifies that nothing should be donated back if the incoming donation is over $1,000,000. This might be because the owner has a prejudice against rich people, because she wants to stop someone acquiring too large a share of her item, or has some other reason to bias investments in her item toward smaller investments.
    - The owner should also be able to step in at any time and manually choose whether and how much to donate back to investors for each donation individually.
    - This might take the form of the owner temporarily disabling the automated response and dealing with each investment manually as it comes in
    - Or she might define a filter in the automated response which refers investments matching certain conditions to her discretion.
  + 3rd party services would be useful for providing item owners with an automated response which is secret to the world at large
    - An Item owner would tell the Zutha network that she gives permission for a 3rd party to make return donations for her
    - In keeping with the Zutha philosophy of security through redundancy, the owner might want to provide her return donation rules to several independent 3rd parties and require them all to agree in order for a return donation to go ahead.
    - If there were multiple 3rd parties involved, any one of them could break their secrecy pact with the owner somewhat more safely.
    - If an owner is concerned that one of her return donation automation providers is selling or using their knowledge of her return donation policy, she may abandon all her providers and get a whole batch of new ones, or set for each one, one by one, to discover the cheater.
    - An owner could set a trap for an automation provider by giving it a policy that differs slightly from the policies she gives her other automation providers. The difference might contain some kind of opportunity for profit such as a policy that returns 100% of a donation under certain very specific conditions.
      * If the owner observes anyone trying to game that opportunity, she will know that the odd one out of the automation providers is the source of the leak.
      * This event will be a public spectacle because the odd one out of the automation providers will respond differently to the others, causing the Zutha network to reject the request. The owner can then declare her trap and hurt the reputation of the automation provider that was caught out.
      * if a long time passes without her obscure policy being exploited, the owner would have greater confidence that that particular automation provider is honest.
    - In most cases, owners will not need to go to lengths to ensure the secrecy of their return donation policies.
    - Some owners may choose to adopt multiple automation providers, though, to be secured against a corrupt automation provider deviating from the specified return donation policy to favour friends.
    - Such precautions are probably completely unnecessary for a sufficiently high reputation automation provider since the loss of reputation of disobeying a client would hardly be worth the ability to give someone back all of their donation just once.
    - No money can actually be taken away from the item owner by the automation provider. They are only able to deny her a donation that was rightfully hers and give it back to the donor – and just once at that, since they will quickly be sacked for doing so.
    - The only gain to an automation provider doing this is that the donor gets 100% of his donation as an asset in the owner’s item. If the donor has gambled correctly that this item is set to explode in value, he may profit somewhat more than he otherwise could have when he sells his investment.
  + The lack of legal obligation behind an owner’s declared return donation policy will be a deterrent for institutional investors since they will feel insecure about “investing” in assets when their investment might be shown up at any time as the donation it really is if the owner spontaneously decides to disregard her publicized return donation policy and keep it.
  + Innovators will realise that they can reduce their risk by making lots of small donations and waiting for confirmation of receipt of the return donation before making the next small investment.
  + I think this policy will deter big institutions anyway because they will struggle to come to terms with the alien model in their legal and risk management frameworks
  + Flexible and innovative investors will thus be able to dominate the market – dominate in the sense that they will be able to make lots of profits from the Zutha investment market while big institutional investors miss out.
  + On the one hand, the absence of big institutional investors in the market will be a lost profit opportunity for Zutha. On the other hand, I think their absence will have a significant positive effect on public perception of Zutha and the culture that permeates the active Zutha community.
  + The donation model of investment is also likely to be more attractive to philanthropic investors because they will feel that the money they are getting back is given freely because the item owner actually believes doing this will increase her revenue. If the percentage of investments kept by the investor was fixed, however, philanthropic investors would simply feel that donating through Zutha is more expensive because you have to spend a lot more to get the same amount of money to the recipient.
  + I want philanthropic investment to dominate the market so that it becomes the foundation of determining the relative value of different items.
  + If institutional investment dominated the market, the self-referential nature of the market would show itself to greater negative effect. This is because institutional investors would wield huge amounts of money and be purely profit investors so their presence would drastically increase the proportion of investment that is profit investment compared to the proportion that is philanthropic investment.
  + The greater the proportion of profit investment the greater will be the arbitrariness of the relative values of items, because profit investment is completely self-referential in that it is about predicting what others will do who are in turn trying to predict what you will do.
  + The anchor which stops the psychology of the market from being an endless self-referential loop is the activity of philanthropic investors. Profit investors must take into account their prediction of what philanthropic investors will do. They must also take into account their prediction of what other profit investors will predict philanthropic investors will do, which amounts to something very similar to predicting what philanthropic investors will do directly. The greater the power of philanthropic investors in the market, the more concretely the market will follow their lead. If philanthropic investing plays a small role in the market, though, there will be a greater amount of arbitrariness in the market since predicting what other profit investors will do in a self-referential way will play a greater role than predicting what philanthropic investors will do.
  + A major benefit of the donation model of investing might also be that it will circumvent gambling restrictions because investments will be, legally speaking, donations.
* How can the donation model of investing be extended to prediction markets?
  + The only benefit of using the donation model for prediction markets would be to get around gambling restrictions
  + There is no reason to keep out institutional investors in prediction markets since they are anchored to a concrete asset: a promise to pay a certain amount given the measurable outcome of an event at a concrete time in the future.
  + In order to make prediction markets into a form of donation (officially), the “promise” part would have to be removed
  + Reputable users could start prediction markets in which the investments are officially donations to them, but they state (without legal binding) that they will “donate” back some or all of the proceeds to the winners of the prediction market.
  + A prediction market could actually be associated with a charity or other deserving recipient of donations. The owner of the prediction market could declare that a proportion of the money “donated” as part of the prediction market game would go to charity, while the rest would be divided up among winners.
  + I think associating prediction markets with charities would also help to improve public opinion about prediction markets.
  + For example a prediction market could be used to predict the effect on American life expectancy, and the total cost (two separate markets), of various proposed Healthcare policies, with 30% of the trading volume being donated to fund cancer research.
* The highest reputation users are likely to have an identity in the real world so they will have more to lose by exploiting the privileges of their reputation on Zutha
  + The reason high reputation users will often have identities in the real world is because most of the things that will receive the largest amount of investment will be real world phenomena. A user has to be associated with a real world identity to be able to acquire that identity’s reputation.
  + If a high reputation user was to “steal” money from Zutha, for example, not only their Zutha reputation would suffer, but also their real world reputation, which will be a very public one if they have a high reputation on Zutha.
  + Another case where this phenomena will protect promises is with prediction markets where the owner of the prediction market is not legally bound to keep his promise to allocate winnings once the prediction market expires. A public figure who reneged on such a promise would have hell to pay.
* How will the donation model of investing be applied to negative investment?
  + A negative investment should be considered a donation to Zutha
  + The Zutha community will continuously be voting (through investment) on the current proportion of negative investments that should be donated back to the investor
  + It might be more challenging to get this model accepted legally as donation, since the amount you will get back is predictable. It still won’t be legally binding, though, and that might be enough, but there would be no way for Zutha’s current return donation policy to be broken.
* Document items should allow arbitrary html
  + It should be possible to construct rich text documents using an inbuilt WYSIWYG editor
  + It should also be possible to import an arbitrary page on the web into an item.
  + Items that represent web pages should cache the content of those pages.
  + The cached version of the page will be accessible on the Document tab of the item.
  + There will also be a direct link to the page itself
  + There should also be an option to show the page through a frame
  + There would need to be active policing to delete items containing malicious scripts in their html documents.
  + The cached version of a page should need to be updated manually by someone with sufficient privileges to modify the item
  + A history of changes to the Web Page item would be available through Zutha’s standard item history browser
  + Rich page content like images and flash objects should not be cached. If the cached version of the page is current enough, the links to these objects will still be correct and the objects will be pulled separately by the browser.
  + This page caching paradigm would allow someone to create a public document on Google Docs, connect a Web Page item to it, and have it cache the Google Document. The document would then be stored as native content inside the Zutha item.
  + It should be possible to convert a Web Page item into a Document item and break its connection to the web page that was the original source of the cached content. The content would then become editable within Zutha.
  + One of the benefits of this paradigm is that 3rd party html document creation tools like Google Docs can be used to build documents without creating a dependency on that specific service. If Google was to go bankrupt (god forbid), none of the Zutha content created using Google Docs would be placed in jeopardy.
* 3rd party application framework
  + 3rd parties could develop applications that run on the Zuthanet and use Zutha’s users
  + Reputation would be a useful currency for rewards in games
  + Users could be charged an entry fee for games which involves investing some zuth in the game
  + The reputation accrued by the game creator would provide him revenue for rewarding players in his games with investments that give them reputation
  + Reputation transferred in this way would have to be permanent commitments. There should be a feature site-wide for investing permanently – giving up the right to sell the investment later.
  + Apps would have to be accepted by the Zutha tech community in order to avoid security breaches
  + Apps could potentially be hosted by 3rd parties using their own resources, but it would likely be useful for some applications if they could be hosted by the whole network
  + The API for app developers would obviously be designed with the intention of preventing security vulnerabilities, but it would still be a good idea for apps to be examined in detail to check for anything the API designers didn’t think of which might pose a security threat.
  + Each host in the network would independently have a responsibility to check any apps they host for security threats.
  + It might be workable for some hosts to refuse to host certain apps. They would declare that they don’t host that app, and would not be asked to process requests relating to that app.
* Example 3rd party applications
  + Poker
    - The host of the game needs to select the cards for each player
    - The host cannot be the Zuthanet because all of its workings are public so the cards would be predictable in advance
    - The host should be a 3rd party who simply interacts with the players over the Zuthanet
    - The host selects the cards for each player and sends this information over the Zuthanet in the form of Zutha items of a type customized for this poker application.
    - The host encrypts the items representing each player’s hand in their public key
    - In this way, the secret parts of the game can be kept secret while the game is going, but all the details of the game can optionally be revealed later. This mechanism also allows Zutha to prove the authenticity of the previously secret data
    - The players make bets by posting items publically to the appropriate place in the Zutha database
    - Actually the players should send requests to the game host, who will authenticate the requests according to the game rules and publically post the authenticated actions so the other players can see them
    - The players will need a graphical user interface for the game on their clients
    - It’s becoming clear to me that the application framework will really be built on top of clients, which means there will be many different standards.
    - Different clients will boast different capabilities and be compatible with different apps, but as long as an app has been built for a client, that client can communicate via that app with anyone else that has that app installed.
    - In fact, most apps should theoretically be useable without any specialised software on the client at all, just by manually creating the items that send the right messages for that application.
    - So a person X without the graphical user interface for the poker game could discover his hand by reading the relevant items through the standard Zutha interface.
    - He could then manually craft a “Poker Bet” item and fill in the field which defines the amount he wants to bet
    - The host of the game would have a query open which will match Poker Bet items for this game so X’s Poker Bet item will be pushed to the game host as soon as one of the Zutha nodes the game host is connected to receives the Poker Bet that X has propagated across the network.
    - The game host will authenticate X’s bet and post back the confirmation item
    - The other players will receive the host’s confirmation of X’s bet and the next player will be given a turn to bet.
  + Augmented Reality
    - Augmented reality is all about visualizing data about the world around you
    - This data will simply be items in the Zutha database with geotags
    - A client running an augmented reality app would make requests for items located within a radius around it
    - The augmented reality app would calculate how to display the data appealingly to the user through the client hardware
    - The client might be HUD (Heads Up Display) glasses which can overlay images on the user’s field of view
    - The augmented reality app would interface with the HUD hardware to generate interactive annotations over the user’s field of view
    - A user might “click” on an object’s annotation in the virtual dimension (using a hand gesture perhaps) and write a review about it.
    - The augmented reality app would simply be providing a customized graphical user interface for the user to create a Review item in the Zutha database
  + It looks like the way to approach apps with Zutha is to consider the Zuthanet a globally shared database on top of which arbitrary applications can communicate
  + I can’t think of a case where it would be appropriate or necessary for Zutha nodes to host 3rd party apps
  + The Zuthanet should be a conduit for the transfer of data between applications rather than a platform for applications themselves
* I don’t like the horizontal menu bar
  + I think a drop-down style menu would work better
  + A dropdown menu can cater for more variation in the set of menu items.
  + It can also contain more items in a pleasing way.
* Item icons should default to the type ancestor with the highest icon weighting
  + Different type items should give different weightings to their icon
  + For example Investment items are also comment items, but Investment items should take their icon from the type item: Investment.
  + So the Investment type item has a greater icon weighting than the Comment type item
* Item worth bars arrangement
  + Every item should display up to 2 bars – one for Zuth and one for money
  + Each bar has two halves: the negative half, which extends from the centre of the page towards the left, and the positive half, which extends from the centre of the page towards the right.
  + A stylish marker should mark the point on the continuum representing the net worth (positive minus negative) of the item.
  + If the net worth of the item is positive the marker should be coloured the positive colour, otherwise it should be coloured the negative colour.
  + So for real money, the marker would be green and lie to the right of centre if the net worth of the item in real money terms was positive
  + For Zuth, the marker would be pink and lie to the left of centre if the net worth of the item, in terms of zuth, was negative
* As the outcome of a prediction market becomes more certain, the amount of money in the market will fall as it becomes less profitable
  + This will be especially the case in a prediction market where a proportion of investments are donated to charity
  + Consider the example of predicting American GDP for January 1st 2013.
  + As this date approaches, the spread of bets will shrink because there will be less uncertainty about the result
  + I guess the market could still retain a large quantity of money even as the spread shrinks.
  + People who had previously invested in values that are now known to have been wildly off target will have already sold their investments
  + Theoretically investors could keep shrinking the spread of their bets arbitrarily as the uncertainty in the final result falls
  + At some stage before the outcome of the prediction market is announced by the judge, the market has to be frozen
  + It might be desirable to have multiple parallel prediction markets which are set to be frozen at varying times before the outcome is announced.
  + The reason this might be useful is that it might draw out more realistic information about the relevant probabilities at the time the market is frozen.
  + If the market is not going to be frozen until very soon before the outcome is announced, then there would be an incentive to wait until just before it freezes to invest, because more information would be available then.
  + Then again, at the time when more information is available, you have to predict more exactly to be profitable.
  + Also, there is an incentive to invest early in a prediction in order to profit by selling once investment by others piles up on that prediction.
  + Predicting the correct outcome and investing in it early on is still the best strategy, since you will have the option of selling that investment for profit once others have invested in the same prediction after you, or waiting all the way until the winnings are divided up if you remain confident that you predicted correctly to the end.
  + Actually, you are likely to invest across a broad range early on, and then sell off the sections of this range that seem least promising as more information arrives.
* How would the donation part of prediction markets work?
  + Every investment could be the donation of an investment asset, of which some is donated back
    - In this model, the part of an investment that is donated to the item owner could be cashed immediately, left unsold until the market expires and winnings are allocated, or managed by the item owner
    - If donations were cashed immediately, this would be equivalent to a flat tax on all investments.
      * Let’s say the item owner keeps 30% of investments.
      * A rational investor would only invest in this market if he is confident he can make more than a 30% return (actually a 1/0.7-1 =43% return)
      * For a perfectly rational investor with a realistic sense of his own abilities, this would probably lead to a very thin market
      * I don’t think the average investor will be a perfectly rational investor with a realistic sense of his own abilities though.
      * Also, the feeling that one would be losing money to a good cause if unlucky might drive the behaviour of a lot of investors in a charity based prediction market
    - If donations were left unsold until the market expires and winnings are allocated, this would be equivalent to a flat tax of 30% on winnings, except that the dynamics of selling investments before market expiry would make things more complicated
      * A lot of the assets owned by the market owner (let’s call them static assets) would be eaten away by investors selling their assets before market expiry
      * A lot of static assets are likely to be eaten away in regions of the distribution that decrease in popularity over time
      * In the regions that become more popular – presumably the regions that are most likely to win – less selling is likely to occur because existing investment holders in those regions will anticipate further appreciation of their assets. This means that static assets in regions that grow in popularity are likely to remain close to their initial saturation level i.e. 30% in the case of a 30% donation.
      * Thus at the final winning outcome, there is likely to be a saturation of static assets of about 30%
      * 30% of winnings will thus be deducted and go to the item owner, who may be promising to relay it to a charity
      * A rational investor will thus only invest in the market if he expects at least a 30% return, since he will lose that 30% return to the charity
      * Actually investors still lose 30% of their initial investment so the return required to break even will be even more than the 43% required in the flat tax model
      * This model would make it far more attractive to make money by buying low and selling high than by predicting accurately and waiting for the market to expiry
    - If the item owner manages his own assets the impact on investors will be unpredictable
      * Investors will still have to earn at least a 43% return just to make up for the amount lost when they first invested.
  + Some proportion of the losing money could be given to a charity and the rest divided up among the winners.
    - I’m not sure how this would impact incentives and the likelihood of profitability, but it seems like an attractive option

# Thu 3-Feb-11

* Cartoon to illustrate typical Aid programs versus a more sensible approach
  + The setting is a crowd of poor people at the base of the wall
  + The top of the wall represents greater wealth
  + There are fat people with condescending, but kind expressions leaning down from the top of the wall to try to haul individual poor people up to the top. These represent well-meaning aid agencies who think they can alleviate poverty by giving money to the poor.
  + There should be a lot of grasping hands trying to pull down the people who are being pulled up by the fat people. Sometimes the fat people lose their grip and their subject falls back into the crowd, screaming.
  + There are thugs standing among the poor people and intimidating them with guns
  + Some fat people (including or maybe just Uncle Sam) are leaning down from the top of the wall to hand wads of cash to the thugs with an accompanying wink.
  + In a small clearing amongst all this is a semi-circle adjacent to the wall, lined by soldiers wearing many different uniforms and facing outward. They are letting streams of people through their ranks, but holding back a number of thugs.
  + In the middle of the semi-circle are carpenters teaching the poor people to build ladders.
  + The first few ladders have been erected against the wall and people are swarming over them.
* Unemployment insurance
  + There could be a philanthropic fund to provide unemployment insurance to those who can’t afford it
  + Those who want better quality unemployment insurance can pay a premium for it
  + An easy way to distinguish between those who can and cannot afford unemployment insurance would be to provide a relatively inferior service to anyone who does not pay for their own unemployment insurance
  + The philanthropic unemployment insurance might also cover people who are unable to get private unemployment insurance because they have such a bad record of keeping their job.
  + I think this would be a better form of job security for society in general, and probably most individuals, than minimum wages, unfair dismissal regulations etc.
* How can greater safety be encouraged without forcing people to pay more for a luxury they can’t, or don’t want to pay for?
  + Donated saw blade covers
  + Philanthropic subsidy on safer goods
* Military decision making.
  + Many overlapping secret discussion groups.
  + Each discussion group is responsible for a certain decision or category of decisions.
  + Discussion groups are like neurons; people are like synapses, each linking many neurons.
  + No one can anticipate Zutha because no one knows more than a small segment of Zutha's "thoughts" - the individual private discussions.
  + Discussion groups should include a random selection of users with a high reputation in the relevant area, as well as a selection of completely random users.
  + It would be expected that information would travel in the form of gossip as well as via direct links between discussion groups in the form of a user common to both.
  + Information that spreads this way will be effective at getting information to where it is most needed, yet it would be very difficult for an outsider to find a specific piece of information because it would be very difficult to track down one of a random group of users, and harder still to guess who they might have told.
  + There should be an expectation that discussion groups publicize their discussion as soon as they believe it no longer contains sensitive information
  + It would be difficult for discussion groups to disobey this expectation, because they would be a random selection of people who don’t know each other, and each would have a stronger bond to Zutha and its norms than to each other. Only one member of the group is required to unencrypt the discussion. To keep it secret the group would need to convince every single member – none of whom would feel any particular loyalty to the group – to keep mum.
* Try to purchase some land from an African government.
  + This will elicit less hostility both from that government and the rest of the world than Zutha simply declaring sovereignty over a piece of African soil.
  + It might help to buy a segment of land at the intersection of several countries and build a city over it, because it would be more difficult for one of the countries to claim back ownership of a corner of a city than a whole city and the other countries would likely oppose them taking the whole city, which includes their old land.
* Zutha the distributed database should have nothing to do with specifying laws. It should only be a useful but neutral tool.
  + The system of government I've been designing has libertarian leanings but it should be possible for a society to use Zutha as a tool for building any kind of social system.
  + I see what I'm planning to start in Africa as a kind of "Startup Country".
  + Hopefully there will be many competing startup countries, all of which will learn from each other.
  + I think it will be important to encourage groups with many diverse paradigms to start up their own societies using Zutha as a supporting tool
  + For example there are many people with similar paradigms to mum and Neil who might want to try to start their own communities using ideas from movements like Transition Town.
  + I think it likely that such groups would find sooner or later that what they are really looking for is something along the lines of what I’m trying to create. I think the association to Libertarianism that my proposed society will likely acquire will put off many people with left wing leanings for a while though.
  + A left-wing leaning society might use Zutha as a tool for collective government which imposes taxes on the rich to give to the poor in an attempt to increase equality. They could use Zutha as a voting tool to impose all sorts of restrictions on businesses such as minimum wage laws and unfair dismissal regulations.
  + I think it likely that Zutha could be used quite effectively as a tool to create an egalitarian society – effectively, that is, compared to a republic, because it would likely produce better decisions more transparently, which better match the aggregated desires that emerge from the collective.
  + I also think that, with competition from more libertarian societies like the one I plan to build in Africa, such an egalitarian society is likely to see some of the flaws in their paradigm exposed – such as that the best generators of wealth will leave.
  + It is imaginable that many people might prefer such an egalitarian society, even if it is much poorer than a libertarian one.
  + There is a lot that could be done by an inspired group with free reign to design their own society from scratch, to facilitate a culture with greater emphasis on fulfilment in life rather than material wealth. I think this is an area where a Zutha society started by people with left-wing leanings might have something significant to contribute to a libertarian Zutha society. This is one of the reasons why it will be important to encourage many independent Zutha communities to be started by people with widely divergent paradigms. They will each do some things well and some things poorly, and they can learn from each other.
* Ownership items should contribute to their associated owner’s reputation
  + Example: a philanthropist pays for an audiobook to be created of a popular book.
    - The ownership items for the audiobook might be the Philanthropist, the Voice Actor and the Author
    - Zutha users invest in each of these Ownership items according to how much they believe each contributed to the value of the audiobook
    - Each of the owners gains reputation from their Ownership item as well as their share of the reputation earned by the Audiobook itself
  + I think it should be possible for an owner to invest in their own Ownership item
  + This would be applicable where someone adds a book to the Zutha database which they co-authored.
    - They would want to be able to assign relative ownership of the Book to themselves and their co-author right from the start.
    - Others could then potentially come along and give their opinion about the relative contributions of the co-authors.
    - It might also be that investment in the ownership items is restricted to a select group of people – in this case, perhaps just the two authors
  + If it were possible to invest in an item you owned, you could cyclically invest in your own reputation without contributing anything of value
  + It is already possible to do this by simply investing between two or more users
  + I don’t think this is actually a major problem
  + The amount of Zuth you can earn from your reputation should be very small – much smaller than the amount of zuth you could earn by putting it to work in well-chosen investments.
  + If you invest a huge amount in one of your own items and others come along and disagree with the value assigned to the item, they will invest negatively in it
  + It would be very counterproductive for your reputation to toot your own horn extensively and then be brutally rebuffed by the rest of the Zutha community.
    - Much of the zuth you invested in your own reputation would be wasted because it would be cancelled out by the negative investments of others
    - There are likely to be a lot of people who will condemn extensive investment in one’s own items and will punish this behaviour severely
    - Budding exploiters might find that they ultimately end up going backwards in reputation by investing too much in items they own that clearly don’t deserve the worth you assign them.
  + If you invest in your own item and others agree with your assessment of your own work, then great – you have simply successfully advertised your creation of value.
  + High reputation users should have the privilege of advertising the value of new things they create by investing in them. Their high reputation is an indication of their credibility
* It would be useful if some domains could be semi-restricted so that the sider community has some influence, but most influence is held by existing reputation holders in that domain
  + Consider the field of economics
  + There are a lot of people who are likely to have strong opinions about facts, decisions and analysis related to the field of economics
  + The consensus that emerges from user voting on Zutha might provide more useful and correct information on such items if the majority of influence was held by people with existing high reputation in the field of economics
  + It would still be desirable for users outside the existing economics reputation domain to contribute their opinions about which users have a lot to offer in the field of economics
  + Consider a discussion about whether raising the minimum wage reduces employment
  + There could be a Question item labelled “in general, does raising the minimum wage reduce employment”, with two subitems: Yes and No
  + The two subitems could be restricted to users with reputation in the field of economics
  + The investments made by users with economics reputation would include a justification for that investment in the form of a comment
  + It would make sense for the investment comments to be in the reputation domain of economics, but be open for anyone to invest in
  + Users who don’t yet have any economics reputation would also be allowed to write comments under the restricted Question item which would be in the economics reputation domain. If they received investment in that comment, they would acquire some economics reputation.
  + There’s no reason why someone else couldn’t create the same Question item but make it open to any user. It would provide useful information to compare the results of the restricted versus the unrestricted Question.
  + It should also be possible to define more complex restrictions on items, such as that any user can invest up to 100 zuth in the item, but users with economics reputation can invest up to 10% of their economics reputation on top of the flat 100.
* Does a Poker game need a host?
  + Someone needs to be the “dealer”
  + The dealer cannot play, since he will be able to choose the players’ cards
  + A dealer will have to stake his reputation on the promise that he is dealing cards randomly
  + A dealer need not be the creator of the game, though
* Distributed MMORPG
  + There would be a number of challenges to running an MMORPG on a distributed platform like Zutha
  + I think Zutha’s data model would be pretty good at getting data to the right places efficiently, though I don’t know whether my current model would be capable of achieving gaming-level response times
  + One of the challenges would be who is responsible for the secret information involved in an MMORPG?
  + It might work to assign blocks of data to many different small groups of nodes
  + Someone who controls a large number of nodes might be able to attain responsibility for a few different chunks of data, but no one would have all the data, or even a large proportion of it
  + Only nodes with very high reputation would be assigned the most secret information. If they were found to be leaking this information to players, they would have a lot to lose.
  + One of the ways data could be broken up would be by virtual space. Blocks of land in the virtual game world would “belong” to different groups of nodes
  + A particular group of nodes might be responsible for calculating the physics for actions that occur in a particular city in the virtual world
  + Another group of nodes might be responsible for controlling the weather for a particular area
    - Weather effects should have an element of randomness
    - The group of nodes could use a selection algorithm to randomly assign one of their number to generate a new random number, which they would then all process deterministically to determine the next weather event.
  + Weather effects generated in one area would need to interact with weather events generated in another area
    - The group of nodes responsible for the weather in one area would have to communicate with the group responsible for the adjacent area
    - This might take the form of group X telling group Y that they are sending a cloud their way. Group Y should take in the data about the cloud formation from X and calculate its impact on Y’s area.
  + It wouldn’t necessarily be necessary for regions of control to take the form of a grid with a specific set of owners for each square in the grid
    - Each weather node could be given control of a sphere of virtual space
    - Every point in space would lie within multiple spheres, each controlled by a different node
    - As a cloud moves through virtual space it would be regularly entering new spheres
    - As the cloud moves into a new sphere owned by X, the nodes currently controlling the cloud would alert X of the approach of the cloud and invite it in to their group dedicated to managing that cloud
    - This would mean X has to make the same calculations as the other nodes whose spheres overlap the current location of the cloud
    - X would not be able to make any independent choices about the behaviour of the cloud because X’s output regarding the cloud would have to match the output of the other nodes currently managing the cloud
    - Random events could be generated by unpredictably selected nodes generating a random number which is then processed deterministically by the relevant nodes
    - For example, there might be a rule that there is a 1% probability every hour that a new cloud will form in any given cubic meter of atmosphere.
      * Let’s examine how this rule is enforced for a particular cubic meter of space called S
      * When the game first starts, the nodes whose regions of control overlap S select one of their number using an unpredictable but deterministic selection algorithm, to generate a random seed
      * Every minute, say, all the nodes generate the next pseudorandom number in the sequence begun by that seed
      * They each use this number in an algorithm to determine whether a cloud will be formed in S at the start of the next minute
      * Perhaps every 10 minutes, they use the selection algorithm to select another of their number to generate another seed
      * In this way, none of the nodes will be able to predict what will happen in the next 10 minute block, but they will all know exactly what will happen for the whole of the current 10 minute block
  + Node groups should be able to easily incorporate a new node into their number if one of their number fails
  + It would actually be useful in many situations for nodes to continuously be evicted from a group and replaced by a new node
    - This would prevent control over a particular task from remaining in any one group of nodes’ hands for too long
    - If the nodes responsible for a particular task was static, then an attacker with a lot of resources could go about taking control of each of those nodes until he eventually has control over the task they were managing
    - If tasks were continuously changing hands, an attacker would not be able to acquire control of all the nodes in the group before the group changed
  + Different node groups could be assigned to different players to manage their stats
    - This would only be necessary for player stats that need to be kept secret from the public, and perhaps even the player himself
    - For example, a player might acquire a hidden disease which causes random bouts of dizziness.
    - The player might have acquired this disease from an environment he passed through.
      * The controllers of that environment would have generated a random event which gave the player the disease.
      * They would have told the Player’s controllers that the player has acquired this disease.
    - The Player controllers will now be responsible for broadcasting the effects of this disease
    - At random times, as defined by the properties of the disease, the Player controllers will tell the relevant Nodes and the Player’s client, that the Player is experiencing a bout of dizziness
    - The implications of the dizziness might be that the Player cannot move or cast spells
    - If the Player’s client tries to give a command to move the Player, the Nodes responsible for object positions in this area will not accept the command because they have been alerted by the Player’s controller nodes that the Player is not currently allowed to move.
  + Any given Node is likely to be assigned a large number of random tasks
    - With every minute task each being assigned to many Nodes, it is unlikely there would be enough Nodes in the network to do one task each
    - Someone who owns a number of Nodes would be able to know a random selection of secret facts about the game world, but would have no control over which facts he knows.
    - It would thus be exceedingly difficult to acquire targeted secret knowledge in order to cheat
    - Let’s say you want to find out what items an enemy player has in a PVP context
      * You would be able to know which nodes know this information, but they would be completely random nodes residing anywhere in the world with no identity except their uninformative ID number
      * It might even be possible to conceal which nodes know a given piece of information to nodes and clients that do not need to know
      * I don’t think this would be necessary, though, because it would be difficult enough as it is to extract secret information from a Node known only by their ID number. Not even their location would be known except by their direct neighbours because of the location privacy of the Zutha network.
  + What would be the effects on game latency of the distributed nature of the game?
    - Let’s say player A is fighting player B
    - Nodes X and Y are responsible for managing player A
    - X is close to A while Y is close to B
    - Player B casts a spell on A which prevents A from casting
    - A moment later, A tries to cast a spell
    - A will ask both X and Y for permission to cast the spell
    - Let’s assume X replies before having received a message from B’s controllers notifying it that A cannot currently cast
    - Y, being closer to B, will receive B’s earlier message before A’s message
    - A will thus receive an affirmative from X and then shortly after, a rebuttal from Y
    - A will know that Y’s message is more up to date because Y’s message will include relevant information missing from X’s response
    - A’s client will begin to allow the spell and then block it soon after
    - This is just like the behaviour caused by high latency in existing MMORPGs
    - How will the latency of the distributed system compare to that of a centrally controlled MMORPG?
      * centrally controlled system
        + let’s assume the server, S, is right next to B
        + B’s message would be sent to the server and arrive first
        + B would receive instant feedback that his block spell was successful
        + A would send a spell cast message to the server before receiving the message from the server that it has been blocked from casting spells
        + A’s client would show the casting of the spell until it receives the message from the server that it is not allowed to cast spells
        + The maximum time that could elapse between A trying to cast and discovering it can’t would be if A cast a fraction of a second after B blocked
        + Actually, if the server didn’t trust clients to assign correct timestamps to their messages, A could send a message before B, but B’s message would be accepted as the earlier one by the server
        + The latency for A would be 2\*AS ms, where AS is the time taken for a message to get from A to S in ms
        + A could cast a spell AS ms before B blocks.
        + S would accept B’s message just before A’s arrives
        + S would then send a message back to A in time AS
        + If S was right between A and B, both would have the same latency of 2\*AS = 2\*BS
        + In this case, A would cast a spell right after B’s client sends the message to block it
        + S would receive both messages AS=BS ms later
        + A and B would each discover the outcome another AS=BS ms later
      * Distributed system
        + Let U and V be B’s controllers
        + U is close to A and V is close to B
        + B sends a block message to U and V
        + One of U and V will be selected as the Master Node to timestamp the message. Let’s say V is selected.
        + V will send its timestamped message on to U as well as to X and Y
        + Meanwhile, A sends his spell cast message to X, who we’ll assume timestamps it
        + A will receive an acceptance response from X, and then, sometime later a rejection response from X after it receives the block message from V
        + A waits AX ms before its message is timestamped
        + Just before A’s message is timestamped, B’s message is timestamped by V. We’ll use the approximation that both events happen at practically the same time
        + A waits another XV+AX ms to be notified of B’s block
        + A’s latency is 2\*AX+XV
        + Of course, if either X or V are contracted by the other nodes with the same responsibility, A’s latency will be longer, but this will be the rare exception.
    - The messages sent in a game context would not provide enough flexibility for crafting a message to manipulate the selection algorithm
      * Therefore there is no need to be worried about the message sender having control over the selected Master Node – the one that gives a message its timestamp and ID
      * The Master Node could be trusted to assign a timestamp without waiting to receive the go-ahead from one of the request handlers
  + How would A and B’s conflict play out if their requests were managed by randomly assigned Master Nodes who don’t have to wait for the go-ahead from a request-handler?
    - B’s message could be sent to a randomly selected Master Node (let’s call it V) from among all superior nodes in the network
    - V would timestamp B’s message and immediately send it to A as well as every other Node in the network
    - A would send a message to a selected Master Node, X
    - X would send a reply back to A with the timestamp and ID assigned to A’s request
    - X would also send A’s message out to the whole network
    - Let’s assume X receives A’s message a moment after V receives B’s message
    - A has to wait AX ms before X receives its message
    - A then has to wait until it receives both V’s relay of B’s message and X’s reply before it knows whether its request was successful
      * This is because it has to find out which message was officially sent first
      * X’s reply tells A what the official timestamp of A’s message is
    - This means A’s latency is AX+max(AX,AV)
    - B’s latency would be BV+max(BV,BX)
    - If X and V are randomly assigned Master Nodes that could be anywhere in the world, then their average distance from A and B will be one quarter the circumference of the earth
      * This is because the longest possible distance is half the circumference of the earth, the shortest is zero, and the distribution in between is symmetric.
    - So in the worst case scenario for A (assuming no disputes between Nodes) the average distance that must be traversed by messages for A to know whether its message has been successful is half the circumference of the earth
    - How does this compare with a centralized server model?
      * The worst case scenario for A is that the server is on the other side of the world (1/2 earth-circumference)
      * A’s latency would be 2\*AS = 1 earth-circumference
      * So the distributed model seems better at handling a global MMORPG
      * The average scenario for the centralized model is that A is ¼ earth-circumference from S
      * In this case A’s latency is ½ earth-circumference
      * A given user in the centralized model will have the same latency for every request, but different users will have different constant latencies
      * A given user in the distributed model will have varying latency, but every user will have the same average latency
      * The average user’s constant latency in the centralized model will be the same as every user’s average experience in the distributed model
      * So the distributed model will be much more fair across users, which is a huge plus
      * The average latency of requests is what is important in the distributed model, because for the majority of requests latency won’t be noticed (since there will be no conflicts)
      * When there is a conflict – as in the scenario between A and B – it is statistically likely that the latency will be the average latency. It is unlikely that a single request will have both poor latency and experience a conflict (from one user’s perspective)
      * The distributed model need not cover the whole globe. Only Nodes from the US could be used as hosts, in which case the same comparison between decentralized and centralized model would apply, except with the measurements based on the size of the US instead of the size of the earth.
      * So the distributed model seems, on the whole, much better
    - I’ve only measured A’s latency assuming nothing goes wrong
    - Once A has received a reply from X and the relayed message from V, A can deduce the correct action to display for its user
    - However, X and V have not yet communicated
    - X will broadcast A’s timestamped spell cast request to the whole network at the same time it replies to A
    - X will initially calculate that A’s request succeeds
    - V will broadcast B’s timestamped block request at the same time that it replies to B and sends the message to A
    - Nodes somewhere between X and V will be the first to be aware of both A’s and B’s message
    - As the Nodes close to X receive the message from B via V, they will update their databases to record A’s spell cast attempt as a failure
    - A acts just like any of these clients
    - A would send its initial request to multiple request handlers, not just X, the Master Node
    - Y is one of the other handlers of A’s request
    - Y would also reply to A’s request to indicate whether it is acceptable
    - Y, being close to V, would likely already know about B’s request by the time its response gets to A, so Y’s response will indicate failure
    - A might receive either Y’s request or X’s response first
    - A will update its state as new information comes in
    - If A already knew that its request had failed by the time Y’s response arrived, Y’s response would just be processed in the background by A, but wouldn’t affect the user’s gameplay
    - If Y’s response differed from the majority of other responses, then A would alert the network that Y is responding anomalously and Y would probably be punished
* Nodes need a form of reputation which is independent of their processing resources
  + There are tasks which need to be entrusted to particular nodes
  + It will be important for the community to be able to punish nodes for wrongdoing that cannot be detected automatically
  + For example, the community should be able to punish a node for leaking secrets it had been responsible for
  + It would also be important for the community to be able to reduce the reputation (and accompanying power) of nodes they suspect of being controlled by entities they mistrust
  + For example, it might be suspected that a node is one of a large set of nodes suspected of being controlled by the CIA. The Zutha community would not want the CIA (or any other single entity) to have control of a large proportion of powerful nodes on the Zutha network, so they would downgrade (by negative investment) nodes they suspect of being controlled by the CIA.
  + A node’s reputation would be formed by investment by users in items they are assigned ownership of, as well as the automated reputation system for assigning greater traffic to nodes which are more reliable at responding to requests.
* Nodes could elect to only manage certain kinds of traffic and store certain kinds of data
  + As the number of ways Zutha is used increases, the total amount of traffic could become too great for smaller nodes to cope with
  + It would still be useful if nodes with less resources could be relied upon to always have data about known categories of items though
  + For example a Node might be known to keep track of Event items, but nothing else
    - Another node might keep track of semantic items, but not Event items, Comments, or Investments
    - Another node might handle MMORPG data, but not standard Zutha semantic items
  + There would be multiple indexes of the nodes that hold each kind of data
  + It would probably be a good idea if every single node in the network was required to store every index
    - This way, when a client wants to connect to the network, the first Node the client encounters will be able to tell it where to find any kind of data it wants
  + When a new item is created, it would only be sent out to the Nodes assigned to manage that kind of item
    - The Event item associated with the creation of the item would only be sent out to Nodes assigned to managing Events items
* The system for sending messages directly to another item could replace the probabilistic system I had devised for propagating items to every Node
  + The probabilistic propagation mechanism
    - Nodes send messages with higher probability to neighbours which, in the past, had most commonly not already received the message
    - This system would be extendible to the scenario where a message only needs to reach nodes that handle messages of that type
    - It would not work very well in this scenario, though, because there might be Nodes that should receive a message which are surrounded by nodes that don’t want to receive that kind of message
  + An alternative would be to propagate messages across the network by explicitly sending the message to every node in the network that wants to receive that message
* Whenever a Node sends a message to a neighbour it should measure the time taken before the neighbour replies to confirm it received the message
  + This latency information should contribute to a neighbour’s connectivity weighting
* Nodes connected directly to the internet would be neighbours of every other Node connected to the internet
  + Nodes that are neighbours over the internet would be a different kind of neighbour to Nodes that are connected directly by capable or wireless
  + Direct neighbours would know which of their neighbours are currently connected at any time
  + Indirect neighbours (such as those connected by the internet) would not know immediately when their indirect neighbour disconnects
* Connectivity weighting would be very important for managing communications over the internet
  + Two Nodes A and B may be connected directly as well as being connected to the internet,
  + Let’s say nodes A and B are connected to the internet via different ISPs
  + When A sends a message to B via the internet, the delay before B replies will be large so A should decrease the connectivity weighting of that connection
  + When A sends a message to B via their direct connection, the delay before B replies will be very short, so A should increase the connectivity weighting of that connection
  + In the future, when A needs to send a message to B, A should use the direct connection far more often because it has a much higher connectivity weighting
  + The usefulness of this latency metric extends to far more complex network scenarios
    - A might be connected directly to C which is connected directly to D which is connected directly to B
    - A and B are connected to the internet
    - A’s connectivity weighting with respect to B of neighbour C should be higher than A’s connectivity weighting with respect to B of neighbour B
* I’m starting to wonder whether number of hops is really a useful metric for connectivity weighting at all
  + Perhaps latency is the only metric required
* How will nodes that are connected to the internet maintain location privacy?
  + Nodes that act as Hosts will need to report their IP address to be findable by other Nodes
  + An internet connected client can still maintain location privacy by sending messages to any internet connected Host they trust
  + A Client could also request any number of remailers, either using the Zutha network for remailing or any 3rd party service
  + A Host could share its IP address only with a select set of trusted Nodes, thus effectively making only those nodes its neighbours
  + Nodes that have no need for location privacy would report their IP address publically
  + Public Nodes would significantly increase the connectivity between nodes that are physically near them and the internet
  + Case study 1
    - Node P wants to keep its location secret
    - P is not connected to the internet
    - P has a neighbour A which is connected to the internet
    - P wants to communicate with C
    - C has a neighbour B which is connected to the internet
    - P’s message will go via A and B to C
    - B will know that A is the closest of its neighbours (in latency terms) to P
    - B can guess that P is probably close to A, but cannot be sure, because the link between A and B may simply be an efficient step to take across the internet in a path that may contain several steps across the internet and several direct links between nodes
    - Someone who controls many hosts connected to the internet may discover that most of them usually send messages intended for P to A
* People vote on advertising.
  + Companies lose reputation if people don't like their ads, even if the ads work on some people and increase their profits.
* I think the Zuthanet will be faster than the internet
  + Links between Zutha nodes will be found by identifying the path with least latency
  + Ultimately this will lead to very direct connections between Nodes
  + On the internet, however, messages can zigzag across the planet as they pass up the chain of name servers
  + The reason the Zuthanet can work like this is because there are few enough nodes that every node can have a database of the best neighbour to send requests to in order to reach any given node
  + It is not actually required that every node knows every other node, but the greater of prior knowledge nodes have about each other, the faster the network will be
  + Most clients that participate in the network will not be known to many nodes in the network, but once clients initiate a connection with a host, the host will know how to get a message back in an efficient way
  + A client’s message can reach a host efficiently because most nodes in the network will know the most efficient way to reach a host (in the form of the best neighbour to send messages address to that host)
* Are request handlers necessary?
  + There should be a set of hosts selected to respond to a client’s request so that the client always receives a sufficient number of responses to be sure of their validity, but not too many responses to be wasteful
  + The hosts that respond to the client do not necessarily have to be the ones the client sends the request to
  + Actually it would be useful if the request handlers had received a direct connection from the client, because then there would definitely be a fast connection formed between them
  + Another purpose of request handlers is to keep the Master Node honest
  + The Master Node could theoretically report a timestamp for a message that is earlier than when the Master Node actually received the message
  + There would be no incentive to do this unless someone was able to manipulate the selection algorithm to pick one of their nodes as the Master Node
  + It would be better if the request handler were not required to keep the Master Node honest as this would increase the latency of requests
* Is the Master Node necessary?
  + The Master Node is a single point of failure, even if only for a single message
  + The Master Node can hold up the whole lifecycle of a message if it is slow to respond or isolated (perhaps temporarily) on the network
  + It would be better if the timestamp could be determined without having to wait for a predefined Master Node
  + The timestamp of the second request handler to receive the message could be used
  + It would not work for it to be the first request handler because then one of the request handlers could report an arbitrarily early timestamp and be considered the first the receive the message and thus the timestamp decider
  + The client would send a request with a GUID identifier to all selected request handlers
  + All the request handlers would immediately respond to the client
  + At this point, the request handler might also send out their preliminary response to other nodes in the network who need to know about it quickly
    - For example, if the client that sent the message is currently fighting another player in a distributed MMORPG, the other player should be notified immediately of the client’s actions
  + Once the client had received the second response, it would know that the timestamp of its request would be equal or earlier than the timestamp on the second response – likely equal.
  + One of the request handlers would be assigned by the selection algorithm the task of giving any created items (such as Event items) their IDs
    - It doesn’t seem appropriate to call this request handler the Master Node anymore, since it is not the central node on which the whole process waits anymore
    - This node should simply be called the Naming Node or Namer
  + The request handlers would all forward their responses to all the other request handlers
    - This would require N2  messages to be sent among request handlers
    - If the number of request handlers was something in the range of about 5 to 12 as I anticipate, then this would not be too onerous
  + Once the request handlers had received the responses of all other request handlers, they would all know which response had the second earliest timestamp
  + The Namer would also have sent all of them the IDs of the items to be created
    - The Namer could theoretically send half the request handlers one ID and the other half a different ID
    - If this occurred, nodes that discover the discrepancy would know that the Namer is responsible so it would be severely punished
    - Two Nodes that had different short IDs for the same request (identified by its GUID) would be able to show each other their differing signed messages from the Namer which would prove its crime
  + The request handlers should all send their complete items to each other so they can all confirm that they ended up with the same thing
  + At the same time the request handlers would propagate the completed item across the network
  + This would be done by every request handler sending out a message addressed to all the nodes that care about this kind of item
  + If a node receives a duplicate message addressed to the same node, it will not send the message again
  + A message will contain many addresses initially.
  + Nodes will deduct their own IDs (if present) from the address list of the message and then pass it on, probably breaking it up into several messages, each with a subset of the addresses, and sending each to a different neighbour
  + Consider a target node X between two request handlers A and B
    - X receives the messages from A and B at the same time
    - The message X receives from A will probably include some addresses for nodes between X and B
    - Damn! There is no way for X to know that the nodes between X and B will already have received the message from B
    - But if X has received the message from both A and B, X will know that it is the meeting place of two propagation circles
    - There could be a rule that nodes don’t send propagation messages to nodes they have received the same propagation message from, even if the incoming and outgoing address lists differ
    - What if there is a node C than will be reached through X from both A and B?
    - X will simply send the message once to C. This does not violate the rule of not sending a message back to a neighbour from which you received the message
  + What if B lies between A and X
    - When B receives the message from A, it will include addresses of Nodes that lie between B and X
    - B will know that the nodes between B and X have already received the message and will not propagate A’s message onwards
  + The propagation could be visualized as webs expanding outward from each of the request handlers
    - When two webs meet, they stop
    - This will reduce redundancy because every node will only receive the message from one request handler
    - One web might leave a patch empty. No other web would ever reach this patch because they would be stopped by the outer edge of the web with the patch in it.
    - As long as sufficient redundancy is delivered within a web, there is no reason for them to overlap
    - Every node could send the message out to every neighbour they have not yet received the message from
    - The message would not have to have a changing address list as it propagated; it would just be addressed to a category of users
    - Actually nodes should not send the message to all their neighbours – only neighbours which are intermediaries to getting a message to one of the nodes on the address list
    - I’m thinking it won’t actually be possible to stop every single node (or at least the vast majority of them )receiving the message with this method
      * Consider a patch of nodes which aren’t on the message’s address list
      * A node on one side of the patch will send many messages through the patch, each targeted towards nodes somewhere on the other side of the patch
      * Practically every single node will be on the shortest path between one of its neighbours and any other node in the network
    - Perhaps I just have to live with the fact that every node will have to receive every single request
    - This will be completely overwhelming for some nodes
    - Consider a cloud of inferior nodes attached only to a single internet connected node
      * This cloud of nodes is not a bridge between any two superior nodes, so they would not have to deal with any requests except those targeted to one of their number
    - There are likely to be many such clouds of minor nodes that don’t form part of a major communication channel
    - I’ve been thinking about the network as though it is flat – perhaps because of the expanding web metaphor.
    - Actually, there will be major communication channels and many insignificant nodes hanging off the main channels
    - The fastest route from one side of a cloud of inferior nodes to the other is going to be around it via a high speed communications highway
    - Thus many less powerful nodes will be able to avoid being flooded by all the messages they don’t want to receive
  + If one of the request handlers propagated a different message than they sent to their fellow request handlers, the nodes in their web would never receive the correct message
    - The message could include a signature from every request handler
  + The vast majority of nodes would receive a propagated message with the expanding webs model
  + If any nodes failed to receive a message they would be such a minority that it wouldn’t matter
  + Nodes would not be punished for not having a piece of information yet, only for reporting incorrectly when they have received all the relevant information
  + Actually it might be desirable to punish nodes if they often don’t have the latest information as this would indicate that they are not well connected to the network and thus should not be trusted as much to reliably respond quickly and correctly
* Users rather than Clients should be the ones to increase Nodes’ reputations when they make requests
  + Users should receive a salary of normal zuth, as well as a separate currency which is only used for praising Nodes for responses
  + Higher reputation users would provide Nodes with greater rewards for their response, so Nodes would have an incentive to prioritize responses to high level users
  + Users should also be the ones that gain reputation for reporting an inconsistency between nodes. This would provide a proper incentive to do so, whereas praising clients would not have really achieved anything since no one would care how much reputation their client has – in fact a high reputation client would be a disadvantage in the old system because it would mean the client would be asked to do more work
  + Users would have an incentive to praise the nodes that respond to them quickly as this will raise the amount of traffic handled by those nodes and thus the responsiveness of the network to that user
  + Users could praise nodes that respond in under a threshold response time and punish them for taking longer than the threshold time to respond
  + They could praise and punish in proportion to the degree the host is under or over the threshold response time
  + It would be possible for users to bully or unfairly praise a particular node by incorrectly reporting their response times
  + Given that the nodes they must make requests to are chosen randomly, they would not be able to have much effect on any one node
  + There should be a limit on the amount of praise or criticism a node can receive for any one request, otherwise a single user could have a large effect
  + Basically, once the response time is greater than a certain amount, the request would be considered failed, and the host would receive a fixed maximum punishment – the punishment for failing to respond in time
  + The minimum response time possible is zero, so that places a limit on the maximum amount of praise a host can receive for a single response
* What is the latency of a request?
  + X is the client
  + N is the Namer node
  + A and B are the request handlers most distant from each other
  + A is also the request handler closest to X and B is the most distant from X
  + C is the node that receives the message second and so is selected to assign the timestamp
  + X sends a message to all request handlers
  + All request handlers respond to X
  + So X receives a preliminary response in time 2\*min(XA,XB)=2\*XA
  + X can be pretty confident of the timestamp of his message (the most important information – for gaming at least) once the second preliminary message has returned, which will be a very short time after the first
  + At the same time as they respond to X, the request handlers all send messages to each other
    - All request handlers will have received responses from all others once B, the most distant node, has forwarded its message to A, the most distant node from B
    - This will take XB+BA time from the moment X sends its message
    - If one of the request handlers fails to respond, the other request handlers have to wait a timeout period to abandon that request handler
    - Some request handler T might abandon a comrade Q, while other request handlers, including U, may have received a response from Q in time
    - T might get the second message from U - a completed item - before T receives a response from Q
    - T could then accept U’s item, which would be accompanied by the signed messages of all the other request handlers including Q
    - T would add its signature to the completed item and distribute it to all the other request handlers, including back to U
    - I think it unlikely that Q’s message would get to U and U’s message to T before Q’s message gets to T
    - I guess there could be a major network block between Q and T which is circumvented by going via U
  + After this is done all request handlers know the timestamp and ID of the Event item
  + At this point all the request handlers could send their completed Event item back to X (and other interested parties)
    - X would receive the first confirmation response after time XB+BA+AX and the last after time XB+BA+BX
    - If X, A and B were all in a line (which is a very rough approximation of the average arrangement I think), then X would receive a complete response after its message had travelled distance 2XB
    - B cannot be more than ½ earth-circumference away, so the maximum latency (assuming no conflicts between request handlers) is the time taken for a message to traverse the circumference of the earth
  + The request handlers then all send messages to one another again, this time so that all will have signed final messages from all the others
  + This will take time AB for a total time so far of XB+2AB
  + Things could be sped up here by only requiring the request handlers to have signatures from half of the others before they propagate their message
  + The message is now propagated to the whole network
  + The first message will be propagated to the network before the last of the request handlers has received responses from all other request handlers
  + An arbitrary node in the network, Y, will receive X’s message after a maximum time of XB+2AB+min(YA,YB)
  + XB and AB could both be up to ½ earth-circumference (actually a bit more because paths won’t be straight), whereas min(YA,YB) will likely be small, so to a very rough approximation, the whole network will know about X’s request after a message has travelled approximately 1.5\*earth-circumference
  + This means a theoretical minimum latency in the worst case scenario of 200ms for the entire global network knowing about X’s request
  + That’s pretty good
* Should View requests be saved as items?
  + If users are able to praise or punish hosts on their response times for every request, then information needs to be distributed to the entire network for every View request
  + Only the most powerful nodes need opt in to the task of storing View requests though
  + The same nodes would also have to manage Node reputation since every View request would affect some Node’s connectivity reputation

# Fri 4-Feb-11

* Arrow fired by player A at invisible enemy B.
  + Temporary controller group, G, for managing B's temporarily secret location.
  + G announces its relevance to A's actions so A includes G as a priority listeners to its messages.
* Every node is owned by a user.
  + Some node tasks (like handling secret data in a game) requires a minimum owner reputation.
  + This would stop a single entity getting control of a large number of powerful nodes (in terms of their trustedness) without it being obvious, because to conceal their ownership of all their high powered nodes they would need many high powered users which would be very hard to get.
* Request handlers prove authenticity of item by including signed responses of other request handlers.
  + They then don't need to swap data a second time.
  + All request handlers select an ID.
  + The ID of the response with the 3rd smallest timestamp is selected. The unused ids are available for use again.
  + Request handlers send to each other a signed package of the hash of the client's initial request as well as a timestamp and candidate ID.
  + The request handlers could immediately distribute the item as soon as they receive the first response from every RH.
  + They would include the client's initial signed message and the signed packages from every RH.
  + Every node would then calculate the timestamp and ID from this data.
  + If a RH didn't receive a message from all the others, it would have to send out a message to all requesting confirmation of their selected timestamp and ID. This request would include all the signed packages that RH had received.
  + When an RH receives such a message it would send all its known signed packages to every other RH unless it has already done so.
  + If an RH has a signed package from every other RH after this step it can deploy to the network.
  + If the only RHs missing were the same ones that did not send a second message, then those can safely be excluded.
  + If an RH does not have a signed package from the majority of other RHs at this point it reports failure.
  + It might be possible that the majority did successfully complete the request and this node was part of an isolated minority.
  + If an RH is not included in the second responses of a majority of the RHs, then it will be excluded. This is the general rule.
  + If an RH is excluded, the second responses need to be deployed to the network as well, to prove it
  + This strategy would incur greater costs if communication between any of the RHs failed, but would be faster for almost every case.
  + Actually how does this method compare to the old one when we consider the failure scenarios as well?
    - In the old strategy, every RH sends out a second message as soon as it has received a message from all the others
    - The second message is simply the completed item without any baggage from the other nodes
    - What does an RH do if it does not receive a message from all the others on either the first or second round of messages?
    - If a response was missing on the first round, something similar to what is described above would be required: each RH would have to send all the 1st responses it had received to all the other RHs
    - Even if a node receives responses from all others on the 1st round, it should not assume all other RHs received responses from all RHs. Thus it should send the signed packages along with its 2nd message anyway.
    - It looks like the old method required practically every part of the new method anyway. The only difference with the old method is an extra round of messages between RHs so that the propagated message can be a completed item with a simple set of signed hashes from each RH
    - And if there is a failure when using the old method, all the baggage needs to be sent along to all nodes anyway to prove that the excluded nodes have been excluded legitimately.
  + The new method looks far superior
    - Its only disadvantage is slightly more processing for every node in the network for a unanimous message (one for which no RH was excluded).
    - The extra processing is insignificant, though. It is far more important to reduce network latency than processing costs.
* What is the impact of the new propagation method on latency?
* Let’s examine a scenario in an MMORPG in which character A fires an arrow at an invisible character B, but B moves out of the way a moment beforehand
  + Assume that if B is hit he will become visible to A
  + A’s client must not think it has hit B if it hasn’t because that would make B visible to A (if only temporarily) when he shouldn’t be
  + Let G be A’s group of request handlers for the action of firing the arrow
  + Let H be B’s group of request handlers for moving
  + Let T be the control group assigned to manage B’s position
  + Actually let T be responsible for managing the positions of all objects in the region around A and B which need to be kept secret from anyone
    - This would include items lying on the ground that A and B can see, but people on the other side of the game world do not know about
  + T will be responsible for reporting collisions between hidden objects and public objects (like trees), as well as between two hidden objects
  + Actually why should even trees and perhaps even cliffs be public items
    - The state of a tree or cliff in the past will be known to all characters who have been to the area, but a character who has never been there should no know every detail of the terrain – or even any of the terrain
  + Why are G and H required?
    - The purpose of randomly assigned request handlers was so that a resourceful entity could not obtain control of enough of them in advance of making a request
    - It is now much harder for a single entity to gain control of a large number of highly trusted nodes, because they need to have many independent high reputation users to do so
    - An attacker needs to gain control of at least half of the request handlers in order to manipulate an item
      * Actually it is not possible to manipulate an item because every single node in the network will check the validity of the request
    - The only remaining purpose of request handlers is to select the ID and timestamp of created items
    - There is no longer a master node. To gain control of timestamp assignment now, an attacker needs to control at least 2 of the request handlers. If an attacker controlled 2 of the request handlers, he could cause both of them to assign timestamps that are much earlier than the time they received the message, thus providing an advantage to the player who sent that message.
    - If the timestamp was taken from the 3rd RH to receive the message and there were 5 RHs, then an attacker would need control of 3 RHs (a majority) in order to manipulate the timestamp
    - Since the RHs would be randomly distributed across all nodes, for an attacker to be able to do this consistently would require control of a majority of all nodes in the network, which is completely unrealistic.
    - The timestamp assigning node could be defined as the (N/2+1)th RH to receive the message, meaning that, regardless of N (the number of RHs), an attacker would always require control of a majority of the RHs to manipulate the timestamp
    - This strategy would trade off latency for security, though, because the later the RH used to determine the timestamp, the later clients can be confident about the (maximum) timestamps of their messages
    - If the timestamp assigner is the middle RH, then the average distance between the client and the timestamp assigner will be ¼ earth-C
    - Communication between two clients A and B on opposite sides of the world would be no worse off under this model
      * On average, A’s timestamp assigner will be ¼ earth-C from A
      * The average distance between two random points on earth is ¼ earth-C so the average distance between A’s timestamp assigner and B will also be ¼ earth-C
      * A will be relatively confident that the timestamp of its message is close to the timestamp on the message from the (N/2+1)th RH to respond
      * B will also be relatively confident that the timestamp of A’s message is roughly the timestamp of the middle message it receives because, on average, there will be an equal number of A’s RHs closer than ¼ earth-C away as there are farther than ¼ earth-C away.
      * Thus the latency between A and B will, on average, be about the time taken for a message to cover ½ earth-C
      * This seems unbelievable – it is the best possible average latency for the worst case scenario!
      * Every user, regardless of how far away from each other they are, gets precisely the same average latency, which is actually very good, particularly for games
      * If the timestamp assigner was close to A on average, then the average latency between A and B would be less when A and B are closer, but the same as the above model in the worst case scenario of A and B being ½ earth-C apart.
    - Thus this model will even out the average response times across different scenarios. The average latency between users will be similar regardless of how far apart they are on earth.
    - The MMORPG context changes things somewhat because T is already the crucial vulnerability
      * If an attacker gains control of T, he can do anything in their domain of control
      * Given that controlling T provides absolute control over events, T might as well be trusted to assign IDs and Timestamps to Event items relevant to their region of control as well
      * If the timestamp assigner is the middle request handler to receive the message, then an attacker would require control of the majority of T to manipulate the timestamp
      * This is perfectly satisfactory since an attacker must be prevented from acquiring control of a majority of T anyway
      * The Nodes in T should roll over regularly. They only need to roll over faster than a resourceful attacker would be able to hack into a new Node on average
      * Hacking into a Node with no initial knowledge of its network address or physical location would be extremely difficult
        + Many nodes would only be accessible by the Zutha network, which would only accept messages that are valid Zutha messages
        + Even if a resourceful attacker had compiled a database of the locations of every Node and the addresses of every internet Node, they would have to gain physical access to any nodes not connected to the internet in order to crack them
        + It might be very difficult to gain physical access to nodes that are satellites or solar powered UAVs flying kilometres above the earth’s surface
      * Thus Nodes would not need to be rolled over very regularly, but they might as well be, so that any dishonest Nodes that end up in the control group would not have access to secret information for very long
    - G and H are unnecessary: all requests might as well be handled by the controllers of the game region – T, in this scenario
  + A sends a request to T to shoot an arrow at the place A thinks B is
  + The first (and every) member of T to receive A’s message will immediately forward the preliminary response (without official timestamp or ID) to all relevant parties, including B
  + When B receives the first message from T, it will know that A shot an arrow probably sometime between the timestamp on the first message and the present
  + Meanwhile, B has sent a message to T requesting that his character step to the side
  + If A and B acted at close to the same instant, B is likely to receive the message from A at a similar time that it receives the response from T for its own request
  + Both of these initial responses from T will have timestamps that are likely too early
  + As soon as the middle response of both requests has been received by B, B’s client can guess pretty accurately what the timestamp of each event will be, and display the result to the user
  + B’s character will hang between the user giving the command and the middle response arriving from T
  + If technology gets good enough, the hang time could be brought as low as 100ms on average and 200ms at worst for all users in all locations on earth
  + It is possible that B’s request is returned quickly, but A’s request is delayed and arrives later even if it’s official timestamp ends up being earlier than B’s
  + In this case, B would show it’s character stepping to the side with no arrow in it, and then a moment later suddenly having an arrow in it
  + I think the nature of the architecture will make the worst of such events very uncommon
  + One thing that will help is that B will likely receive the first message from A before its own request has its timestamp confirmed
  + B will thus know that it is going to be a close call between whether its character is hit by the arrow or not, and it can hesitate an extra fraction of a second or two to wait for enough of A’s messages to arrive to confirm its timestamp.
  + The case with the arrow is actually much less likely to cause a discrepancy than this, because the arrow takes time to travel
  + B will likely have received A’s message about firing the arrow before it receives a response back about its sidestep request
  + B will calculate the trajectory of the arrow and know at what time it would hit its character by the time it receives notification of the timestamp for its sidestep request
* Distributed MMORPG allocation of control
  + If the game world is in a 2D setting (such as the surface of a planet as opposed to empty space), it should be divided up into overlapping pillars
  + Each half of a pillar, in each dimension, should overlap half of another pillar
  + Any point in the game world will lie within 4 pillars simultaneously
  + Each pillar is controlled by 3 nodes
  + Any given point in space is thus controlled by 12 nodes
  + An object at the intersection between pillars could overlap up to 9 pillars
  + Perhaps a better model would be pillars that only overlap each other by 33%
  + This would mean there would be areas controlled by just 1 group, as well as areas controlled by 2 and 4 groups
  + I’ve found a pattern of overlapping squares for which any point lies in either 1 or 2 squares
    - Each square X has 4 squares spiralling around it, not overlapping one another, and leaving a small square in the centre of X
    - There are intersections of 3 and 4 squares in this model, though
    - In this model, the intersection of 4 squares occurs on the boundary of all of them, which is worse than the simpler overlapping model
  + It would also be possible to use squares with cut-off corners (octagons)
    - This would mean regions are owned by either 1 or 2 groups
    - However there would be short intersections where 3 groups meet, and points where 4 groups meet
    - This is the same as the spiralling squares model, except with more complex shapes
    - No help here
  + Perhaps it’s best to just have a small amount of overlap so that there aren’t large areas controlled by too many nodes
  + There should be at least 5 Nodes per region. That means up to 20 nodes controlling a single region
  + Given that everything is done in parallel, I don’t think having more controller nodes would actually slow down performance at all
  + The distribution of Nodes in a square does not have to be even
    - The centre of a square could be controlled by 12 Nodes
    - 3 of the nodes would remain in control in each of the 4 corners
    - 6 nodes would remain in control in each of the 4 sides (top, bottom, left right)
    - This is actually the same as the first model with squares overlapping by 50%
  + When a non-public game object passes into an overlap region, the object’s current controllers share control with the other controller groups that share control of that region
  + Controller groups would also share information about objects they control with controller groups of surrounding regions.
    - Let’s say character C1 is in region R1 which is controlled by G1
    - Character C2 in region R2 (controlled by G2) is within site range of C1
    - G2 regularly keeps G1 up to date about the current location of C2
    - G1 calculates whether C2 is in C1’s line of site. If so, G1 tells C1’s client about the location of C2.
  + Controller groups would have as much time as an object resides in a shared region to transfer control to the other controller groups controlling that shared region
    - Actually I don’t think this would be useful or necessary
  + Controller groups would long have been sharing information about an object with their neighbours by the time they come to sharing control with them, so their neighbours should be able to seamlessly take up control as soon as they know the object has passed into their region
  + When a player moves into a shared control region, it would immediately know to start sending requests to the larger control group (who will act as the request handlers for every request)
  + If a controller group tried to do something which didn’t match the rules of the game, their neighbours would detect this and punish them
  + A controller group would be able to manipulate a character, though, against the will of its user
  + Actually they couldn’t even do this because the character’s client would be able to prove to other nodes in the network that it had given commands which did not match the actions taken by the control group responsible for the character
  + It might even be required that controller groups include the signed action requests of players they are managing
  + All messages would be distributed across the whole network, but they would be encrypted with the public keys of the Nodes that need to know the information they contain
  + If there was ever a discrepancy between the calculations of different controller groups, both groups could be expected to unencrypt the relevant data so that other Nodes can get involved in resolving the dispute
* Game clients could speed up time for certain events so that the visuals can catch up with the actual timestamp associated with the event
  + For example an explosion might occur on a distant hilltop which has an animation that lasts 10s
  + A client X might be notified of this explosion 1s after it happens
  + X could play the animation slightly faster so that it takes 9s and finishes at exactly the time the explosion finishes according to the game clock
  + Similarly, an enemy might have run out of some bushes towards X’s character 1s ago
  + X would increase the movement speed of the running animation until its position in the display matches its actual position according to the game clock
  + I think MMORPGs already use this time acceleration effect to some degree, but if there is really high latency, they also often skip sections of animation
* Message paths
  + A client X sending a message to a single target T sends it to a single neighbour A
  + A replies when it receives X’s message so X knows that A has taken on responsibility for the message
  + If A does not reply before X’s timeout period, X tries to send the message to the next best neighbour
  + Assuming A does reply, it is now A’s responsibility to ensure the message reaches T
  + A sends the message to its neighbour B
  + If B replies, it is now B’s responsibility
  + B sends the message on to C in the same way
  + There is a flaw in this strategy: if any node in the path fails (accidentally or deliberately) when they have responsibility for the message, the whole transmission fails
  + B could reply to A again after C has replied to B
  + B would also reply to A if it cannot get a message through to any of its neighbours
  + Now if A didn’t get a second reply within its timeout period, it would know that B has failed or been delayed to an unacceptable degree
  + A could then try to send the message through a different one of its neighbours
  + B might reply a second time after A has dispatched another request to a different one of its neighbours
  + This would have the effect of creating a fork in the message path
  + This is probably the best time to create a fork: when the current path has been delayed or possibly blocked
  + A would also try to send through a different neighbour if B replied a second time with the message that it was blocked
  + A Node could still deliberately block the path of a message
    - For example, B could reply to A that it has received A’s message, then reply again, pretending it has received a reply from its neighbour when it really hasn’t
    - B could be required to send A proof that C has received the message by sending C’s signed reply to the message on to A
    - This would reduce the degree of location privacy of the network, but only slightly. Nodes would know the next two nodes in the path towards a target node instead of just the next one node in the path – their neighbour.
    - If B and C were owned by the same entity they could still work together to convince A that its message had been passed on
    - It would be very easy for B to select a neighbour that it owns to forward A’s message on to, so sending A a reply signed by C would not really prove anything
    - There is already a lot of redundancy in the network
    - There is not a single scenario where a single message is important
    - Thus there would be no incentive for B to trick A like this
    - The only significant effect B could have by blocking A’s message would be too cause T to be punished by failing to reply to A’s message
    - B would not be able to do this very often, though, because B’s neighbours would learn that B is not a good neighbour to send to to get messages to T, so they would stop sending messages to T through B
* I wonder whether there will ever be more than 35 trillion nodes in the network?
  + If so, the ID system will need to be upgraded, because it currently only supports up to 35 trillion Nodes with the ability to assign IDs
  + The first digit of an ID must be numeric and it indicates the number of place values used for the Naming Node ID
  + All digits after the Naming Node ID are used to distinguish different items named by the same node
  + IDs starting with 9 are followed by 9 digits in base 32 to define the Naming Node ID, which means over 35 trillion possible Naming Node IDs
  + If I needed more, I could allow the first digit to be any character in the base 32 range
  + This would allow 32^32=1\*10^48 Naming Nodes
  + If I did this I would not be able to identify IDs in strings by their first character being numeric
  + It wouldn’t be hard to just assume that an unquoted word in a query represents an ID with first priority, and a search keyword if it does not match an ID
  + IDs that start with letters will be at least 12 characters long, so most words will not match an ID
  + I think I should use this rule from the start; this way there’ll be no chance of ever running out of Node IDs instead of just practically no chance
* Node owners would be the receivers of their revenue from Zutha
  + A user could hire the services of a cloud computing provider at a flat rate and potentially earn profits by becoming a Zutha Node owner
  + All users would also require client hardware to connect to Zutha
  + Many Client devices will be powerful enough to host some categories of items on Zutha and will thus become full-fledged Nodes
* Request handlers with timestamps after the selected timestamp will be punished while request handlers reporting timestamps before the selected timestamp will be rewarded
  + No this won’t work as it will give all nodes an incentive to falsely report timestamps that are too early
  + Punishment and reward for Node response times would better be handled by nodes reporting on each other’s ping times from their perspective
* The theoretical minimum latency for a global distributed network is in a similar range to the latency of the human brain
  + With brain-machine interface, it might be possible to significantly reduce the time between people’s thoughts and their detection by their game client
  + It might thus be possible to achieve an experience of zero latency virtual reality even when playing against an opponent on the other side of the world

# Sat 5-Feb-11

* A proportion of the revenue zutha gives to items is divided among real money investors in the item.
  + Investors in an item thus partly own the item and receive a dividend from it which depends on its worth in zuth and zutha's income.
  + This creates an incentive to invest real money in promising projects like open source software projects that you expect to become very popular on zutha.
  + This investment will provide money for the project to get underway.
* Allow steepness of investment vase to be modified individually for each item.
* Shortest path finding
  + During shortest path finding, nodes should only send on a message once, but they should emulate having forwarded the message every time it arrived by delaying their response to the slower incoming messages by the same amount as they arrive after the first message.
  + During a shortest path finding propagation from S, the first neighbour to send X a message from S is the closest of X's neighbours to S.
  + A shortest path finding propagation from S tells S the fastest way to get a message to any node, and tells any node the fastest way to get a message to S.
  + Nodes should run shortest path finding propagations when they first join the network and on random occasions after that.
* Location privacy
  + Clients don't tell neighbours that they are the source of messages they send. The user is the signer of the message.
  + The client can choose a new GUID regularly and its neighbours will think they have lost a neighbour and gained a new one.
  + Users can send out shortest path finding messages with their user id as the source. Even if the user's client changes its id, the network will still find the user because it is nearby its old location. Its neighbours will receive a message addressed to the user. They will have lost the neighbour that used to be closest to that user (the client's old id), but when they send out seeker messages one of them will go to the new unknown neighbour (the new client id).
* Neighbour connectivity weighting is a rolling average of latency to a target through that neighbour. Nodes can report that they have moved and other nodes will wipe their rolling average data on the moved node and start again.
* Even nodes need not tell their neighbours who they are. Neighbours can just be identified by GUIDs.
* Internet connected nodes need not associate their node id with their ip address. Nodes can post their ip address publicly on a list of internet connected zutha nodes without associating the ip address with their node id. Nodes can then connect anonymously to these public nodes, sharing their ip address only with them.
* A node can artificially increase latency times between it and a neighbour to make that neighbour think it (the node) is further away than it really is.
* No node has any idea where any other node in the network is - it doesn't even know who its neighbours are - and yet every node can communicate efficiently with every other.
* Triangulation could roughly locate a node. Using a single random remailer with each request would make a node or user completely unlocatable.
* During shortest path finding, nodes could send back average latency information they have about other nodes.
* I think it would be unrealistic for a new node S to do a complete shortest path finding propagation to every node in the graph.
  + The initial propagation should be to tell other nodes about S's location.
  + Messages would go out but none would come back.
  + The first neighbour a node X receives S's locator message from would be the closest of X's neighbours to S.
  + Only nodes that intend to be hosts would need to do this.
* In order to know of the existence of a node X, a node S must have received a login message from X - in which case S will know the closest neighbour to X - or S must have asked the network for the list of currently connected nodes, which would include X.
  + For X to be in this index, most of the network must have received its login request and will know how to find it.
  + If X is a very old not which has not been contacted in a long time, then its location might only be known by a small section of the network.
  + The first node to contact it - S - will send out seekers across the whole network.
  + Eventually the section of the network that knows X will be found.
  + In the process of S locating X, it also tells the rest of the network how to find X...
  + no I don't think this is true. S will find X with an expensive amount of redundant parallel paths. X will receive a lot of requests from S and know that it is no longer well connected.
  + X should then send out another locator beacon to advertise its location to the whole network again.
* Pathfinding
  + Path takes the form of an arbitrary size seeker beacon, followed by relatively straight paths through the part of the network that knows the target, and possibly another seeker beacon at the other end of the target has moved, changed its GUID identifier, or disconnected.
  + The final seeker beacon is limited in size to a certain number of hops. The first node to send out a seeker beacon will start a counter with the message that is incremented with each hop.
  + Receivers of a seeker beacon do not use their existing knowledge of the target's location, but continue propagating the seeker beacon. This is so that messages can be sent "uphill" against the direction the target used to be in.
  + The problem is how does a node know when to start a seeker beacon rather than just try another route?
  + I'm starting to think the seeker beacon strategy doesn't really work at the end of the path.
  + The only nodes that need to be found are named nodes (with node IDs). Such nodes should have the responsibility of notifying the network when they move by sending out another locator beacon.
  + Only clients should need to move often but clients never need to be contacted impromptu - clients always initiate contact.
  + If a node only moves a short distance, gaining some new neighbours, it can send out a minor locator beacon which would only cause receivers to relay the message if their closest neighbour to the source changes.
  + Actually perhaps this should be the rule for all locator beacons. Once a message reaches a communications highway it is going to travel along it the same way it did last time so there isn't any need for it to do so again.
  + If an internet connected node receives the first locator message from S through a different neighbour than its previous closest then it should forward the message to all the internet nodes it knows about.
  + S may have found another internet node so the two internet nodes need to be compared by the rest of the internet nodes to find out which is closer to S.
* Nodes should be able to ask each other their location, but using their GUID as identifier because only neighbours should tell each other their location.
  + A node can use triangulation from the positions of its wirelessly connected neighbours to figure out its own location.
* Arrow fired at invisible character scenario revisited.
  + Both characters' clients connected to T on entering T's region. They each regularly connect to new members of T as the nodes in T cycle through. They query the network each period for the members of T for the next period.
  + All members of T immediately forward messages they receive to their connected users
  + On average, A's timestamp assigner X and B's timestamp assigner Y will each be 1/4 earth-C from both A and B but may be 1/2 earth-C from each other.
  + If Y receives B's message first it will not immediately tell A about B's sidestep because B is invisible to A.
    - But Y will on average receive A's message at a similar time to B's if both A and B sent at about the same time.
    - If A's arrow is going to hit B (Y can predict this reasonably accurately a few hundred ms in advance) then Y will tell A about B's location so A can display B’s character when the arrow hits.
    - Y will always tell B immediately about the arrow so B can figure out for itself whether it will hit.
  + X will behave similarly to Y
    - If X receives B’s message first it will not immediately tell A about it
    - Once X has received both A and B’s message, it will be able to estimate whether the arrow will hit
    - X will be the timestamp assigner of A’s message but won’t know it for sure yet
    - Since A will have been communicating with T for a while now, X will likely have been the timestamp assigner for A for a while – at least since the last change of T’s members
    - X will have approximate timestamps for both A and B’s actions, but a more confident estimate of A’s timestamp – probably its own timestamp for A
    - If X estimates that the arrow will hit, and the error range around the two timestamps (defined by the typical spread of timestamps for a message to T) is smaller than error margin in deciding whether A’s arrow will hit B, then X can tell A about B’s location and send A B’s sidestep message.
  + There will likely be a member of T, Z, closer to both A and B than X or Y - probably between the two of them so that the message path A-Z-B is close to 1/4 earth-C. This means both clients will on average receive notification of each other's events at about the same time that they officially happen at X and Y respectively.
  + Let’s say Z is very close to A
    - A will receive its first response from Z
    - Z will not know about B’s request when it responds to A’s message and so it will not tell A about B even if it turns out the arrow will hit
    - This should not make any difference to A’s behaviour
    - A will have displayed the arrow being fired as soon as the user gave the command, and thus before the official time the arrow gets fired
    - Actually, A might have a policy of delaying the display of user events by a standard period of half its latency with X, which would lead to them being displayed at their official time on average
    - Z’s reply to A’s message means little to A except that it is allowed to fire
    - Perhaps for this reason A should wait at least for the first response from T before displaying the action; there might be something secret to A which currently blocks A from firing. I don’t think it would be necessary for the client to be kept in the dark about whether it is allowed to fire, but there might be other kinds of events where A should wait for permission from T to display events.
    - If the time taken for A’s arrow to reach B is longer than delay(AX+max(BX,AX)) then A will probably be notified of B’s presence before the arrow passes him.
    - Since the minimum time (according to the speed of light) for a message to travel this distance is 66ms, with 200ms being realistic for a futuristic global network, it is likely A will find out about B in time.
    - When A knows that the arrow will probably hit B, A should slow down the arrow until it matches the trajectory it would have taken if launched at its official launch time according to the game clock.
  + Z will probably be the first RH in T to be able to get a message to A after knowing the exact timestamp of both events
    - Let’s say Q is the most distant RH from A and Z, and R is the most distant RH from B and Z (such that the distance BR+RZ is maximised)
    - Z will know the exact timestamps of both events after A’s message has travelled via Q to Z and B’s message has travelled via R to Z
    - This will take delay(max(AQ+QZ,BR+RZ)) which will be maximised by Q being ½ earth-C from A (and Z), leading to delay(1 earth-C) > 133ms
    - At this point Z will know for certain whether the arrow will hit and will tell A
    - Given that all of T is familiar with the latency profiles of both A and B, the RHs in T will be able to estimate pretty accurately what the official timestamp of a message will be when they first receive the message.
    - Thus Z will know to a reasonable degree of certainty whether the arrow will hit as soon as it receives the first message from B
    - If Z estimates that the arrow will hit despite B’s last movement command, it will tell A about B’s location
    - Z should wait until there is nothing B can do to avoid being hit by the arrow before telling A about B’s location
    - Since Z is right between A and B, this setup achieves the minimum possible latency: the time taken for a message to pass between A and B, despite involving secret triggers managed by a (distributed) server.
  + Clients will want to display user events when the user actually gives commands so the clients will have negative latency for user events of 1/4 earth-c on average.
    - Latency of 1/4 earth-c could theoretically be brought below 50ms.
    - Users would hardly notice if their clients waited this long to perform their commands.
  + A client could calculate the usual latency between sending commands and their official timestamp - this will be about half the latency of receiving the response back from the middle host, which tells the client the official timestamp.
    - If the client waited this long before displaying its user’s commands, the user would see them happen at roughly their official timestamp.
    - Let’s say another client B acts at the same time as this client A and both commands end up with the same timestamp
    - A will usually receive, at about the same time it display’s its user’s command, the preliminary notification of B’s command
    - A can estimate roughly the official timestamp of B’s command – it will probably be about the time A receives this first notification of the command – by adding to the timestamp of the preliminary notification, the average timespan between the first and middle receipt of a message. Every message A receives will contribute to its measurement of this average.
    - This means that both A’s event and B’s event will occur at very close to their official timestamp on both clients
  + Given that events are triggered some real time before they will interact with other users' events, it may not be necessary for clients to delay displaying their users' commands.
    - If the arrow is going to hit B, it will need to have been shot some real time before the moment of collision.
    - B will likely have received A’s event and be able to estimate the moment the arrow was fired before the moment it should collide with its character.
    - If B's character now steps to the side, B will know whether it will be a close call with the arrow because it knows roughly when the arrow would hit and can estimate the delay before the official moment its character will step to the side.
    - B can slow down time for these events while it waits for the official timestamp of its character's sidestep command.
  + If there is no chance of a close call, B can display all its user's actions immediately.
  + Let’s examine this scenario with numbers
    - Let’s say the time taken for a message to get from A to X (roughly ¼ earth-C) is 100ms (this is relatively conservative since it would take light 33ms in a straight path)
    - The paths AY, BX, and BY are also 100ms
    - A and B are also ¼ earth-C apart
    - Z is very close to A so the path AZB takes 100ms also
    - Q is ½ earth-C away from A and Z so the path AQZ takes 400ms (something greater than 133ms anyway)
    - At time 0, A gives the command to fire the arrow
    - A’s command will reach X and be timestamped at t=100
    - A’s command will reach B via Z at t=100 also
    - Let’s say the arrow takes time 400ms to reach B’s character’s position
    - Z (being right next to A) will on average find out about B’s actions at about the moment they official happen (when B’s message reaches Y)
    - Z will find out about A’s actions before they officially happen
    - At t=0, Z will effectively say: A is going to fire an arrow in 100ms and it will take 400ms to hit B if B does nothing
    - Let’s say B’s sidestep will take his character out of harm’s way in 200ms (because the arrow will just miss his head as he begins to move)
    - So B must officially sidestep before t=300 in order to avoid being hit
    - If Z receives B’s sidestep message just before t=300, Z can estimate that B will avoid being hit by the barest margin and will not notify A of B’s presence
    - If Z has not received a movement command from B by t=300, Z will know that B cannot get out of the way in time and will alert A of B’s presence so A can display the arrow hitting B.
    - By t=300, Z will have received X’s relay of A’s message
      * Z should have received X’s message at about t=200
      * This will be roughly the middle response Z receives because Z is close to A and X is the middle RH to receive A’s message.
      * Z will also know that X is usually the timestamp assigner for A’s messages (or has been with this configuration of T)
    - When Z receives X’s message it will be pretty sure that the timestamp on this message is exactly the timestamp of A’s fire command
    - Thus Z will now be almost certain that t=300 is the last moment B has to act to avoid being detected by A
    - Z could give some leeway to B – perhaps by waiting until t=350 to notify A of B’s presence
    - It is possible that Y receives B’s message before t=300 but Z doesn’t receive the message until after t=350 even though they usually receive messages from B at the same time.
    - Y’s message to Z should still arrive after the usual 100ms, though, since it would be unlikely for 2 paths to be delayed significantly in such a coincidental manner.
    - If Z receives a sidestep message from B before t=350 it will wait for confirmation from Y of the timestamp of B’s message before deciding whether to alert A of B’s presence
    - In this case, Z would receive Y’s message at t=400
    - If Z has not received a message from Y at t=400, Z will immediately send a message to A to notify it of B’s presence so A can display the arrow hitting B
    - T=400 is exactly the time the arrow hits B from A’s perspective
    - The official time the arrow hits B is t=500, but A displayed the event as soon as the user gave the command so the trajectory of its arrow is 100ms ahead of its official path
    - Actually, A would know the official timestamp of its shoot command now, and thus the correct trajectory of the arrow.
    - A should slow the path of the arrow subtly until it matches its official trajectory
    - Thus A will receive notification of B’s presence at t=400 – in time to display the previously invisible character B when the arrow hits him
    - These numbers are of course pretty arbitrary, but they show, I think, that a game could be made to flow smoothly despite involving messages that traverse the entire planet.
    - It is of course possible that it is such a close call that Z’s estimation of the timestamp of B’s sidestep is off by enough to cause A to incorrectly find out about B
    - To A’s user, it might look like the arrow slices the side of B’s face as B flickers into existence. A moment later, though, A would realise the arrow didn’t hit B and would have to fix up its error. The arrow would suddenly jump back into flight and B would disappear again.
    - According to the numbers in this example, it will take 400ms for A’s message to get to Q and back to Z so Z will only know the exact timestamp of A shooting the arrow at t=400.
    - If B sidesteps at almost exactly t=300, then B sent the command at t=200 because it takes 100ms for B’s command to reach Y where it receives its official timestamp
    - Let R be ¼ earth-C from B and ½ earth-C from Z, so a message from B to Z via R takes 300ms
    - Let’s say R ends up being the timestamp assigner for B’s message
    - Let’s say Z, at t=400, estimates the timestamp of B’s message as 310ms after receiving the relays from half of T
    - Z will tell A about B’s location at t=400
    - Z will receive the response from R at t=200 + 300ms = 500ms
    - Let’s say the timestamp R assigned to B’s message is 290ms
    - Z will immediately tell A about the amended timestamp for B’s message
    - A will have calculated B’s position (still invisible) to be in the path of the arrow by the barest margin
    - at t=500, when A receives the second message from Z with the amended timestamp, A will be showing the arrow just entering (or at least scraping) B’s body
    - As soon as A realises that B’s movement trajectory is slightly further ahead than A had realised, A will make B invisible again and let the arrow pass by
    - A’s client will see the barest flicker of B’s body and might mistake it for a deliberate effect caused by his arrow passing so close by the invisible character.
  + This system, despite involving a lot of estimation during the time-critical moments, would be quite resistant to accidental and deliberate malfunctions I think
    - Z waits until it has received B’s message from roughly half of T before being confident of its timestamp
    - Y should on average be the middle RH from which Z receives messages from B
    - It’s just as likely though, that Y is one of the first or one of the last RHs to relay messages to Z from B
    - For each RH in T, Z will know in roughly what order they receive messages from B
    - Y will know the typical distribution profile of the first half of the messages it receives from B
      * That is, it will know whether the first half of the messages it receives have a median timestamp which is higher, lower, or roughly equal to the median timestamp of all the messages T receives from B
    - Thus once Y has received half of the relayed messages from RHs that came from B, it will be able to calculate to a high degree of accuracy, the median timestamp – and thus the official timestamp – of B’s message
    - (I knew I needed a better phrase than “middle” timestamp) – it’s “median” timestamp)
    - If B’s message to Y is delayed, Y will not be the timestamp assigner for that message so this will have no effect
    - If Y is the median receiver of B’s message, but Y’s relayed message to Z is delayed, then Z will estimate the official timestamp from the distribution profile of the first half of the relayed messages it receives
    - Z is similarly unessential. There will be another RH in T which is reasonably close to A and can provide A with low latency information about game events.
* The more controller nodes used per region in a distributed MMORPG, the better
  + The more controller nodes there are, the more evenly they will be distributed across the planet
  + The evenly distributed the nodes, the more consistent the latency profile will be
  + With more nodes, it will be more likely that some of them like between the clients of two interacting players, thus reducing the latency between them
  + With more nodes, nodes and clients will have more data points to estimate the median timestamp of messages, and so will be able to do so more accurately, reducing the degree to which clients have to change their minds about the details of events that may have already been displayed
* MMORPG controller groups should distribute all their data (in encrypted form) to all nodes who accept MMORPG data
  + They should wait until low activity times to do this, though, otherwise all nodes would be receiving all traffic from the whole distributed game, which might overload many Nodes
  + It would take a lot less resources to simply store lots of incoming data than to process those requests as the controller nodes have to do, though
* Shortcuts menu
  + Inbox
  + My investments
  + Favourites
  + Recommended items
  + My items
  + My actions
  + Items created by me
  + <Custom queries….>
* You can donate zuth to an item.
  + This adds to item reputation in a separate pool to investments.
  + You cannot sell this donation like you can sell an investment.
  + Used to pay someone in zuth. You spend zuth; they get reputation.
  + Maximum donation is your reputation.
  + Donations shrink if your reputation falls below their size.
    - This is to allow the community to take away reputation given without merit by destroying the reputation of the donating user
    - Actually perhaps total donation to a single user must not exceed your reputation
    - Otherwise a high reputation user U could donate tiny amounts of zuth to many items owned by a target user T and the community would have to bring U’s reputation down to practically zero in order to remove any reputation from T
    - Perhaps this rule should also apply to investments: your total investment in items owned by a given user cannot exceed your reputation
    - A user U could still max out his reputation investing in each of T1, T2, and T3, and then get all of those users to invest in X
    - It would take a lot longer to get X’s reputation to 3\*U’s reputation, though because T1, T2 and T3 would take a long time to accumulate the cash to invest that much in X.
    - Also, X’s reputation would fall at 3 times the rate of U’s reputation as the community pulled down U’s reputation. This is because each of T1, T2, and T3 would lose reputation at the same rate as U
    - Actually U could invest small amounts in a very large number of users who would all invest in X. Then U’s reputation would have to be brought near zero to have any effect on X’s reputation.
    - The main barrier would still hold though: it would take a very long time to accumulate the cash in new users to invest more than U’s reputation in X
    - Also, if U forces the community to push his reputation to zero in order to end his exploit, then… well, he’s more likely to have his reputation pushed to zero
  + Zuth that is donated disappears from the economy.
* Total zuth injected into economy each period is not fixed. It is calculated to increase size of the economy by a fixed percentage compared to previous payday.
  + If more zuth is taxed or donated in a period, users will get a larger than normal salary at the end of that period.
* People can digitally sign stuff and store it on zutha.
  + They can regularly change their private key and even release the old ones publicly and their signature remains secure because documents signed on zutha have a secure timestamp.
  + Agents who discover a private key you had in the past cannot sign a document with it that has a valid timestamp for that private key.
  + Each period you could tell zutha what your public key will be in the next period.
  + Attackers would have 1 time period in which to both crack and use your private key (because it would expire at the end of the period)
  + This scheme would only be useful for signing of course.
  + If you thought your private key had been discovered, you could announce this (using the same private key that was stolen) and future actions by that key would be frozen until the associated account could be authenticated with the owner using some other means - such as through openid accounts or calling them at their university etc.
  + If someone stole your private key they could read private messages sent to you during the lifetime of that key, but do nothing else.
  + If someone stole your client device and managed to hack into it they could theoretically discover all your past keys. But while they are hacking you could authenticate yourself with zutha by other means and get yourself a new client. At worst the attacker could read all your private messages.
  + Users that want to stay very anonymous could rely on many secret questions as their backup authentication. When their user account is first created (and is thus worth nothing and not worth infiltrating) the creator should encrypt lots of answers to secret questions and send them to zutha with their associated questions. If someone stole their client (or they lost it) they could buy a new one, logon to zutha with a fresh public key and authenticate the fresh public key with their zutha account by sending the answers to the secret questions to zutha. They would then have to make a new batch of secret questions because the answers to their old ones would now be known to the whole zutha network. Actually perhaps only a select set of high reputation nodes would need to know the secret-answers in order to confirm the user’s identity. They could then vouch for the user’s identity.
  + It would be cool if answers to secret questions could act as private (asymmetric) keys.
  + Question and answer could be hashed together to prevent dictionary attacks.
  + Asymmetric key pair could be determined deterministically using the hash of the question-answer pair as the seed for its pseudorandom function.
  + Public key of this algorithm is sent with question. Private key is thrown away. Private key can be generated on a fresh client with the answer to the question - which the user holds in his head.
  + If a user needs to re-authenticate, he generates an asymmetric key pair from one of his questions on a new client. He then chooses a new random public key and signs it with the private key generated from the secret question before sending this to the network.
  + Several question-answer pairs could be combined for the generation of one asymmetric key pair. An attacker would then have to guess multiple answers at once which would be far harder even for an attacker with a lot of knowledge of the victim (a friend or relative? ...or the CIA?) let alone a computer.
  + Your client would know you have answered your secret questions correctly because the public key it generates would match the one associated with those questions on Zutha.
* Drink taps as waypoints in "zuthaspace" or "z-space" (play on "d-space" from daemon)
* Remailing
  + User U sends message m3 with client C to target T via remailer R.
  + A is a node between C and R.
  + B is a node between R and T.
  + m3 is contained within a private message to R: m2.
  + m2 is contained within a public message m.
  + m is a normal looking message container containing a private message to R.
  + U makes a new client C1 with a new random GUID
  + m has source C1 and target R. This is all m contains and is all A knows about the message.
  + R receives m and reads m2 to discover it is a remail request to T from C1. For all R knows, C1 might be another remailer.
  + R creates a new client C2 which R manages on behalf of C1.
  + R sends m3 to T inside m'.
  + m' is a public message from C2 to T. This is all B knows about the message. Neither B nor T know about R.
  + B updates its connectivity weightings to point messages to C2 towards R.
  + T opens m3 and discovers it is from U. T thinks U has sent it a standard private message using C2 as a source alias to conceal its location from relaying nodes.
  + T replies to U with private message r2 contained within public message r. r is from T to C2. This is all B knows.
  + R will receive r because its neighbours know it is their closest neighbour to C2.
  + R knows C2 is an alias for C1 so it forwards r2 to C1 inside a new public container message r'.
  + r' is from R to C1. A thinks r' is R's reply to m.
  + C will receive r' because C's neighbours know that C is their closest neighbor to C1. They learned this when they received m from C1 through C.
  + C now opens r2 with U's private key. r2 is a private message from T to U. U does not know what direction T is in because it has never sent a message directly to T. U could get this information without sacrificing privacy though by sending out a seeker message from one-use client.
  + T thinks U is in the direction of R but doesn't know about R.
  + If U communicated to many targets through R they could collectively triangulate U's apparent position to roughly the location of R.
  + U could use a different client alias and different remailer to communicate with each target node.
  + None of U's remailers would know anything about U and even if one agent owned all U's remailers, it wouldn't be able to find anything common across them to identify them all as remailers for the same node. Every remailer would have a series of communications between an anonymous source and a public target. No 2 remailers would know about the same anonymous source.
  + An attacker would have to track many of U's targets to their associated remailer. This would involve hacking in to every node on the path from the target to the remailer until the remailer itself is hacked and discovered to be the remailer. Actually only some of the nodes on the path need to be hacked but then you'd have to guess which nodes are on the path which would be practically impossible. Hacking into every node on the path would be practically impossible as well though.
  + Each remailer would have to be searched for the source client associated with the alias it uses to remail messages to the target of U's that it manages.
  + Once many source clients are associated with U, U's rough position can be triangulated from all the remailers. That's unless U was using more than one remailer in which case the remailers' source clients will not triangulate to one position but will all be in random directions from each remailer.
  + T could be asked to be a second remailer.
    - Message m3 could be a remailer request from C2 to T2.
    - This scenario is symmetric with that faced by R. T thinks C2 is sending it a remail request from the direction of R, just as R thought C1 was sending it a remail request from the direction of C.
* Nodes should get rated by users on several metrics like
  + performance (done automatically by clients on behalf of users)
  + secrecy (voted on by users and estimated automatically by set intersections of groups of nodes responsible for releasing secrets)
  + correctness (done automatically by whole network)
* Discovery beacon
  + User U uses client C to send out a discovery beacon from one-use client C1.
  + The discovery beacon asks C's neighbours to send back to C1 their average latency with every major node they know, where a major node is one that can receive requests. If a neighbour does not yet know a node it should send out its own discovery beacon. It should tell C1 the time taken to run it's discovery beacon so C1 can subtract this from its latency calculations.
  + All nodes (including inferior nodes like clients) should send out discovery beacons when they first connect to the network.
  + C could use a different alias client for each neighbour so it would be harder to know there is a node at its location.
  + Nodes should use GUIDs to identify themselves to neighbours in the same way as clients (clients are also nodes).
  + A major node (or host) will send out a locator beacon from its host id but through a one-use node id - a GUID.
  + A host can send private messages from a node id if it wants to keep its host id secret to nodes carrying the message. For example if the host is acting as a remailer.
* Give reputation to a community of poor people and divide it evenly. Then let them invest in each other to allocate reputation based on individuals' ability to use money wisely to increase the wealth of the collective.
  + Donate money to the group so it is divided according to this measure of reputation.
  + Individuals could then vote with their reputation on the best use of money to alleviate poverty in their community.
  + They would also have money to "vote" on the things they want by buying them.
  + People that are perceived to be wasting their money could have the donated part of their income reduced by their community.
* Transactions can be processed instantly by zutha in terms of zutha dollars without any real money changing hands.
  + Zutha would just make the transaction between zutha accounts.
  + Zutha dollars would be defined as a bundle of other currencies.
* Employers in poor African villages that use zutha for some transactions would have an incentive to buy zutha clients for employees that don't have them in order to reduce transaction costs.
* People with no way to get money in or out of zutha can make money on zutha by gaining reputation and grow their money stock by investing.
  + As businesses start to accept payments by zutha people will be able to buy things with this money - all without having a financial identity.
* This still wouldn't help money launderers because all transactions would be public knowledge.
  + Wherever money enters or leaves zutha, the identities of the transactors would be knowable to policing agencies because they would all be backed by banks. Zutha would know the target account and at least one account owned by the source, for every transaction. If either party were suspicious, their real world identities could be discovered through the bank that owns their account.
  + A criminal could grow a stockpile of stolen money by investing in zutha. If this criminal sends money to a suspicious target using zutha, his source of money could ultimately be tracked, even if he did a lot of work to o acute the path of money through zutha.
  + If a criminal earned all his money in zutha then this would not be a criminal act. If he had no associated account and sent money that he had earned on zutha to suspicious targets then his real world identity would not be discoverable.
* Any user could be allowed to handle money for zutha up to the amount of zutha dollars they own on zutha.
  + They would get more interest this way since they would get any interest they can earn themselves minus a small amount taken by zutha.
    - If they only held zutha money they would also have to pay the money owner cut of the interest.
    - If they only own money, then they would only get the owner cut of interest.
  + Someone who holds the same amount of zutha dollars as they own could withdraw money from zutha and nothing would happen except the amount of money they officially hold for zutha would fall by the same amount as they withdraw.
  + This would only work for zutha dollars held as cash because money in investments could vary dramatically outside the user's control.
    - The amount of money they own could then fall below the amount they hold and they would owe zutha money.
    - They might have a low reputation and have nothing to lose by not giving zutha the money.
    - Zutha could confiscate the user's zutha assets but this would not incur any monetary cost on the user since he was holding their monetary value himself.
    - Zutha would effectively be bearing the user's risk and would take the loss if the user's assets fell in value.
  + A user could only be allowed to hold, say, half, of his zutha money himself. If the user failed to pay zutha when asked now, confiscation of his zutha assets would be a real threat.
  + Investments still couldn't be included.
    - You could arrange to have another user you own invest a huge amount in an item before you, then sell and take more than half the value of your (only) investment in one go.
    - It could be required that at least half your zutha money be held in cash.
    - This would be an annoying restriction and would be same as only counting cash to count towards the amount of zutha money that could be self-held.
* If you have access to instant money transfer to zutha, why keep any cash on zutha?
  + Because you think the zutha dollar will appreciate relative to your own?
  + No, because whenever you spend your zutha cash (to buy an investment or send to another user) you have to pay zutha money in zutha dollars from your bank account of money in the currency whose value is diminishing relative to the zutha dollar.
  + I don't think there is any value in holding zutha cash if you can do automated instant money transfers to and from zutha. You might as well hold the cash in your bank and get the higher interest rate.
  + So it looks like the only way users can be trusted to hold zutha money is through reputation.
* Managing zutha money without hitches for a long time could be rewarded with a slow flow of zutha granted reputation. This would in turn allow more money to be handled.
* Handling zutha money could be very lucrative because you wouldn't have to worry about the debt being called in all at once.
  + It would be like being a bank except with practically no chance of a run on the bank.
  + This is because zutha would distribute requests for money across all its many money handlers, never asking very much from any one money handler at a time.
  + It is also very likely that there will be a net inflow of money to zutha as it grows, so money handlers are likely to constantly be receiving more money from zutha.
  + It is only if zutha started to shrink that any money-holders (all of them at once actually) would have to pay zutha money in net terms.
  + If zutha shrank so much that no money-holders were able - or willing - to pay anymore, then money owners would suffer what remains as losses.
  + I can think of no scenario which would cause zutha to shrink dramatically once it became really big except maybe a major global catastrophe, in which case I think zutha would do relatively better than the rest of the world. In fact I think money would probably flow into zutha during a catastrophe because its distributed nature would make it a safe place to store money.
    - Reputation on zutha would remain valuable throughout the catastrophe so money owners would be able to trust that money holders will still want to keep up their side of the bargain.
    - In fact holding money for zutha will be such a lucrative business that money holders will have an incentive to try to keep Zutha popular.
      * Money holders will also be high reputation users, who will be receiving a large income which is proportional to Zutha’s income, so that will be another incentive for them to work hard to keep Zutha growing.
      * They could do this by continuing to create the great content (of which, being the high reputation users, they are the main producers) which attracts investment in the first place
      * They can also invest a lot in Zutha themselves, using their income from Zutha and any other income they have
      * They can even reinvest some of the money they are holding for Zutha
      * This will have a multiplier effect on the amount of money Zutha reports as being present in its economy
      * The more reinvestment that occurs, the greater Zutha’s leverage because the underlying amount held by its money handlers will not have changed.
      * Increased leverage during a crisis would be dangerous, but it may help to maintain confidence that Zutha is robust by preventing Zutha’s economy from shrinking
      * Actually reinvestment would be worse than leverage. It would involve money handlers owing money to Zutha that they don’t have. Even borrowing to buy a house or car isn’t this bad because there is some asset (the house for instance) used as collateral.
      * Money handlers would have to be made to realise that reinvesting the funds they hold for Zutha back into Zutha is not investing the money but effectively spending it. Or is it?
        + It would increase the amount of money on the table in Zutha investments without increasing the amount of money Zutha holds to back up that money on the table.
        + The reinvestor could lose money by having earlier investors sell from underneath him (as it were), reducing the value of his investment
        + This reduction in the value of the Zutha investment asset would be just like any loss from an investment in the real world
        + If the reinvestor invested well in Zutha and was able to sell for a profit, the profits would come from either outside investment which means new money flowing into Zutha, or other reinvestors
        + If a reinvestor wins money from another reinvestor, the gains and losses cancel out and nothing changes about Zutha’s “leverage” or whatever the effect of this reinvestment should be called
        + If a reinvestor wins money from outside investors then he decreases Zutha’s “leverage” be increasing his own capital stock without reducing the capital stock of any other Zutha money handler.
        + Reinvestment seems to have all the characteristics of normal investment.

An investor takes a risk in order to have a chance at profit.

A reinvestor in Zutha is simply gambling on the behaviour of the Zutha market instead of some other arbitrary market

* + - * + One could argue that investing in Zutha is a Ponzi scheme because you only make money by taking it from other people rather than contributing to the generation of real wealth, but I would disagree

For a start Zutha will generate a massive amount of wealth because it will pay people to make information that people want and provide it for free

Investment is one of the things that drives this process because it gives Zutha the income to pay the information creators

Investment also produces wealth directly by providing information about the relative quality of different concepts like products and ideas.

Investment in prediction markets provides predictions of the future, which is very useful information and thus also constitutes wealth.

Investment also generates wealth by the donated part going to fund various projects; and the part that potentially goes to other investors magnifies the amount of money donated to a project by creating an incentive for more investors to invest

* + - * + You could also argue that mostly, the way people make money from investing in the stock market is a Ponzi scheme because you can only sell a stock for higher than the price you bought it at if the demand for the stock goes up.

This means you make money by taking it from later investors

You created no wealth by making that profit because no more money was given to the company to invest for the production of wealth

Actually the way you created wealth was by retroactively providing an incentive for the first investors to buy stock from the company and provide it with capital to generate wealth.

Another way to phrase this is that the only reason those first investors bought stock is because they hoped they would be able to sell the stock for more later.

By buying the stock from the first investors you prove to future first investors that they are likely to be able to make a profit by being first purchasers of stock

By selling your stock for profit you continue to contribute to the belief that buying and selling stock can be profitable, and thus you contribute to the psychology that leads to initial investment in public companies, which leads to a lot of wealth generation.

Dividends are also only contribute to wealth creation by the incentive they create to the first investors

* + - * So what would be the consequences for money handlers that reinvested if Zutha’s economy shrank to zero?
        + On the way down, some money handlers would make net profits when they sell their investments and some would make net losses
        + The money handlers who profited will have money to spare and will easily be able to make their commitments to Zutha. Their profits will have come from other users’ losses so Zutha’s leverage is not affected
        + The money handlers who lost money from their Zutha investments will have to recoup their losses from somewhere else if they can. If they can’t, then Zutha will lose this money.
        + Something still does not add up. What happens if no one profits or loses; say every investment is liquidated?
        + Then the reinvestors will receive all their investments back, and will then use this money to pay back their debt to Zutha (by paying out to other users whose investments have been liquidated)
        + Shit! There’s no problem
        + Money handlers could all reinvest 100% of their loan from Zutha many times and arbitrarily increase the size of Zutha’s economy
        + Money handlers each take on the risk associated with that investment. If they think they can safely invest 100% of a large loan from Zutha in Zutha, then it’s up to them. If they lose money, it’s up to them to recuperate the losses somehow or risk the loss of a valuable asset – their Zutha reputation.
        + I think it unlikely money handlers will chose to reinvest a large proportion of their loan from Zutha, since Zutha is not a very reliable investment for large sums of money. For a start, any investment might be completely taken from you by the item owner because it is really a donation. And if it is not completely taken from you, a non-trivial proportion of it is lost the moment you invest, so you have to be pretty confident that the popularity of the item is going to rise to be willing to invest. The Zutha real money investment market is better designed for low key investors with not much to lose than serious institutional investors which is what Zutha money handlers will be if they want to make significant returns on their loan from Zutha.
    - It might be known that, on average, money holders will have lost money, but the amount lost will not be visible in the way losses in the stock market are, so money owners are less likely to panic. Instead zutha will seem - and will be - safer than the average bank because it will be the most diversified financial entity in existence.
    - Even if money-holders have lost money in aggregate by the time the crisis is over, they can slowly build up their accounts again from interest - provided they haven't lost so much that zutha's interest demands on a larger sum of money than they now have exceeds the interest they can earn on what they have.
    - Zutha could lower its interest rates after the crisis to make it easier for its money handlers to recuperate their capital stock.
* What is Zutha?
  + A distributed company for organising and ranking the world’s most valuable information

# Sun 6-Feb-11

* Criminals could purchase weapons (or other goods) from businesses that deal in zutha dollars.
  + The transaction would be public
  + If the source of money was legitimate, this would be no different from a criminal buying weapons the way the already do
  + Except that if the criminal had earned all their money on Zutha they would not be traceable to a real world identity
  + If the source of the money was from outside Zutha, then a policing agency could trace the identity of the purchaser of the weapons
  + If the source of the money was from a criminal act, this would have to be traced using standard techniques of tracing money laundering starting with the account used to send the money into Zutha.
* Money handlers should be able to purchase a lower interest rate
  + Money handlers would take on a larger obligation to pay Zutha (basically expanding their loan without receiving any more money) in exchange for a lower interest rate on their loan.
  + The reduction in interest rate should be large enough to make the money handler’s interest repayment costs fall slightly despite the increase in the size of the loan
  + For example a money handler could double his obligation to Zutha in return for a 51% reduction in interest rate on the loan
  + In effect this would be like giving money to Zutha and then having it loaned back long term at a negative interest rate.
  + From Zutha’s perspective this would be like being given money which it pays a small interest rate to be allowed to keep. If it spends the money (actually if the size of the whole Zutha economy falls) then its interest costs would fall.
  + Zutha would never actually be liable for any interest charges because the same user they are owed to would be paying more interest back to Zutha because of the loan they have with Zutha.
  + Money handlers would have an incentive to do this deal if they had excess capital and felt secure putting it on the line in exchange for free interest. It would be free interest because it would be interest on top of whatever they are already getting by investing the capital themselves.
  + They would be putting their excess capital “on the line” in the sense that they would be promising to pay it to whoever Zutha directed if and when asked by Zutha to do so. But they would know that Zutha operates programmatically and predictably distributes its payment requests across its many money handlers, so they are only actually going to have to pay out if Zutha’s economy shrinks.
  + This deal could be looked at as an asset that states: pays x% of S per month if Zutha doesn’t shrink; costs about 1% of S for every 1% reduction in the size of Zutha’s economy
    - S is the fixed size of the obligation and x is the free interest rate provided by Zutha for holding that obligation
  + This scheme could be an effective way for Zutha to accrue more money once it is big enough and powerful enough to be trusted to grow (or at least not shrink) indefinitely
  + It is likely that money holders with excess capital will take up this deal more frequently than money holders with too little capital. Then again, investors with too little capital will be paying a lot of interest to Zutha compared to the size of their capital stock, and will thus want to reduce their interest costs so they can recuperate their capital faster.
  + If Zutha always maintains a large reserve of savings then it can wear the costs of some money holders not being able to pay out as Zutha shrinks.
  + If Zutha always held in reserve the money promised to it through this interest for obligation deal, then it will effectively mean that the money handlers who promise money to Zutha and have the capital to back it up will act as guarantors for the money handlers who run out of capital or just refuse to keep paying Zutha as it shrinks. The money handlers who stop paying out could include some that promised additional money to Zutha, but since this money was not spent, this incurs costs on no one (except Zutha itself, but it would only accept these costs once it had run out of money handlers able to pay up, in which case the entity called Zutha would be practically destroyed anyway and the concept of it owning money would be meaningless.)
  + If you increased your obligation to Zutha you would have to pay out more for every percentage point that Zutha shrinks
    - Money handlers are thus made more vulnerable to Zutha shrinking by taking on more debt in this way.
    - If you have less capital available, then it will hurt more to have to pay more of it per unit that Zutha shrinks
    - If you doubled your obligation to Zutha without having the capital to back it up, then you would have run out of money by the time Zutha shrinks by 50%
  + There are some vague similarities between this interest for obligation deal and a public company selling a batch of shares
    - The interest is vaguely analogous to a dividend
    - A major difference is that when you buy shares in a company you no longer hold the money and cannot earn revenue from it in another way at the same time
    - The concept of being able to sell shares could be emulated with the obligation deal by money handlers paying another money handler real money to take on some or all of their obligation to Zutha
    - A money handler might do this if Zutha is shrinking and they are close to running out of money to pay Zutha when it asks
    - A money handler in such a position would have an incentive to find some way to pay Zutha so it doesn’t lose reputation.
    - Paying someone else to take on the risk of Zutha shrinking too much further could be an attractive deal for both parties
    - The payer would reduce the amount he has to pay per percentage point that Zutha shrinks, but would incur greater interest over the long term. This would be a good deal if the payer was in dire straits if Zutha shrank any further, but was confident of being able to keep up with the increased interest rate.
    - The receiver of the payment might have plenty of capital to spare and be confident that Zutha will soon start expanding again, regardless. The receiver would also then receive more interest from Zutha, or at least have the interest he owes Zutha from his own loan reduced.
    - A money handler could also be allowed to pass on the loan component of his Zutha obligation to another money handler.
      * This would be more expensive, though, because the receiver would then have to pay more interest to Zutha.
      * In fact I can’t see that anyone would accept such a deal for much less than the full amount of the loan
      * Actually if Zutha’s interest rate is sufficiently low, a money handler might accept a loan obligation from another money handler for less than the size of the loan because he believes he can make much higher returns per unit of capital than he has to pay to Zutha in interest.
      * For example if someone thought they could get more than 4 times Zutha’s interest rate they would be willing to accept a loan obligation for ¼ of its size in payment.
      * The reason someone would take such a loan with such a high effective interest rate instead of borrowing from a standard bank would be that they need no collateral except their Zutha reputation and they are confident that they will never have to pay back the loan, only the interest on it.
      * Again this is a good deal for both parties.
        + The off-loader of the loan would effectively have paid off his loan for a fraction of its size, and the receiver of the loan would get more money to invest with an associated interest rate that is lower than he can get from investing it.
        + The receiver of the loan would only take higher interest rate loans in this way if he had already maxed out the amount he is allowed to borrow from Zutha according to his current reputation.
* I can think of a source of financial trouble for zutha: major disagreement about zutha policy which causes it to split in two.
  + Money holding users would have to decide which network they remain loyal to
  + It would be possible for money handlers to continue handling money for both networks. This would mean effectively doubling their obligation by owing the old amount to two networks now instead of just one
  + Money owners would also officially own twice as much money – the same amount as before on both networks
  + If one network began to shrink money handlers might cease transacting with that network and only maintain their relationship with the other network
  + One network would shrink faster than the other if more users wanted to abandon it than wanted to abandon the other
  + When users decide which network they like better, they will likely try to extract as much money as they can from the network they dislike, and perhaps inject it into their account on the network they like.
  + If both networks shrank, money handlers would have an incentive to very quickly chose a network to remain loyal to so they don’t have to pay out $2 for every $1 reduction in their loan
  + If both networks grew, many money handlers might maintain relationships with both networks
  + If one network shrank and one grew, most money handlers would probably move to the growing network, thus accelerating the demise of the shrinking network and likely leading to its complete annihilation.
  + For a while, there will be a Darwinian struggle between the two networks to maintain the most loyalty
  + If the money handlers were to split roughly in two, then both networks would have their economies grounded on half as much capital
  + This would be the worst case scenario. More likely. One network will win out over the other and less than half the underlying capital will be lost by the winning network.
  + If the winning network manages to weather the split and continue expanding, it might be able to grow sufficiently to allow money handler to recuperate their losses. It might grow reliably even if the money handlers never recuperate their losses actually.
* A healthier way for more Zutha networks to spawn would be for them to duplicate an existing Zutha database, but start its real money economy at zero again.
  + Users would keep their reputation on the new network, but would have to migrate their real money across if they preferred to have it on the new network
  + High reputation users would be allowed another quota of money to handle. They wouldn’t want to fail to repay either network because their reputation would probably suffer on both as a result.
* You’re only allowed to pass on x% of your reputation to another user
  + Let’s say x=10%
  + A user would have to create 100 dummy users to pass on 1 times his reputation to another user
  + This exploit would be absurdly obvious, but I’d rather it be impossible or at least much harder to do at all
  + A user could be required to have a minimum reputation r provided by a minimum number of users n before it is allowed to invest
  + Users have to wait for their initial salary to accrue before they can invest anyway, so some additional restrictions won’t have too much effect on the average new user.
  + For the exploiter, though, it will add major challenges
  + To make things extra hard, a single user could be restricted to contributing a maximum of p% of a user’s reputation
  + The n different investors required to give a user permission to invest could also be required to be independent investors. Associations between users that might disqualify them from being considered “independent” could be:
    - Ownership of one user by the other
    - Suspected single controller
      * The community could label groups of users with this association when they notice suspicious coordinate behaviour between a group of users
    - Declared single controller
    - Users are friends
    - Users are related
    - Users are colleagues
    - Including the last 3 might be going too far because it includes a person’s whole social network in a dependent group
  + Actually the independence requirement could remain throughout the user’s lifetime
    - No dependent group of users would ever be allowed to contribute more than p% of a user’s reputation
    - A dependent group would be defined as a set of users with a common association with a single user.
  + For a friendship chain A-B-C-D there would be two overlapping dependent groups according to this definition
    - I can’t think how to deal with overlapping dependent groups in calculating what share each is allowed to contribute to a user’s reputation
    - For this reason, I think social connections should not be considered dependencies
    - The main purpose of identifying dependent groups is to block controlled armies from acting together to accomplish exploits
    - A dependent group should be defined as a group of users known or suspected of being controlled by the same agent.
  + To make things more general, we can remove the rule requiring a minimum of n separate investors contributing r reputation to a new user before it can invest
    - The maximum reputational contribution of p% per dependent group encompasses this rule
    - To cater for the unique case where a user has zero reputation, users should be allowed to have investment in a user up to the maximum of r\*p% and p% of the total reputation of the user
  + Let r=50 and p=20%
    - This means a new user needs a minimum reputation of 50 provided by investments of 10 zuth from 5 different users before it can invest
  + Let’s say an exploiter E has reputation 1000 and practically unlimited zuth cash
  + He cannot get a single dummy node off the ground because he needs secret control of 4 other users with at least 100 reputation each to get a single user to the point of being able to invest
  + Let’s say E controls users U1-U5 but no-one has discovered that one agent controls all of these users yet
  + Users U1-U5 have 2000 reputation each and all have practically unlimited cash
  + They can invest 200 zuth each in users D1-D100 so each of these 100 dummy nodes has reputation 1000
  + E then waits for users D1-D100 to accrue lots of cash before having them all invest 100 zuth each into user T
  + T will then have 10,000 reputation
  + This will be such an obvious exploit scenario that an algorithm will pick it up easily
  + Some reasonably high level user will come along and associate U1-U5, D1-D100, and T under a “suspected single controller” Association
    - Immediately U1-U5 will no longer be allowed to invest more than 10 zuth combined in any of D1-D100, so they will be forced to sell back to investments of 2 zuth each in each of D1-D100
    - D1-D100 will all fall back to 10 reputation each, which is not enough to invest, so they will all be forced to sell all of their investments in T
    - T’s reputation will fall back to 0
  + Even if no users feel the desire to punish E for his attempted exploit, he will have lost the ability to perform any more exploits because his hard won independent high reputation users are now practically considered the same user, so their exploitative capacity is lost.
    - They are actually worth less than a single user with 10,000 reputation though because none of them can perform actions that require more than 2000 reputation.
    - For decision items that restrict investment size to k% of reputation, U1-U5 combined would now be limited to investing k\*2000 instead of the k\*10,000 they would be allowed to invest if they really were a single user.
  + E also has 101 completely useless users with 10,000 zuth between them which he will probably delete.
  + E still has 100,000+ zuth in his 5 high reputation users. Perhaps he will reform and start investing that zuth intelligently now.
* A Conversation should be hosted be a persistent controller group
  + As with MMORPGs, this controller group should steadily roll over its members
  + Let A and B be users participating in a conversation
  + Let Z be the Request Handler (member of the current controller group for the Conversation, G) which provides the shortest path between A and B
  + A will receive a message from B after it has travelled from B to Z to A
  + This first message will contain only a rough timestamp and no ID
  + A’s client can display B’s message immediately, though, and simply update its timestamp and ID when they arrive
  + If G is sufficiently large there is a high probability that there will be a RH that allows a path AZB to be very close to length AB
  + Thus the latency for a conversation between an arbitrary group of users around the world would be as small as is physically possible. Every message would reach all other participants in the conversation by the fastest possible route.
  + Yet no participants in the conversation need know anything about any of the others
  + As soon as a new member joins (by initiating connection with G), he will start receiving messages by their fastest possible route by all other participants
  + This architecture could provide very efficient Google Wave like instant messaging, where every conversation participant receives everyone else’s messages character by character
* A user should retain the same controller group (group of request handlers) for a whole timeframe
  + This way, there would be no way to manipulate which request handlers are selected for a message
  + The selection algorithm would operate on your user ID and the timeframe key for the next timeframe
  + This would have the added benefit that the network would always have a way to contact a given user
  + If a message is destined for a particular user, it would be sent to its current controller group, who would push the message directly to the user
  + Thus a node or user could send a message to an arbitrary user using close to the shortest path to that user, without having any idea of that user’s location in physical space or on the network
  + Even if every user used a single remailer, the average length of a path between two arbitrary users would be 3/4 earth-C
    - That’s 1/4 earth-C between each user and its remailer, and ¼ earth-C between the two remailers with a request handler between them.
    - That means a theoretical minimum latency between arbitrary users of 100ms for the average case
    - The path length for the worst case scenario would be 3/2 earth-C: ½ earth-C between each user and its remailer and ½ earth-C between the remailers
    - That means a theoretical minimum latency of initial contact between arbitrary highly location private users of 200ms in the worst case scenario
  + With this kind of performance, you could shout at a person in a virtual world and have them respond as quickly as they could in real life, yet you would have no way of knowing anything about that person’s real world identity or location, even if you were the National Security Agency.
  + In Daemon, John (I think that was his name) had to use many zombie machines all over the world to conceal his location temporarily from the NSA while he communicated with them through Sobol’s game.
    - John’s latency in that scenario would have been enormous
    - Every message to and from John would have to zigzag all over the globe to each of his many zombies, and every connection between zombies would be via the internet which is far from efficient at taking the shortest route between nodes, especially for nodes in obscure places, which is where he would have to put his zombies to keep them safe from the NSA for as long as possible.
    - Zutha’s distributed architecture would provide far longer lasting and higher quality location privacy at a tiny fraction of the latency
* If Zutha did crumble completely, it is very likely that a lot of money holders would stop paying out their obligation to Zutha long before Zutha’s economy reached zero
  + This is because the reputational collateral which underlies the loan to the money holders would crumble with the destruction of the Zutha economy
  + Zutha itself would be able to bear a large part of the losses from its savings if many money holders failed to pay out as Zutha crumbled
  + It is likely that money holders would stop paying before all money owners had extracted their money, though, so the last money holders to get out would bear some of the losses
  + This is really no different from a run on a standard bank, though, and in fact I think a greater proportion of money owners would get their money back in the collapse of the Zutha economy than in a serious run on a bank because of the money Zutha would hold for such an eventuality.
  + Normal banks do not want to hold too much capital in cash form because they don’t get any return on it.
  + Zutha however, gets a small interest rate from a vastly larger stockpile of money than it owns itself. It also has no reason to do anything with the money it owns itself except spend it.
  + Zutha could be mandated to keep a minimum proportion of the size of its economy in savings. All of this money would actually be being put to full use in the world economy, and Zutha would be earning a small interest rate on it by not spending it as well.
  + If Zutha crumbled, all its money handlers would have to sell all their assets in the real world economy which could have disastrous effects for the global economy once Zutha is large enough.
* If a neighbour reports false latency stats when it responds to a discovery beacon, the node that sent the discovery beacon will discover this when it first sends a message through the dishonest neighbour
  + The latency stats a node receives from its neighbours when it first connects are just a starting point. The node will refine these data as it actually communicates with other nodes in the network and receives its own latency stats.

# Mon 7-Feb-11

* Ownership items contribute reputation to no one. They are like decision items.
* You cannot invest in yourself.
* Empty items are ranked by the reputation of their owner.
* You can still create an item that belongs to someone else and invest in it. But you couldn't invest in your own book or comment. You'd have to rely on others finding it and liking it. Others will be more likely to find your new items if you have a high reputation.
* Defensive weapon
  + dragonfly-like UAV which shoots you in the eye with tranquilizer.
  + If you use goggles it sprays paint on them so you can't see. If you take them off to see (which many people will do instinctively) it tranquilizes you by shooting your now exposed eyes. Those who don't take off their goggles are neutralized anyway. Allied human soldiers can come along and capture the blinded enemy soldiers while the UAVs hover nearby in case they remove their blackened goggles.
  + While enemy soldiers are writhing in blindness or to protect their eyes from the tranquillizing missiles, UAVs could disable their weapons by injecting superglue into a critical mechanism.
  + There needs to be something which can defeat these dragonfly UAVs. Ideally they could defeat each other so they can police each other and defeat any smaller force of their own kind.
  + They could have claws to tear each other apart. Two or three could gang up on one victim and rip off its wings. Zutha owned devices should have an easily accessible hardwired disable button so that misbehaving devices can be easily disabled without damage by their peers. This will be necessary to make a distributed system work because vagrants need to be able to be taken down by the majority.
  + The dragonflies' air to air weapons need to be general purpose so they can fight alien UAVs. By the time opponents are that sophisticated, though, battles will be so futuristic I can't imagine them.
* MMORPG
  + Game users are zutha users. To play you just create a new standard zutha user account or use an existing one.
  + You get domain specific reputation in the game which contributes to your universal reputation.
  + You control characters in the game which can give and take reputation from each other in the game. Real world users can invest in character actions and creations and contribute to their game reputation just as other reputation domains are open to outside investment for some items.
  + The user that owns a character inherits its reputation.
  + The user is allowed to access the full zutha database, but the character can only perform actions in the game.
  + A user acts like a spirit which partially controls the mind of a character. The user is like the character's conscience and is responsible for the character's actions even though the character has an independent will and may do things the user does not command. The character will continue to live in the game world as an NPC when the user is not controlling it. The character's personality is developed by the user's influence on its mind over time.
  + There will be no way to prevent trading of a game-specific currency for real money. Thus the game currency should simply be real money.
  + The price of objects found or created in the game will be established by trade.
  + The story to account for why some players are rich without having earned the money in the game world is that the spirit entities bring their wealth with them from their world of origin. In the game the world of origin for the spirits will be a mysterious place. The allusion will be that the spirits come from the real world and enter the game to control characters there.
  + Characters can die, but this should be a rare occurrence. Powerful magic may bring a character back to life but this will be rare. Instead, the game engine should be designed to make user controlled players lucky like the heroes in stories. Arrows will miss them or not be fatal; NPCs will appear to save them at opportune moments.
  + If two user controlled players duel to the death it will be a serious event and one of the characters will die permanently.
  + When a character dies the spirit must find a new host. The spirit should be able to move between hosts anyway so that a user can have multiple characters. When a character dies they can switch to another of their characters or enter the mind of a new NPC that is sufficiently weak.
  + Users/spirits can identify other spirits regardless of their host. Your zutha identity is thus persistent in the game. You can create a new anonymous user if you want though.
  + Trade of objects or services in the game for real money is taxed by zutha. Players could arrange the transfer of money outside the game and donate the object for free within the game, but it would be much more convenient to use zutha dollars which can be transferred instantly using the game's native interface. This would also allow anonymity to be maintained whereas real world transfer of money would jeopardise this. The tax should be small so that there is little incentive to bother avoiding it.
  + Users can purchase some items from NPCs and the whole cost goes to zutha. The prices should be found by auction. The NPCs will act as merchants trying to maximize profits. This supply of NPC provided objects should be finite. The more abundant an object is the cheaper it will become.
  + The supply chain of goods in the game can involve both human and computer controlled characters.
  + Reputation is the more valuable resource in the game (than money) as it must be earned entirely inside the game.
  + Zutha - via in game NPCs - will grant users reputation for completing quests and doing various good deeds. Users can lose reputation for bad deeds like killing children. Players will give and take reputation from each other using zuth. Users with a lot of zuth will carry this into the game world, but they must build up their influence by earning game specific reputation - of which there might be multiple subtypes.
  + Magic is a general purpose power which can do practically anything but costs the energy required to perform that action in mana. People can write algorithms to manipulate the game API and do cool stuff. The game physics engine calculates the energy required and that is the cost in mana. There is no conservation of energy in this fantasy world. Instead energy accumulates in magical players in the form of mana.
  + People can sell spells (algorithms that manipulate physics using the game API) to each other or give them away, perhaps in exchange for reputation. You could probably make a lot of money designing spells.
  + Spells would include charms and curses that respond to arbitrary curses. You could make a spell which crushes someone's heart - costing little energy. A defensive charm could detect actions that affect a character's body and oppose them. In a power struggle between such an attacker and defender the character with access to more mana/energy would win.
  + Players could team up to pool their mana and provide it to their beat magical user to control.
  + There would be magical objects - some made by players and some existing artefacts - perhaps some that contain large reserves of mana, or perhaps some that have infinite mana available for a very specialized behaviour.
  + If you were cursed you could read the code of the spell to work out what it will do. A spell with more code should take longer to cast so the defender has more time to read what it will do.
  + Spells should be written in a language dedicated to the task. The language might even be designed to look mysterious and aesthetically pleasing rather than just being a purely functional programming language.
  + Becoming a skilled magical user would involve both the collection of ingenious spells designed by others, and an adeptness with using the magical language yourself so you can figure out what your opponent is doing while they cast.
  + The zutha community would develop a database of known spells which would be identified by their code. Spells would have descriptions and suggested counterspells. This information would appear in game if available.
  + A player could obfuscate their code to make it harder for their opponents to understand what they ate doing.
  + It should take a lot of effort to create a new spell. It should not be easy for a player to create a massive collection of powerful spells with multiple versions of each one so that their opponent won't find them in the zutha database. It should require varying levels of magical reputation to include particular features in a spell. It should also take a long time to transfer a spell to another player. You might have to "teach" a player a long and powerful spell over the course of a few days.
  + A player's mana replenishment rate and capacity would increase with experience bit would have a random component - some players would be naturally better than others at magic, just as with other skills. You could also improve your magical adeptness with rare artifacts and substances including potions made by players with potion making skill and rare and expensive (in real money) materials.
  + A certain amount of gameplay should be free, but you should have to pay an increasing rate per hour as the number of hours you play increases. This would be an attempt to discourage people playing too much but it probably wouldn't work because you could probably earn enough money in the game to pay your expenses. Perhaps there should be a maximum play time per day per user. You could have multiple users, but I think a limit per user would still have some effect because people will get attached to developing a single user I think.
  + Spirits (users) should carry the "software" part of the characters they develop in the game, but the characters' bodies should hold the hardware.
  + If a character dies, you lose his physical attributes like strength, speed and endurance; his physical skills like shooting accuracy and swordsmanship; his mental skills like speed of learning skills and general intelligence; and his magical skills like mana capacity.
  + The spirit however retains knowledge such as of the game map and of spells, as well as relationships and reputation with other spirits (but not NPCs who aren't aware of the spirit world).
  + A spirit also gains knowledge of the skills it learns while possessing a character. A spirit can teach its host skills it has learned while possessing another host, but this takes time. The better the spirit knows a skill, the faster it can impart the proficiency to its host.
  + Obviously the spirit retains its wealth because the game currency is real money measured in zutha dollars.
  + If the user hid items anywhere he can return to find them with a different character.
  + The spirit/user can also entrust items to entities (like other users/spirits) that have a relationship with the user/spirit rather than his character. Long term deals and services can thus be provided to players regardless of which character they use so such deals won't be affected by the death of a character.
  + Spirits develop their potency and power over time, increasing their ability to possess ever more powerful characters and increasing the strength of their influence over the characters they possess. Eventually a spirit might develop the power to possess such things as dragons or even conscious trees. It should be harder to possess more alien things and the skill of possessing alien beings should be separate to the ability to possess powerful things. Possessing a powerful tree would require substantial proficiency in both talents.
  + Provide an API to manipulate all aspects of the game (subject to the capabilities of your character and spirit). Encourage people to build programs to drive character behavior both while the user is driving it and when the character is acting alone while the user is absent. People could sell programs and earn money through zutha for building a program that gains a high reputation. You could probably sell a controller program for a while if you sold it secretly and convinced your buyers that it is worth their while keeping it from others for competitive advantage. Eventually the program will leak to the public domain though and you'll have to make your money by publishing it on zutha. You should probably publish it on zutha in encrypted form when you first create it so you can prove you made it once it leaks to the public.
  + If you stole goods from a character in the game you would be stealing real monetary value.
  + The currency for trade with npcs should be gold. You can trade gold or any other commodity with a player controlled character in exchange for zutha dollars.
  + You can trade gold at an npc controlled bank for zutha dollars.
  + Gold should be a limited resource in the game. You can mine more of it and its value relative to the zutha dollar would fall. But demand for gold will increase as the game economy develops which will cause its value relative to the zutha dollar to increase. You could earn gold by selling goods or services to npcs. Zutha would ensure that it always sells more goods and services than it buys. An npc might buy iron ore mined by a player, make steel from it and then sell the steel for a profit at the market price to another player. The fact that npcs work for free would make this very profitable for zutha.
  + If you commit a crime in the game you can be fined. If the offending character does not have enough in-game gold to pay the fine, it could be imposed on the spirit if the spirit is responsible. The spirit could be fined directly in zutha dollars which could be taken automatically from the player's zutha account. If a user withdrew all his zutha dollars and refused to pay a fine he could receive additional reputational punishment. He would also still owe the fine, which could deducted slowly from any income he makes from the game or zutha more broadly.
  + A character could be punished by npcs or players for crimes like murder and theft by being locked up. If a spirit was to commit a crime against another spirit, the player community could vote to punish the spirit by fining it, or reducing its privileges even up to banning it from the game temporarily or permanently. Of course the user would also lose reputation, but may gain it back from other users which supported the crime. The spirits with legal control over the place the crime was committed would have rights to impose punishments for crimes comitted under their jurisdiction. They might capture the spirit - thus blocking the player from playing.
  + A spirit might go to an alien land and be captured. The spirit's allies might come and rescue him or put pressure on the foreign power to release him.
  + Upgrades to the game would happen just like upgrades to zutha and the Zuthanet. The game developer community would develop features and vote on the ones to include in the next release. Node owners would have to upgrade their nodes' software voluntarily.
  + Game clients would just be any device that knows how to send and receive game messages. You could read every message manually in the standard zutha client if you wanted. People will develop client software of their own and possibly try to sell it, and client manufacturers will innovate on the hardware side. The hardware will likely be very general purpose such as a PC or HUD device. Zutha client software will probably include inbuilt support for the most popular applications, of which the MMORPG will probably be one. Of course there could be many distributed games developed on the Zuthanet platform each with their own api and collection of both official and user-developed client software.
  + When a spirit possesses a character it puts its mark on it. The more time the spirit spends in control of the character the more strongly the character becomes bonded to it. A powerful spirit can force itself into a character that is bonded to another spirit and slowly turn the character to its will. A very powerful spirit could even push out a weaker spirit from its host and take its place.
  + Stealing a player's character in this way would be considered a serious crime and would incur significant reputational costs. A spirit that commits such a crime might be hinted down and imprisoned by other spirits - if they can catch him.
  + A player could go down a path of evil with their spirit, building reputation with evil entities and societies and losing reputation with the beings they harm.
  + Once a spirit took a path of evil they would be no turning back. You would open doors of great power by dealing with the evil beings of the world. You would learn dark magic which requires dark crimes to be committed. Taking this path would block you from accessing good magic; even if you knew the code of the spells you would not be able to perform them.
  + NPCs will engage in bartering and automatically learn the market price of the things they sell over time
* Nodes can be marked as "suspected single controller". Actually all hosts are associated with a user, so if the users are marked as single controller, their nodes will inherit this. No more than one node per dependent group is allowed in a node controller group. In a dispute between nodes, each dependent group of nodes gets the weighting of their highest reputation member.
* Some kinds of messages should only be distributed to the network in batches.
  + For example the letter by letter actions of users working on a shared document would be accumulated for a period of time by those users' current request handlers, and then sent to the whole network as a single message containing all the edit actions a over period of time.
  + A game region controller group would only share game messages in real time among the nodes and clients that need to know. They would send out a batch of game event messages to the whole network at regular intervals. The messages in the batch would still be encrypted with the public keys of the nodes who received the messages in real time. A game client could run a full replay of any period of gameplay in a player's history by reading this data from the zutha database. It would only be able to read the data that is encrypted with its public key - that is, the data it received during gameplay. Other players would only be able to see the history of their own experiences in the game world.
* Zutha will make it easy to avoid paying taxes. A government might be able to demand that a bank identify the owner if an account suspected of money laundering but they would have to demand access to the accounts of every zutha user in order to identify which ones should he paying them tax.
* Zutha could declare itself a sovereign nation so earners of money within its realm could legitimately claim they pay tax to zutha instead of the country they physically live in.
* Zutha could put a small tax on every transaction in zutha dollars. If it is small enough it wouldn't be worth the inconvenience and lack of privacy to pay someone via external services. It need not be called a tax but simply a fee for using zutha services. It is important that zutha be seen to only make money through free transactions of value for value, rather than making money by "taxing" people forcibly the way a government does.
* When nodes reply to a message, they should include any messages they received very recently which affect the way they replied. This way receivers of conflicting messages from different nodes will know which node is more up to date, because it will include more accessory messages.
* How to defend against rockets and suicide bombers?
* World "leaders", as we call our presidents and prime ministers, do not really lead; they just take actions with varying degrees of power behind them. Zutha will allow the community to find people worth following and make them their leaders in a real sense. A high reputation user's greatest power in zutha will not be the power of his vote; it will come from his ability to influence others' opinions and actions by the respect he earns from the community.
* External data that varies unpredictably, like exchange rates, can be determined on zutha with prediction markets.
  + For example there could be prediction markets for the exchange rate of every major currency with the zutha dollar
  + To make things easier for investors, there could be a calculator that takes as inputs the exchange rates of the target currency with all the currencies that are part of the currency bundle that defines the Zutha dollar, and outputs the exchange rate of the target currency with the Zutha dollar
  + You would profit in this market by investing as early as possible in the new exchange rate as soon as it changes, or by predicting the direction that exchange rate will move and investing in that rate in advance.
  + Those that sold late when the correct exchange rate moved away from the value they had currently invested in would lose money because others would sell out from under them and reduce the value of their investment.
  + Zutha’s official exchange rate with this target currency would be the median of the investment distribution in this prediction market
  + I think the output of this prediction market would be the more accurate end of the current range of exchange rates published by various institutions.
  + It is also possible that Zutha’s published exchange rate would slightly anticipate the actual exchange rate according to official institutions. This would mean Zutha’s reported exchange rate would be slightly wrong.
  + Let’s consider what opportunities this might pose for arbitragers and the possible costs or benefits for Zutha
    - Let’s imagine that 1 Zutha Dollar is currently worth 1 Euro and 2 US dollars
    - Some new information reaches Zutha investors and they collectively predict that the US to Zutha exchange rate will soon rise because the US dollar will become weaker relative to world markets.
    - The US exchange rate on Zutha moves to 2.10 US dollars per Zutha Dollar, which means 2.10 US dollars per Euro as well
    - The US to Euro exchange rate according to official institutions is still 2 US dollars to 1 Euro
    - An arbitrager could now purchase 1 Zutha Dollar with 1 Euro, withdraw the Zutha dollar as 2.10 US dollars, and then exchange 2.10 US dollars for 1.05 Euros via an external institution.
    - The arbitrager appears to have made a 5% return with zero risk because this transaction could happen instantly
    - Let’s say the official US to Euro exchange rate now moves to 2.10 US dollars per Euro
    - The arbitrager has 1.05 Euros which is worth 2.10 US dollars.
    - The arbitrager has made a profit in terms of Euros, but his purchasing power in the US market has not changed
    - Zutha gained 1 Euro and lost 2.10 US dollars, which are now equal amounts, so Zutha comes out even
    - The Zutha money handler who received the 1 Euro also acquired a debt to Zutha of 1 Zutha dollar, so he comes out even
    - The Zutha money handler who paid out the 2.10 US dollars reduced his debt to Zutha by 1 Zutha dollar. Initially this was a loss, but once the official US exchange rate lifted to 2.10 US dollars per Euro (and thus per Zutha dollar), this money handler’s loss of 2.10 US dollars equals his gain (by reduction of debt) of 1 Zutha Dollar so he comes out even.
    - The external money trader took 2.10 US dollars from the arbitrager and gave him 1.05 Euros in return. He makes a loss because the 2.10 US dollars he gained is now only worth 1 Euro.
    - So Zutha comes out neutral and allows the arbitrager to profit risk free at the external money trader’s expense. Zutha is basically offering a free service to take on risk for arbitragers who agree with Zutha’s judgement about future exchange rates.
    - Zutha is likely to receive a lot of Euros and lose a lot of US dollars while its Exchange rates don’t match those of the actual market
    - What happens if the US exchange rate weakens further?
    - Let’s say the US dollar weakens to 2.20 US dollars per Euro and Zutha’s reported exchange rate matches its movement rather than preceding it this time
    - Zutha will be providing no arbitrage opportunities so its distribution of currencies should not change over this time as a result
      * Actually Zutha’s distribution of currencies will change if the market in aggregate is predicting the weakening of the US dollar
      * If this happens Zutha will receive lots of US dollars and pay out Euros to people wanting to get ride of their US dollars before their value weakens.
    - A zutha money handler who receives $2.10 US dollars in exchange for a reduction in his debt to Zutha of 1 Zutha dollar would be worse off once the US exchange rate rose to $2.20 US dollars per Zutha dollar. If this money handler was one of those who predicted the weakening of the US dollar, though, he could immediately trade this $2.10 US dollars for 1 Euro and come out even.
    - Regardless of what happens in later periods, Zutha gained by anticipating the weakening of the US exchange rate in the first period because it will end up with more Euros relative to US dollars than it otherwise would have.
  + Talking about “Zutha” gaining or losing from currency trades doesn’t really have a sensible meaning.
    - There is no single entity that gains or loses purchasing power as a result of the exchanges of currency that occur through Zutha
    - Instead, the gains and losses or born distributedly by all Zutha’s money handlers
    - Each money handler is independently responsible for managing his portfolio of currencies to maximise profits
    - I think this distributed management of Zutha’s money will actually lead to very efficient allocation of currencies to maximise the aggregate profits of the money handlers.
  + The entity called Zutha which can more appropriately be described as gaining or losing is the collectively owned user called Zutha which owns Zutha dollars
  + The Zutha entity that owns Zutha Dollars – the Zutha Government as it were – does not trade currencies but keeps all its money in Zutha Dollars, so the only way it gains or losses it experiences as a result of exchange rate changes are due to the changing value of the Zutha Dollar relative to the rest of the world economy.
  + When a user buys 1 Zutha Dollar with 2 US dollars, the Zutha government is not involved and does not participate in gains or losses due to subsequent changes in exchange rates.
    - If the value of the US dollar subsequently falls relative to the Zutha Dollar, the purchasing power of the user’s Zutha Dollar in the US increases
    - The money manager who took the user’s US dollar will suffer if he keeps it because his additional debt of 1 Zutha Dollar to Zutha does not change when the value of the US dollar falls.
  + It is likely that Zutha’s prediction of the future exchange rate will match the aggregate prediction of the market such that no net flow of currencies occurs through Zutha.
    - This will benefit Zutha’s money handlers in aggregate because they will not be forced to accept trades that are undesirable according to future exchange rates
    - Individual money handlers will be forced to make such trades on occasion, but on average a money handler will participate equally on the good and bad side of trades and so the net effect will be neutral for every money handler.
    - If the net flow of currencies forced upon a money handler is zero, he will be completely free to trade currencies as he sees fit to maximise his profits, whereas if Zutha was forcing its money handlers to make net trades of one currency for another it might be an opposing force to the distribution of currencies they want to hold.
    - If Zutha imposes no net currency flow on a money handler then the money handler will be no better or worse off in the currency trading market than he would have been without being a Zutha money handler
  + If a new country joins the Zutha network this may cause a net inflow of their currency to Zutha money handlers
    - This would be because of cash these new users want to hold in Zutha, which would have been purchased using their native currency
    - The net flow of this currency into Zutha will reduce the supply of that currency in the native country, thus the Zutha money handler will easily be able to trade this currency – probably for a profit – because demand for it will have increased relative to supply.
* Currency Exchange revisited
  + By definition the equilibrium exchange rate is the rate such that demand for a currency equals supply which means there will be no net flow of that currency
  + Presumably the world’s official exchange rate will reasonably accurately track the equilibrium exchange rate
  + If Zutha is able to predict the equilibrium exchange rate in advance it should be able to make a profit for its money handlers
  + The fact that Zutha uses a prediction market to determine its exchange rate rather than finding the current equilibrium of the market might mean that it does predict exchange rates in advance of the world market
  + The reason Zutha didn’t make a profit from the scenario in which the US exchange rate rises from 2 to 2.10 is because it didn’t sell US dollars for more than their real value despite the fact that the market was exaggerating their value.
  + If Zutha was to keep its exchange rate for US dollars slightly higher than the market price as it rose from 2 to 2.10, it would gain Euros (according to our example) for less than their future worth, every step of the way. Once the exchange rate reached 2.10 Zutha would have accumulated a lot of extra Euros which are now worth 2.10 US dollars each, and it would have paid less than 2.10 US dollars for each one.
  + If the prediction market used by Zutha to select its exchange rates were comprised of a large group of users who mechanically invest in the current exchange rate and a small group of geniuses who in aggregate predict the turning point of the exchange rate from its current trajectory, then the median exchange rate determined by the prediction market would lie slightly closer to the future exchange rate than the current exchange rate.
  + In the scenario of the US dollar weakening, this would take the form of Zutha setting its exchange rate very slightly higher than the world official exchange rate until the US exchange rate reaches 2.10.
  + This would be the ideal rate for Zutha to select since it would allow Zutha to purchase large quantities of Euros for what it knows is less than their real value in US dollars.
  + Even a small difference in Zutha’s exchange rate compared to the world exchange rate would encourage arbitragers to trade large quantities of money through Zutha because they would be making free profit.
  + The arbitrager who trades 1 euro for 1.05 euros is making 100% pure profit with zero risk. Zutha is essentially making its profit from the external currency trader which has inferior information to Zutha.
  + Is there any reason to believe that Zutha’s prediction market would do a better job at predicting future exchange rates than the world market, which is really just a huge prediction market to do just that?
    - What would it mean if the currency market itself was the best possible prediction engine for its own prices?
      * I think this would mean that all available information is captured in current prices such that there is precisely even odds of the next price move being in either direction.
    - Are there any reasons to believe the world market is flawed as a prediction engine of its own prices and that it would be possible to do better?
      * I can’t think of anything about the system itself that is flawed
      * Of course, it is obviously not a perfect prediction engine because there will always be relevant information that no one, or not enough traders, know.
    - Is there any reason to think that my prediction market system would do better?
      * Perhaps that a larger proportion of the market will be engaged in predicting than in the world currency market?
      * In the currency market, if a big event occurs that affects exchange rates, such as the recent flooding and hurricane in Queensland, the biggest force affecting exchange rates is the reduction in demand for Australian Dollars caused by reduced exports and the fall in investor confidence about the Australian Economy.
      * There will have been arbitragers that predicted the weakening of the Australian Dollar in advance of its actual fall and they would have been selling Australian Dollars as fast as they could. This would have quickened the weakening of the Australian Dollar but the market power of the arbitragers would be small compared to the market power of the bulk of currency traders who are not trading, not in an attempt to predict future exchange rates, but for practical purposes such as buying Australian goods and investing in the Australian economy.
      * In a prediction market, however, all the market power lies with the predictors so the output of the market will exactly track the aggregate prediction of future exchange rates.
      * What about my prediction market specifically?
      * In a combined double auction you can only predict the exchange rate at a specific moment in the future
      * In a pari-mutual market you can similarly only predict the exchange rate at a specific moment in the future and you would only have accurate data right before the market closes.
      * In my prediction market I think you would get a continuous prediction of the direction the exchange rate will move next, with the gap between the prediction market output and the current exchange rate being an indication of the rate at which the exchange rate is likely to move in that direction.
      * There is still a question of whether my prediction market will produce useful results despite not being anchored to anything real.
      * Eugene mentioned a version of the pari-mutual prediction market he has been working on which allows continuous predictions by simulating an infinite number of markets each predicting the outcome at a different moment in time.
      * This could be done with a 2 dimensional prediction market in Zutha; one dimension for exchange rate and one for moment in time.
        + A judge would specify at some reasonably fine resolution the trajectory the exchange rate actually took and winnings would be divided up accordingly
      * I’m not sure how this type of prediction market could be used to select Zutha’s exchange rate though. It could be selected as the predicted exchange rate at a fixed amount of time into the future, but how far?
      * I think my first idea would actually work better for actually selecting Zutha’s exchange rate. The 2-dimensional prediction market for predicting the exchange rate at every moment in time into the future would be a very useful source of information for investors in the decision market for selecting Zutha’s exchange rate.
      * Yes, a better way to think about this market would be as a decision market.
      * It would be in the interests of high reputation Zutha users to select an exchange rate which is most profitable for Zutha. The users with this motive would then drive the behaviour of profit investors because they would determine the underlying meaning of the market; the anchor to some external metric as it were. As with other Zutha investment markets, the profit investors would predict and magnify the behaviour of the independent investors – the investors who invest with a different purpose than to make profit, such as to praise an item or sway a decision towards the outcome they believe would be best.
  + Most currency traders have a spread between their buy and sell price for a given currency
    - This spread could be interpreted as a representation about their uncertainty about the current exchange rate
    - It could be a kind of insurance policy that protects them against sudden changes in the exchange rate which would leave them stuck with currency they paid more for than its new market price.
    - It could purely be a source of revenue
    - Or it could be a combination of all of the above
  + Would Zutha benefit by not having a spread between its buy and sell price if it could very accurately predict future exchange rates?
    - The one benefit I can think of is it would increase the flow of currency through Zutha because it would be cheaper to trade currencies through Zutha than through other currency traders
    - If Zutha predicted future exchange rates reliably, it would profit from this increased throughput because its rates would be picked to cause flow of currency in the direction that would lead to profit for Zutha
* Persistent Controller groups should be assigned to manage categories of actions with a strong likelihood of interfering with each other
  + For example, each item should have its own controller group
  + If two investments in an item happened very close together, their correct order could be established more quickly and with less wasteful messages if the one controller group was handling all the investments in that item
  + Similarly, a single controller group will receive conflicting edit requests with less latency and be able to collectively decide on the correct outcome without involving the entire network
  + Each user should still be assigned its own controller group so that it can have messages pushed to it directly
  + A user’s controller group would be responsible for managing item creations by that user. The same group would also manage changes to the user’s permissions so it would know quickly about conflicts involving a change in user permissions that prevents it creating an item.
  + Actions could involve multiple controller groups
    - The controller group for an item edit request would be the union of the controller group assigned to the item and the controller group assigned to the user
    - This way the two controller groups could immediately share recent information that concerns the success of the action such as a change in the user’s permissions or a change in the permission level required to edit the item.
  + All controller groups should roll over some members each timeframe
  + The MMORPG should work in the same way
    - Actually it already does to some degree: the overlapping parts of controller regions are shared by both controller groups
    - The concept of merging controller groups could be extended to add controller groups for such things as users, characters, NPCs, guilds etc.
    - If a user had its own controller group then it would be contactable by someone who doesn’t know what region the user is in
    - If the user was always controlled by the controller group of the region it is in, then if you could know what a user’s current controller group is you would also know that user’s location. This would not be desirable since a user’s location should be secret to most users in the game world.
    - Actions that only concern the user itself, such as remote communication from another user, would be managed only by the user’s own controller group.
    - Actions that concern the user in the context of the user’s location would be managed by both the user’s controller group and the region’s controller group.
    - If you were interacting with a user in a way that involved the user’s location, such as by trading items with the user, then you would know both the user you are trading with and the region you are both in, so you could find the appropriate controller groups to send these messages to.
* Commodities and even arbitrary goods could be added to the zutha dollar bundle at any time.

# Tue 8-Feb-11

* Private caretaker rights to natural resources
* Allocation to Money handlers based Credit rating, loyalty - both types of reputation
* Every item is a reputation domain.
* Index quantity of zuth to quantity of zutha dollars. This amount will grow with influx of users and popularity of zutha but will be indexed to a goods bundle, avoiding inflation.
* Inflation of zuth would harm investors because you would need to make higher returns to overcome inflation. It would benefit reputation holders because salaries would be higher relative to what investors can get. So less inflation would encourage more investment and less attempts to invest in ones own reputation. This would be good.
* Vase steepness represents sensitivity of price to demand for investment in an item.
* When forced to sell investment due to fall in permission, sell price is minimum of current price and purchase price. So it is dangerous to invest to your maximum allowed capacity.
* Call the height of the vase - the commodity you buy when you invest - zest.
* Restrict the rate at which you are allowed to sell zest. Thus you can't get out of the market quickly.
* This would make incentive to invest with many different users.
* Impose item-wide sell rate restriction. One large investor could fill this rate and block others from selling.
* Divide sell rate evenly among current sellers.
* Measure item value in zest. Ranking the same as ranking by worth, but scale can now be linear. Can't do this with arbitrary price functions. So abandon this idea.
* Item worth equivalent to market capitalization.
* Reputation still linked to item worth.
* Item owners can specify arbitrary price function.
* Zest represents partial ownership if an item.
* Decision items have no real money bucket. Actually they might as well but only zuth counts towards decision.
* Price function must be fixed. If it could change it must be an increasing function and all investment would have to be reallocated. The owner would still have to much manipulation power. No. Price function should be fixed at item creation.
* Item owner specifies proportion of ownership to share with zuth investors. Income from money investors shared among all owners.
* Rename h (vase height, zest) as z.
* Multidimensional decision market with multi-variable restriction rules for allocating ownership.
* Call the zutha mmorpg zuthaverse - at least for now, so it has a name.
* Continuous investment markets have a thin layer of zuth spread evenly across all dimensions. This way the median will always be able to slide continuously, even if the distribution is one of spikes, mountains and empty plains. The thickness of this layer would need to scale with the size of the market. Thickness should be such that the hypervolume of the film over the region which contains investment is equal to the quantity of zuth invested in the market. But this would allow a tiny investment to have a massive effect by expanding the size of the region which contains investment.
* The thin film concept is too contrived anyway.
* If there was a reward for having an investment where the equilibrium outcome lies, then someone would always invest in open space if it would bring the equilibrium on top if their investment.
* The owners of the decision item could be the investors in the current winning outcome. They would earn reputation from the total zuth invested in the decision item. This would increase the attractiveness of investing in the consensus as the size of the market increases.
* Users could invest in their own reputation with a decision market. The reputation they gain because of their own investment in the decision item could be subtracted.
* Self-given reputation could be subtracted from any kind of item allowing item owners to invest in their own items.
* Ownership items should allow some investment from universal reputation but give most influence to the owners themselves.
* 2 owners would be unstable. An owner that unfairly steals ownership of an item from his co-owner could be punished outside that ownership item both by the co-owner and by the whole zutha community.
* Investors are always part owners of an item. Being a group they are less likely to be single minded so they could settle ownership disputes between 2 owners.
* Maximum ownership of an item by investors should be restricted to something like 20%. This should also be the default. Owners can reduce the proportion of ownership by investors if they want. They have an incentive not to because this would reduce investment in their item.
* Investing should take time just like selling. Total Max rate and per user max rate determined by owners.
* Decision items cannot have their parameters (like price function, max flow rate) modified individually. These parameters are determined globally for each type of decision item.
* Current "winners" of a decision item are beneficiaries rather than owners of the item. They earn money from it but not reputation. Money comes from zutha in proportion to decision item worth/reputation.
* Reputation equals worth plus sum of reputation of owned items.
* Where reputation equals worth only worth is displayed.
* I don't think it would be such a good idea to let item owners pick parameters like price function and max flow rate. These would be better determined at a global level and apply to all items.
* Changes to the price function would only apply to new items.
* Every item earns and accumulates money. Every payday it distributes its current balance among its owners and beneficiaries as a dividend. The amount a user earned due to his own investment in the item is subtracted from this dividend and thus remains in the item's account until next payday. This is done by multiplying a user's dividend by the proportion of item worth contributed by others.
* Every item has a different payday, simulating independent companies. This will also spread out the computation costs of calculating dividends.
* Item owners collectively choose how much of money donations to donate back.
* Microwave radiation powered, electrically driven aircraft.
* Microwave beam redirection power distribution satellites.
* Solar orbit solar power stations.
* Beam energy from nuclear power stations up to power distribution satellites.
* Accelerometer location tracking with occasional confirmation or correction via GPS and wifi.
* GPS style time delay triangulation using local wireless timekeeping devices which know their exact location.
* Construct 3d real-time virtual parallel world. Bidirectional mapping between virtual parallel world and z-space. Real people appear in z-space and virtual parallel world, and virtual beings appear in both dimensions too. You could meet a real person at their real location as a virtual avatar.
* Zuthanet could serve the entire internet. It could actually speed things up by finding the quickest path between client and server. It would also provide effective translation between ipv4 and ipv6, which zutha nodes would use.
* All descendants of an item are in its reputation domain. Multiple association types would be considered parent-child. Examples:
  + Item-comment
  + Author-book
  + Category-item
  + Container-contained
  + Type-instance
  + Owner-owned
  + Book-character
  + Product-review
  + Package - code object (class, interface, object, trait etc.)
  + Source file - commit
  + Question-answer
  + Category-subcategory
* You can invest more than your quota in an item bur the extra dies not contribute to item reputation.
  + Ranking metric is reputation. Reputation can be more or less than worth; less if some investors have exceeded their quota; more if the item owns items.
  + The quota on contributing to an item's reputation includes contributions via investment in owned items (recursively).
* Fix investor ownership at 50%. This way no owner ever has absolute control over determining ownership distribution. Investors will be less likely to act unanimously, but they will have more zuth at their disposal. If a single owner has enough zuth to fill his quota, he can likely prevent the investors from adding another owner because they will be unlikely to all invest to their quota. Investors can increase the stakes by investing more, increasing everyone's quota and eventually breaking the stubborn owner's bank.
* You get reputation with an item by owning descendants of the item that receive investment.
* An item has a set of ancestors which are the reputation domains it resides in. You get an investment quota of 1% of your reputation in each domain, additively. So if you have x reputation in the immediate parent item P and that parent item has 2 ancestors then you get to invest 3% of x rather than just 1% if P had no ancestors.
* You are likely to have far more reputation in the higher domains so with this scheme the power will still mainly lie with the highest reputation users in the broader domains.
* Different item types may have different distributions of investment quota by reputation domain.
* There is an absolute maximum investment cap of 10% of universal reputation.
* An alternative quota distribution scheme could be that the quota of the highest level domains is 1% and the quota of a child domain is twice that of the parent domain.
* Or narrowest domain could have quota of 100% and each parent gets half the quota of its child.
* If a domain is reached via multiple children the shortest path should be used to determine the quota of the parent.
* Each parent-child association type could have a specified degree of abstraction which determines the fraction of the child's quota that the parent gets.

# Wed 9-Feb-11

* Call z zest again. Call the height of a real money investment vase "merit".
* Merit, zest, and reputation ate 3 separate comparison metrics for an item. This will require all items to have the same price function.
* Negative investors get reputation from other negative investment. They are also 25% owners of the item and can vote on ownership. They only earn money from negative money donations. They decide how much of negative "donations" to donate back.
* Negative investors share half the negative reputation of an item with the owners except they get it as positive reputation.
* Positive investors share half the positive reputation with the owners.
* Only the owners are affected by both positive and negative investment.
* Perhaps reputation quota for investment in the ownership decision item should be calculated the same as every other item: 100% of your reputation in the item itself and less for each parent domain according to the degree of abstraction of the parent-child association. This would significantly hinder a user contributing to the ownership decision item of an item he owns which gives him net negative reputation. This is good because otherwise a user could escape from criticism by disowning an item that gives him negative reputation.
* 50% of item reputation is divided among investors by ownership of zest.
* Investor stake need not be 50% anymore but that should be the default. What about negative investors? Negative stake equals positive stake. Actually the default investor stake should be decided by decision market. Max investor stake is 50%.
* It is good that the workings of reputation and dividend allocation are complex because then the majority of users won't understand them, but can still benefit by following their natural instincts, which is what I am trying to design the complex allocation system to reward. The more investors follow their natural instincts the mote grounded the market will be in something tangible, whereas if too many investors are trying to understand the system to take advantage if it, the self-referential nature if the investment market will show its head in the form of increased arbitrariness.
* Worth is the amount of investment of zuth in an item that counts. It excludes investment over and above a user's quota. Net worth is the sum of positive and negative worth.
* Zutha pays items according to their worth.
* "All items" is a beneficiary of the zutha item, which receives all zutha income from taxes, direct donations, advertising subscriptions etc.
* Beneficiary groups can have income divided up evenly, by worth, by reputation, or some custom formula.
* Item income divided between beneficiaries and owners first, then among beneficiaries and owners (which include investors).
* New model
  + Zutha donates to users and specific projects according to reputation.
  + An item receives revenue from the retained part of external positive money donations.
  + Each item has its own dividend payment policy which determines what proportion of revenue is paid to zuth investors.
  + Items receive "benefits" from internal donations through the beneficiaries system. Benefits do not fall under revenue.
  + Negative money donations are paid to zutha. The proportion retained by zutha is the same as the proportion of positive donations retained by the item.
  + Zutha pays dividends to negative zuth investors at the same rate as positive dividends.
  + An item's net income is revenue + benefits - dividend payments.
  + An item pays a proportion of its income to beneficiaries and the rest to contributors.
  + All of an item's worth contributes to the reputation of contributors - subject to restrictions on maximum contributions of reputation by each investor.
* "All users" is an item which is owned by all users where ownership is allocated by universal reputation.
* All Users is a beneficiary of zutha.
* All Users divides up its benefit from zutha among its owners as salary.
* All Philanthropic Projects is an item which defines its beneficiaries as a query for items that are philanthropic projects and weights them according to universal reputation.
* Money handlers could issue zutha dollars in exchange for any good. The meaning of this issued currency is simply a promise to pay back an amount of equal value on demand. This is all money is. Zutha dollar value would be indexed to a bundle so it would not fall in value on increase of supply.
* What causes government issued currency to fall in value when supply increases? A government does not promise to pay, in exchange for a dollar, an amount of equal value to the issue value of the dollar. The promise is rather that "others" will be willing to pay... something - the current market value - in exchange for the dollar. If the dollar is to retain its value, the additional obligation would be spread out over the whole economy. The government takes the value it received for the dollar from the rest of the economy.
* The additional debt that has to be paid by someone is spread out over all existing dollars as a tax that must be paid on buying - or paying off the debt of - a dollar. Like any tax, it is shared by both buyers and sellers. If buyers must pay more for the dollar, sellers will have to sell lower to clear the market. Selling a dollar lower means you get less for the dollar. Thus we have inflation.
* Why does the dollar rise in value with demand (deflation)?
* People have a demand to issue a debt to pay (in concrete goods) later, but they cannot issue their own currency. They must pay off another's debt by buying a dollar before they can issue that debt themselves. If there is excess demand for the ability to issue such debts, then people will be willing to pay a premium to get their hands on the dollars that allow them to do it.
* If people could issue their own debts (currency) then there would be no inflation or deflation of currency - provided people were held accountable for the debts they issue. This is what zutha does with its money handlers.
* If issuers default on their debts, the burden is passed on to other debt issuers to pay their debt for them (this is what a government does when it prints money). Other debt issuers include people who pay off someone else's debt and then reissue the debt to someone else.
* If others will pay of your debt, there needs to be a way to force the initial debt issuer (the one that didn't pay off someone else's debt before issuing their own) to pay off someone else's debt to balance the books. Zutha does this by forcing money handlers to pay their debt in exchange for a zutha dollar which is then destroyed.
* If a money handler defaults, will this cause inflation of the zutha dollar?
  + No. By indexing the currency to a bundle, zutha forces currency buyers (ultimately money handlers) to pay the whole tax caused by the default, and doesn't allow currency sellers to sell for less.
* Zutha could allow users to issue currency up to a limit subject to their reputation as collateral. Zutha would keep track of who has issued how much debt, and would be responsible for ensuring they pay up on demand.
* Since zutha does not know a user's financial means, this would not be a good idea, but if zutha knows the currency issuer has received value equal to the debt he issues, zutha can be more confident of his ability to repay the issued debt. Actually it is always the case that the issuer has received something of equal value to the issued debt - that is what he exchanges the debt (currency) for. That is unless he donates the debt.
* Zutha should restrict the collateral on which new currency/debt can be created. Initially only government issued currency will be given to money handlers as collateral for issuing debt in the form of zutha dollars. Later, such things as gold, oil or even financial assets could be used. A money handler could pay out his debt in other ways than as government issued currency, such as with gold, assets or the provision of services. The latter would pose difficulties because it is not a continuously divisible good.
* If anyone could create their own currency at any time, would people create too much?
* If there was too much supply of these debt assets (currency), then there would be an excess of people wanting to sell these assets and not enough buyers. Normally this would lower their price. Would the indexing to the bundle change this? Only if the index to the bundle had meaning in the form if a guarantee by someone to buy the currency at the value of the bundle. This guarantee would be needed when there is too much currency and there are prospective sellers who can't find a buyer at the indexed price. For zutha this would take the form of a desire for a net outflow of currency from zutha - the selling of zutha dollars.
* Zutha's guarantee would be provided by money handlers who must buy zutha dollars at the bundle price and destroy them if asked to. This fixed priced purchase guarantee is precisely the mechanism that would keep the value of the currency fixed.
* What's crucial is that currency is destroyed when there is too much of it. Need the equilibrium be found by the market deciding how much currency needs to be destroyed? What if the market found the equilibrium by determining how much currency to create, and currency was just destroyed at a constant rate? Zutha could do this by regularly destroying zutha dollars owned or earned by money handlers, taking it off their debt. If demand was still sufficiently high for zutha dollars, the money handlers would soon be asked to issue more currency and their debt would return to its original size.
* The challenge would be getting the rate of money destruction right. It would just have to be fast enough to keep up with a fast reduction in money demand. It shouldn't be too hard to detect inflation and increase the rate of money destruction in response. If there is excess supply of money, people aren't going to want to buy new money from money handlers at the high fixed buy price. Since this will be the only way new money is allowed to be created, the money supply will always shrink at the destruction rate when demand for money is falling.
* The fixed price purchase guarantee would still exist. I just wanted to check whether inflation could be avoided if there was no convenient commodity for people to receive when they sell to money handlers at the indexed price - for example in a world where there is no electronically tradeable good which has persistent value and can be continuously divided - like e-gold or us dollars.
* But why must the commodity be continuous? Money handlers could specialise in buying and selling particular goods in exchange for zutha dollars, such as us dollars, shares in a company, particular virtual assets in zuthaverse, gold, Japanese yen, cars, haircuts, tours, training services, nanobots etc. They would report to zutha which good they have available to sell at any one moment. When someone wants to trade their zutha dollars at the guaranteed price, they could buy any of the wide range of goods available from all money handlers combined. Zutha would automatically match them with a money handler that sells that good. It would also distribute trades evenly among money handlers so the sizes of their debts all rise and fall proportionally with the entire money supply.
* If there were enough money handlers actively engaged in the economy trading zutha dollars, zutha would not need to play matchmaker. Instead of forcing money handlers to make trades at a fixed value of zutha dollars, it could allow them to trade freely in the economy with any goods at any price they like. Every time someone traded with a money manager it could be treated as someone taking zutha up on its guarantee in which case zutha should destroy currency equal to the size of the trade. This would be destroying far more currency than necessary but it wouldn't matter; it only matters that enough is destroyed, since any excess that is destroyed will be recreated due to sustained demand and lack of supply.
* Is there anything special about the amount of currency that would be destroyed under a policy of destroying currency sold using the guarantee? I don't think so. It only matters that the destruction rate is high enough, not how high it is. Thus zutha could maintain the guarantee effectively by simply destroying currency evenly across money handlers at a rate that is obviously higher than necessary. To ensure that money handlers receive sufficient trade to be able to do this they could be required to always value the zutha dollar as defined by the index in specifying their prices. They would also be required to sell new zutha dollars at the indexed price. Zutha would enforce this by calculating the value of the purchase commodity (gold, us dollars etc.) itself and adding the appropriate amount of debt - in the form of issued zutha dollars - to the money handler that receives the purchase commodity. The purchaser would receive the issued zutha dollars.

# Thu 10-Feb-11

* Any user who is a high income earner in zutha dollars could be allowed to be a zutha money handler because they would have a high enough income of zutha dollars to cope with the destruction rate of their zutha dollars associated with being a money handler.
* Money handlers must hold a minimum amount of zutha dollars at all times, subject to the size of their debt, so that zutha can extract interest from and destroy those dollars as necessary. They are unable to voluntarily spend money when their account falls below this minimum, but taxes, fines, interest and other costs can still be extracted from their account. They lose reputation to the degree their balance falls below this minimum - including into the negative.
* Money handlers can now sell global currencies instead of giving money away when zutha demands. They remain accountable to their debt by keeping their balance of zutha dollars replenished. The slow and steady disappearance of one's zutha dollars is likely to be more psychologically palatable than having to pay out large sums of money for nothing in return (except a reduction in debt which is not a very tangible commodity). This will likely keep money handlers more loyal. Also, the requirement that they hold a reserve of zutha dollars would make it less appealing to money handlers to abandon zutha and their debt to it, because they would lose a kind of bond in the form of the zutha dollars they would leave behind in their account and be unable to spend. It would be purely a psychological difference between the minimum holding being a large positive number and zero, because the higher the value of the bond (minimum holdings), the more currency would be needed in the economy and the greater the debt of the money handler would be. But a psychological effect is a real effect; it would make defaulters feel they are losing more than they really are - actually the size of the bond is meaningless because they can never spend it voluntarily so they don't really own it.
* The maximum debt a money handler can have is the minimum of their current holdings of zutha dollars and some multiple of their reputation.
* So currency is backed by 50% collateral. This would not allow expansion of the money supply. Maybe 25% collateral could be required.
* If 1 of your dollars was destroyed, you would have to add 0.75 dollars to your account to return it to minimum balance. If you wanted to add the lost dollar back onto your debt, you would have to add another 0.25 dollars into your account first. Again this is purely psychological because the 3/4 of a loan you could still would be the same size as the 100% of a loan you could steal if there was no bond. This is because of the effective increase in demand for many caused by taking the bond amount out of the economy. The real collateral is still reputation.
* Issuance of money inherently cannot be backed by tangible collateral, otherwise the loss of liquidity caused by the requirement to hold collateral would counter the liquidity added by the money, so the net supply of liquidity in the economy would not be changed and no additional demand for money would be satisfied. Reputation is the only kind of collateral that works for the issuance of money because it cannot be traded so no liquid commodity is withdrawn from the market by offering it as collateral. It is a negative asset. You can give it to someone without losing anything or you can take it from someone without gaining anything but you cannot trade it.

# Fri 11-Feb-11

* Item owners earn some interest for money invested in their item.
* Community selects item owners. Owners decide how much of donations to donate back and how much of their earnings from donations go to investors as dividend. Their choices apply to both positive and negative donations and investors. Community determines defaults for both percentages.
* For the decision item about donation return percentage, influence attenuation is set to 100% for all association types. Actually both the inner and outer reputation domain should be set to the item the decision item is about. There should also be an additional restriction on investors that they be owners (recursive - so owners of owners as well) of the item.
* When you own an item you inherit its reputation in all domains (in proportion to your ownership).
* The Wheel of Time owns The Great Hunt. Robert Jordan has reputation in The Great Hunt by owning The Wheel of Time.
* An owner's influence in item admin (like choosing donation return rate) is dependent not just on percentage ownership but on reputation in the whole reputation domain of the item. So a hacker may have a small percentage ownership of an open source project but have a lot of influence in making decisions about it because he has high reputation in the reputation domain of the project for code commits and comments he received lots of reputation for.
* A review or comment about an item can include multiple investments: one for the item and each of its attribute subitems.
* Amount of donations specified to be kept is sold immediately for profit to the owners. The part that is to be given back is held by owners for a period of time. In that time the owners can choose whether or how much of the donation to return. It would be a bad idea for them to do this since it would undermine trust. It would also increase the relative desirability of investing negatively in your item. You could take money from investors/donors you don't like though such as institutional investors you don't want profiting from your success and popularity.
* Owners decide the amount of reputation an ownet needs to manually keep (part of) a donation. If an owner has a grudge against an investor/donor and keeps his donation (on behalf of all owners), the other owners can override him by opening a decision item about it that is only open to owners. If the group decides to, they can donate back the donation late. There should be a cooldown period after an owner decides to keep a donation during which the investment asset cannot be sold by the item owners. This allows the decision to keep the asset to be reversed by other owners, by giving the asset back to the donor.
* Negative donations are donations to the whole zutha community. The zutha community determines a default reputation required to keep a donation and some exceptions for specific items. A user with sufficient reputation can mark any negative donation to be kept by zutha. This opens a decision item, which will pass if it remains empty at expiry, which allows other users to dispute the initiator's decision to keep the donation.
* Item owners still determine the fixed tax rate (proportion of donations kept) for both positive and negative donations, but the tax on positive donations goes to them and the tax in negative donations goes to zutha.
* Zutha should no longer tax positive donations at all. This will put off many potential donors and it just doesn't seem right to me anyway. A donation should not be taxed. Negative donations however are to the zutha community; they are a philanthropic act to criticise an item for the benefit of the zutha community.
* Ideally there should be no taxes in zutha.
* Money handlers (or issuers) pay interest on their loan from zutha. Is this at all like the tax of government caused inflation? No because the interest is paid by the receivers of a loan, not by all holders of money regardless of whether they receive anything in return.
* Current "winners" of a decision item are its owners and receive reputation from the worth of the decision item. No I don't really like this. Winners should not receive reputation but they should earn the money zutha assigns to decision items according to the reputation if their decision item and the stake they have in the winning outcome.
* You can invest negatively in a decision item and reduce the weighting of a particular outcome.
* A proportion of negative real money donations to the decision item should be divided up among all negative zuth investors as dividend.
* The same should apply to positive donations. I was considering that only current winners should receive dividends, but this would create too much incentive to invest on the current winner and make the market too static. Also it would not be symmetric with negative investment.
* In this way money investors can indicate the decision items they are interested in and encourage zuth investors to invest more in those decision items, but money investors do not influence the outcome of the decision item.
* zuth investors have an incentive to invest in empty outcomes because zest is cheaper there and zest determines how much of the group dividend you get. Zuth investors also have an incentive to invest in the winning outcome provided it is not too full because owning zest there gets you a proportion of total item earnings which includes payment from zutha and the retained part of positive donations minus dividend. The net effect of all these incentives should be a spread out distribution of investment with higher areas where people have strong opinions and a peak at the consensus point (not necessarily the highest point) which will be the winner.
* The proportion of donations in decision items that are kept should be determined by a universal default (which can vary over time).
* Item owners can specify a proportion of money their item should hold in savings. They can then use decision items to make discrete payments to others from the item's account. This would be a useful model for charities and even companies.
* Philanthropic organisation provides unlimited employment at their selected minimum wage. This would provide a recruitment ground for employers to find and employ the workers worth more than the minimum wage. It would be better that a for profit company employ the more valuable employees since they would be more specialized in producing wealth. The philanthropic organisation would be more specialized in finding work for large quantities of unproductive workers and training them.
* It would be in the interests of companies to contribute to the philanthropy organisation for the recruitment service it provides. This might take the form of a commission to the charity for each worker they employ - perhaps in proportion to the salary of that employee. This would create an incentive for the charity to innovate in the education of their army of initially unproductive workers.
* Money handlers could accept any kind of asset in exchange for issuing zutha dollars. players of zuthaverse could exchange virtual gold for zutha dollars at a virtual bank. Zutha would distribute the gold among money handlers who accept zuthaverse gold, deduct the market value of that much virtual gold from their accounts as debt and give that much money to the seller of the virtual gold as zutha dollars.
* It is important that zutha currency is only issued in exchange for wealth of equal value as defined by the bundle of goods underlying the zutha dollar. If the wealth given is destroyed, the issuer will suffer a slight increase in interest costs and an increase in debt which will mean a slightly higher rate of zutha dollar destruction which he'd have to pay for. The value of the Zutha dollar, however, would be sustained, regardless of whether the payment for a zutha dollar was kept or not. A virtual gold coin exchanged for a zuth dollar could thus be destroyed. This would be asking a money handler to owe zutha money in exchange for nothing though. A dollar could be destroyed when the virtual gold coin is exchanged for a zutha dollar. This would just be making the two currencies exact equivalents and using their exchange as a trigger to remove money from the money supply. I cannot see any benefit to this over using zutha dollars as the game currency and destroying money at a constant rate. It would be better if the value of a virtual gold coin could float so that it could be created by game events without the need to explicitly assign debt to anyone. If virtual gold can be created arbitrarily in exchange for nothing, it would be very difficult to pin it to a particular real value. Governments face this issue when they issue currencies arbitrarily - though they are in a somewhat better position because they sell the issued currency rather than giving it away, so they could theoretically sell it at a fixed price defined by a bundle and then regularly destroy currency they get from taxes to oppose the increase in money supply caused by issuing the currency.
* Virtual gold could still be traded with a money handler in exchange for issued zutha currency. The market value would be found by what virtual gold is trading for on the open market in terms of zutha dollars.
* Is it destructive (or at least wasteful) of wealth for players of mmorpgs to spend hours making virtual gold to sell for real money when the mmorpg owner can produce that wealth (the virtual gold) for free?
* if the game owner gave virtual gold away for free in infinite quantities the price and the value of a unit of gold would both fall to zero. If they restrict how much they give away and it has positive value to players, then players will trade what they get for free at the market value. The quests and monster kills required to obtain virtual gold in mmorpgs is just a fun and fair way of selecting who gets the free virtual gold. The fun of the game, and thus the value of the virtual gold, is dependent on the gold and other virtual assets being difficult to get.
* You could argue that wow gold farmers expend effort and get paid without producing anything. Would the world be better off if blizzard sold gold directly? At least they wouldn't expend any effort to produce it. Every gold piece blizzard issued would be taking value away from all other gold holders, just like government issued currency. The difference with zutha issued currency is that the issued currency is a debt that will be paid by those who issue the currency. In zutha's economy this takes the form of the currency that needs to be destroyed to sustain the value of the dollar being taken from the issuers of the currency and thus the holders of the debt. In an mmorpg the currency that is destroyed to avoid inflation is taken away from players when they buy virtual goods. But players get value in exchange for these purchases. Where does it come from? The mechanics of this economy would probably be better illuminated by taking the virtual gold out of the equation. If blizzard gave virtual items away for free they would have no value. A good analogy is a medal; if you gave away a medal for free it would have no value but if you (or someone) had to work hard for it (probably doing something that produced no wealth like beating some people in a race) the medal can be worth a lot of money. Virtual items in mmorpgs are like medals; they are proofs of having achieved something difficult; carriers of reputation. The value of gold in wow is precisely the difficultly of obtaining it. Blizzard is selling reputation in exchange for the service of providing a challenge whose completion people will accept as worthy of reputation. Gold farmers are selling the reputation they could have gotten from the gold they farm. By doing so they reduce the value of the gold as a reputational currency and thus reduce its monetary value as well.
* If blizzard sold gold for real money the stuff you could buy with it would most of its reputational value. But real money carries its own status - just not as much as gaming time and skill in the gaming world. There is also a pure fun component of the value of virtual items which is not dependent on the accompanying reputation. Much of the fun relies on having something others do not though so it would be essential to maintain scarcity. Blizzard would then be selling entertainment value and the right to flaunt stuff others lack. If they inflated the currency by issuing too much they would still be taking value from others who now have less value than they paid for. The virtual goods still count as currency - in fact they are the real reputational value - so the only way to take currency out of the system would be for blizzard to buy back currency and destroy it. Actually consumable goods are demanded at a rate rather than at a particular static quantity for a given a fixed currency value, so that could be a way of taking currency out of the system without destroying wealth. There would still be a need to give players virtual gold in exchange for exertion in the virtual world. How could the value if the virtual currency be kept constant with both an inflow and an outflow that are unrelated to monetary value? The outflow could be made to be higher than the inflow by increasing the prices of virtual consumables and decreasing gold drop rates as necessary. A real money gold purchase price could then be set and demand would drive up the supply of gold until its value fell to its cost. Provided the net outflow of gold exceeded the desired outflow rate during any periods of decreasing demand, the value of the virtual gold would remain constant.

# Sat 12-Feb-11

* How to create good incentives for doctors
  + When a doctor provides a medical service, he also offers an insurance policy at a constant price that he selects, from that day forward. The insurance policy covers medical expenses for operations in the same domain as the medical service he is offering, including any health issues that could be caused by the medical service being provided or arise as complications from the current illness the patient has.
  + Because the patient may stick with the offered insurance policy potentially for the rest of his life (though maybe this deal could be limited to a set number of years), the doctor must select a price that will cover the expected costs of the patient over the longest possible duration of the offered policy.
  + It would be up to the doctor to estimate the expected costs of his patient as accurately as possible. If he got it too low he would suffer net losses in insurance payouts, and if he got it too high, he would be undercut by the competition.
  + The doctor now has an incentive to do the best job possible in the medical procedure because the more he can reduce the patient’s expected future medical costs, the more competitive will be the insurance policy he can offer.
  + A patient can now relatively safely choose the cheapest doctor that has been that cheap for a long time because this doctor will be the one that reduces your chance of needing future medical care in that domain the most. The reason the doctor needs to have been around for a while is that that will filter out the doctors who charged too low a price and went out of business from consistently paying out too much in insurance payouts.
  + A patient can switch to another doctor and transfer his insurance policy to that doctor at any time. The new doctor will specify his own insurance policy price at the time of the first appointment. Patients can shop around and find the doctor with a sufficiently high and long lasting reputation that is offering the cheapest medical costs and insurance policy for the intervention the patient is after. The fact that the doctor offers the cheapest insurance policy is an indication that he is the best doctor (provided, as we said that he has been in business for a long time). Given that he has the cheapest credible insurance policy, it can be trusted that the price he charges for the medical intervention itself will accurately reflect costs but not be a cheaper intervention than necessary to maximise your reduction in future medical expenses in that field.
  + Doctors could, and likely would under this scheme, band together to pool their risk. A collection of doctors, say a hospital, would collectively provide the insurance policy at a price recommended by the doctor who provides the initial intervention. It would be up to the hospital to figure out how to identify the best performing doctors and get rid of the bad ones. The hospitals that did so effectively would win in the market.
  + Crucially, the second and later appointments with the doctor still provide the doctor with the same incentives because the more he can reduce your future medical expenses, the less he has to pay in insurance payouts.
  + The reason it is important that patients can change doctors whenever they want is that their existing doctor could start charging higher and higher prices for medical interventions once the patient already has insurance with them. The fact that the doctor has to pay a proportion of the medical costs doesn’t stop him charging more than their actual cost so that he gets more money from the part of the expenses the patient has to pay. If the patient can change doctors at any time, she can see whether there are other doctors who would provide the intervention being offered by her current doctor at a cheaper price with an equal or lower insurance premium. If the doctor offers an equal priced insurance policy and a cheaper intervention, the patient need not change insurance providers. The price the doctor is willing to offer for the insurance policy is an indication that he thinks he is able to have just as positive effect on the patient’s health at a lower cost.
  + There is no reason doctors could not make a profit from offering insurance. Insurance providers can make a profit because people are willing to pay more than their expected future costs to the insurance company in exchange for a reduction in risk and uncertainty in their life. Insurance companies are forced to charge more than the actual expected price of a customer to them because they cannot accurately measure the expected costs so they must err on the side of caution. Both the demand and supply side of this equation combined make it possible for suppliers (insurance companies) to make a profit.
  + The best part about this scheme is that it does not rely on patients acting in their best interests to identify the best doctors in order for them to benefit. If, as is common, patients don’t bother to shop around and compare doctors, they will pay more than they need to and won’t get the best possible care, but the doctor they do chose will still have an incentive to be as effective as possible at reducing their future health costs, so the patient will be far better off than under all current medical care schemes I am aware of.
  + A charity could do the shopping around part for a patient they offer free or subsidized medical care to. The charity, as an institution would not suffer the same psychological disinclinations that humans do to question the incentives of their doctor. The charity would have an incentive to make their funding go as far as possible in order to attract more funding through their success at helping lots of patients. So the charity would search for the cheapest, and, under this scheme – provided the doctor has a longstanding reputation – therefore the best doctor to treat each one of their free health care recipients.
* Only high reputation Zutha users should be allowed to create a superior host – one that hosts all data.
  + This would make it harder for just anyone to get hold of the entire up-to-date database
  + This, in turn, would make it harder for someone to “touch” your data without notifying the Zuthanet of that action.
  + Daniel Suarez expounded the virtues of a system where people at least know when others have touched their data, even if they cannot realistically stop them from accessing it.
  + If you cannot stop people from accessing your data than you cannot ultimately prevent them from viewing it secretly. However, by providing a service like Zutha that makes it extremely easy to view someone’s data but be tracked doing it, and quite difficult to view someone’s up-to-date data secretly, you make it very likely that people will know about their data being touched most of the time.
  + Anyone could create an inferior host and read and store all requests that pass through them, accumulating a small subset of the Zutha database. Unless you managed to become a major conduit for traffic, you are likely to miss most new items because they will pass by you on more efficient routes to the superior nodes. You also would not be permitted to download the entire Zutha database as an inferior node, the way a superior node does when it is first created. It is likely, though, that historic copies of the whole database would find their way into the public domain because there would be a demand for legitimate uses of a local copy of the database that even high reputation users would be willing to satisfy. This is why I qualified that it would be difficult to get access to an up-to-date version of someone’s data. It might be that not up-to-date just means more than a few days old, though.
* A client can ask a neighbour to manage a request for it
  + Client A signs a request with its user’s private key and asks node B to manage it
  + B calculates the request handlers for the message and sends it to each of them
  + B receives the results from the request handlers and sends a single consolidated reply to A
* Selection algorithm
  + Every named node has an associated large random number which it may change at most once at the start of each timeframe. It should change it with a low probability each timeframe.
  + The timeframe key for timeframe t1 is calculated in the second half of t0 after all hosts have had a chance to change their random number and any new additions to the named host list have been processed.
  + Let D be the hash of the ID of the item representing the domain of the request whose request handlers are being chosen. A domain could be a specific item (which is being edited or viewed), a user, a conversation, an MMORPG region etc.
  + The timeframe key K is the sum of all the hosts’ random numbers plus D
  + Let R be the sum of the reputations of all named hosts
  + Let X be K mod R
  + Order all the named hosts by their host ID
  + The selected host H is the nth host such that X lies between the sum of reputations of the first n-1 hosts and the sum of reputations of the first n hosts
  + H is added to the set of request handlers (the controller group) that managed this domain last timeframe, and the oldest member of that group is removed.
  + If a new domain has just been created and had no controller group last timeframe, the process of selecting a host is repeated several times, each time excluding the hosts selected so far from the list of named nodes.

# Sun 13-Feb-11

* How to ask for something to be done, answered or created and offer a reward for those who satisfy you most
  + Create a Request, Question or Product Idea item, or something else along the same lines
  + The owner of these item types is the creator of the item, so you get control over the item
  + In the item description you describe in detail what you are after
  + You set the division of the item’s income so that a large percentage goes to beneficiaries and a small percentage goes to owners (you).
  + You leave the beneficiaries empty. Revenue for the item that would have gone to beneficiaries is now put into the Item’s own savings account instead.
  + Now invest a large amount of Zuth in the item to advertise it, and donate a large amount to the item to show there is a monetary reward awaiting those who satisfy its requests best
  + Others who also want your Question answered or Product Idea realised can invest zuth in the item and increase its and your reputation, and they can donate real money to the item to increase the incentive for someone to satisfy its requests.
  + If you have given yourself a positive proportion of the item’s income, you will slowly eat away at the initial endowment you gave the item because you will take away a percentage of the item’s current account each payday.
    - I should design a way for people to prevent this happening while still allowing savings to flow out when appropriate
  + Once someone satisfies your request you make them one of the item’s beneficiaries and assign them a proportion of revenue assigned to beneficiaries depending on how satisfied you are with their solution to your request.
    - If the percentages assigned to beneficiaries does not add up to 100%, whatever is left over is sent to item savings. This is consistent with the way 100% of money assigned to beneficiaries is sent to item savings when 0% of beneficiary allocation has been used up by actual beneficiaries.
  + If competing solutions to your request to the first one arise, you can add them to the beneficiaries of your Request/Question item as well, and give them what you think is a fair proportion of the beneficiaries allocation.
  + It should be possible for the wider community to contribute to selecting beneficiaries
    - The Beneficiaries Decision Market for a Question/Request item should specify a low (restrictive) investment cap on the reputation domain of the Question/Request item itself, and a high degree of influence attenuation for all parent-child associations.
    - This will mean the owner of the Question/Request item, being the only one with reputation in the domain of that item itself, will have the most power over determining beneficiaries of his Question/Request item
    - But owners of child items of the Question/Request item, like comments, will have reputation in the domain of the Question/Request item itself as well
    - Perhaps the reputation you get for a domain should attenuate the more associations it is away from the item you actually own
    - Reputation attenuation could be the same as influence attenuation for any given parent-child association
    - So given a parent and child item, the owner of the child would have the same influence over the parent as the owner of the parent would have over the child.
* Zutha should pay money to items based on their worth, not their reputation, otherwise people could make a long chain of ownership and produce total reputation equal to an arbitrary multiple of the worth of a single item. If Zutha paid by reputation, this would allow exploiters to draw more of Zutha’s money without producing any more worth.
* Investment cap and edit permission should both be measured in the same domain-weighted reputation metric
  + Investment cap is determined by a percentage of your reputation in the inner reputation domain of the item, plus a progressively lower percentage of your reputation in each domain outside the inner one.
  + Let’s call this domain-weighted reputation metric for a given item your “Influence” over that item
  + The permission required to perform some action should be measured in Influence over a particular item
  + Example: It might require Influence of 5000 over a Question item in order to add a beneficiary to that item
  + Example: Influence of 10000 in Zutha Ontology Administration and Influence of 500 in the Book type item is required to modify the description of the Book type item.
* The expected behaviour of Zuthanet nodes should be defined by a standard
  + There may be many implementations of the standard. The more the better because the more different implementations the more resilient the system will be to bugs in any one of the implementations
  + The standard may change over time.
  + Once a new standard is accepted, the developers of the various implementations should be given time to update their implementations to the new standard. Once the major ones give the go-ahead, a date should be set for the new standard to go live. Node owners will have until then to update their software to the latest version of the implementation they have chosen.
* Money handler issuance allowance proportional to current money supply and their reputation. Total issuance allowance should be slightly higher than current money supply. A money handler thus has some leeway to voluntarily create some additional money than what zutha would ask of him. He could pay for a haircut with money issued on the spot if he had spare quota. In order for the money to be created to pay for the haircut, both parties would have to confirm to zutha how much they each traded. The hairdresser should know that he is selling his services in exchange for created money and that, in pricing his services, he should value the zutha dollar at its indexed price. If there is a shortage of money the actual value of the zutha dollar will be higher than its indexed value and the hairdresser will get a bargain by selling at the indexed value. The money handler gets the haircut effectively for free so he will likely also feel he is getting a deal. If there was excess money the hairdresser would only be willing to trade using the lower actual value of the zutha dollar which would mean a higher nominal price for his haircut than it would cost in terms of the index. The money handler though would have an incentive to offer the higher trading price because he won't have to pay the full indexed price until later and only gradually even then. Zutha would need to prevent money being created on such trades or it would allow continued expansion of the money supply when there is already too much. If zutha knew what the hairdresser would charge in other currencies it could determine the correct indexed price of the haircut in terms of zutha dollars and only give the hairdresser that amount. This would not likely be practical in the case of a haircut, but if zutha could know how much the zutha dollar was actually trading at compared to its indexed value... well if it knew that it could prevent any extra currency being created at all until the trading value of the zutha dollar rose back up to the indexed value.
* Prediction markets should estimate the price of all the commodities in the Zutha Bundle in terms of the zutha dollar. If the price of the zutha bundle is greater than one zutha dollar then the value of the zutha dollar has fallen below its indexed value so the money supply needs to shrink. zutha should thus stop any additional creation if currency and increase the rate of currency destruction until the price of the zutha bundle equals one zutha dollar again. If the money supply needs to expand, zutha should increase the quotas of its money handlers so they can create more money. Under this scheme there would be no reason why money handlers couldn't create money whenever they want (up to their quota) and spend it on anything they want at any price they want. In fact the money supply could be increased and decreased by simply adding and subtracting appropriate amounts from money handlers' accounts and modifying their debt accordingly. If they don't want to use up all their allowed quota they simply leave the zutha dollars in their account and zutha will remove them and take away the corresponding debt when it needs to.
* Zutha's monetary policy can be defined by a single function that raises the allowed money supply (sum of all money handler quotas) when the price of the zutha bundle is less than 1 zutha dollar and shrinks the allowed money supply when the price of the zutha bundle is greater than 1.
* Zutha can still play matchmaker for the trading of zutha dollars for other electronically transactable currencies. Zutha can enforce the indexed value of the zutha dollar when it does this and this will contribute to the stability of the zutha dollar. If the dollar is weak (too much currency) people won't want to buy zutha dollars through the automated system. Money handlers will have high demand for zutha dollars at such times though because they will need to be replenishing their accounts to the minimum level as zutha destroys money in their accounts. They will thus be more likely to be willing to pay the slight premium to purchase zutha dollars through the convenient automated system. Since holders of zutha dollars will be eager to sell at the higher price, currency should flow to money handlers where it can be destroyed.
* When money is scarce most people won't want to sell zutha dollars using the automated system because they will get less than market price. Money handlers however will have excess cash in their accounts due to zutha expanding the money supply. They will likely want to get currencies of countries they want to invest in so they can earn a return in the loan they have from zutha. Thus they will likely be more willing to sell their abundance of zutha currency cheap.
* Zutha can now maintain the value of the zutha dollar without knowing anything about any of the transactions involving zutha currency - it will know who sends how much to who but need not know what was traded in return.
* Pinning the zutha dollar to a bundle will push world currencies towards purchasing power parity in that bundle.
* The price of a good in zutha dollars would find some weighted average of its price in all the currencies in the zutha bundle.

# Mon 14-Feb-11

* Namibia could be a good place to start Zutha in Africa
  + The community run conservancies there have characteristics similar to the kinds of communities I envisage arising under a Zutha umbrella
  + Conservancy communities could be offered Zutha technology: Zutha clients and accounts to communicate with each other and the rest of the Zutha community
  + Conservancy communities could make great use of the Zuthanet to monitor poaching, organise the allocation of funds and make collective decisions. Having direct access to the Zuthanet, including the entire internet, from handheld or wearable clients would provide enormous opportunities for education and income from the Zutha community and the whole of cyberspace.
  + I think there is a strong possibility that being connected to the Zuthanet could facilitate self-government arising naturally among existing communities, potentially without conflict if Zutha is able to provide them with sufficient defensive weapons like tranquilizing UAVs.
* It might be feasible in the near future to transport rainclouds to dessert areas to make them more habitable for humans
  + Cloud Seeding already allows rain to be precipitated from humid air, so if water vapour could be transported from damper regions (like over the ocean) to dessert regions, we already have the technology to make them rain.
  + Cloud Transportation technology would be worth investing in to allow Zutha communities to create habitable territories for themselves in currently uninhabited areas, and to make poor dessert areas capable of far greater prosperity.
  + It could be more efficient to harvest existing clouds for rain where they are and transport the liquid water to dessert areas. For example aircraft could suspend lightweight fabric catchments beneath clouds over the ocean and use cloud seeding to make the clouds rain into the catchment. The water caught by the catchment could perhaps be piped directly from there to land using suspended lightweight pipes made of carbon nanotubes or something. It is possible this process could become cheaper than desalination.
* Money handlers that fall behind in paying back their loan to Zutha should be able to get a period of less or no interest in order to catch up with their debt, if they can convince the Zutha community to allow it.
  + For example if a money handler was to leave very little buffer in their account and became inactive for a while, the Zutha money supply might shrink enough in the meantime to push their account below their mandatory minimum balance. This would lead their reputation to start falling, which would cause their quota to fall, and over time their balance would fall further as Zutha destroys their money to reduce their loan to their lower quota.
  + During this time their interest payments would also be deducted from their account leading it to fall even faster
  + As their balance falls further below their required minimum their reputation would fall even faster, causing their quota to fall faster and leading to a vicious cycle
  + They would thus begin to owe more and more money at a faster and faster rate
  + There is a trade-off here between ensuring that loan commitments are binding and that irresponsible money handlers are not rewarded for bad behaviour, and making it realistic for money handlers that fall behind to catch up.
  + I think it should be possible for sufficient community support to grant temporary relief from falling reputation and large interest requirements if a money handler shows that he is working diligently to repay his loan. If he is actually depositing new Zutha dollars regularly into his account this shows that he has a source of income – whether from investment within or outside Zutha, from Zutha items he owns, or from conventional employment – and that he is serious about catching up on his loan. In such a situation it would be in Zutha’s interests to encourage the money handler to continue working hard to repay the loan, rather than discouraging him to the point of giving up by burdening him with rapidly increasing debt that he cannot keep up with.
* Decision items can have arbitrary filters on allowed participants, including a specific list of users
  + An group of people could privately use this system to democratically make a group decision.
  + Example: 10 friends want to decide what movie to go and see.
    - This is very difficult to work out verbally because everyone has varying degrees of preference for any given candidate movie that has been put on the table and it is very difficult to find the movie that will maximise collective utility just by group discussion.
    - With a Zuth Decision item, and perhaps a finite allocation of zuth per person specifically for the purpose, each member of the group can allocate his zuth among the various movies that are proposed according to his preferences. The movie that receives the highest aggregate worth after all the voting is the one that maximises collective utility.
* People can put items on their “watch list” Wikipedia style.
* Integration with Wikipedia
  + It might be possible to connect with Wikipedia’s edit stream via an API.
  + Wikipedia users could be mapped to corresponding Zutha users
  + The stream of edit requests coming from Wikipedia for a given article could be turned into Zutha edit requests for a corresponding Zutha item
  + Wikipedia users would accumulate reputation on Zutha in their corresponding Zutha account
  + If an edit request coming from Wikipedia was from a user without sufficient “influence” on Zutha to edit the corresponding Zutha item, then that edit request would be added to the events for the Zutha item, but as a failed attempted edit.
  + A user with sufficient influence to edit the item could go through the attempted edits from Wikipedia and accept the ones he thinks are acceptable
  + Edits to Wiki items could be permitted directly through the Zutha interface
  + If an edit request came from Wikipedia through the API that conflicted with the existing state of the item on Zutha, the edit request would be placed in a list of conflicting edit requests and users with sufficient influence could go through them and resolve the conflicts, perhaps by manually editing the Zutha item to add any additional information provided by the Wikipedia edit request.
  + This model would create a sort of competition between Wikipedia and Zutha to create the best articles, though many of the contributors would be users of both services and contributing to the same articles in both of their manifestations.
  + I think the Zutha reputational (and eventually somewhat monetary) reward system would lead to improvements over Wikipedia in the quality of articles
  + It might actually be a better model to delegate such integration with specific services like Wikipedia to individual users who develop their own bots for carrying information from other services into Zutha
  + This would be a more general model and would be more scalable
  + The Zuthanet should be a standard which is defined as broadly and as generally as possible.
  + 3rd parties with Zutha user accounts can design bots to mine data and inject it into Zutha, and synchronization bots for synching data from such services as Wikipedia with Zutha.
  + There might be some issues with conflicts between different bots trying to synchronize the same service with lightly different operating rules, but Zutha’s permissions and reputational reward and punishment model should be able to deal with this in exactly the same way as it deals with conflicts between different people.
  + If a bot has poor synchronization rules and causes a mess when another bot is working at the same time as it, then this bad bot will lose reputation if it does much damage and will soon not be allowed to continue causing damage.
* Donating negatively to an item is effectively donating to every other item in proportion to its reputation
  + This is quite appropriate because donating negatively to an item is a statement that it doesn’t deserve to be valued as highly as it is, so you increase the value of all other items relative to it, both by reducing the real money worth of the item you invest negatively in, and by donating money to Zutha which will be distributed among all other items.
* The current rate of change of an item's worth is sent to clients so they can show smooth real-time movements of item reputation bars. All buying and selling of zuth investments happen over time so an aggregate rate of change of total worth can be calculated as the sum of all the positive and negative rates of change of the current buy and sell operations that are going on. When new buy and sell operations are requested this information can be pushed to clients in the form of a new fixed worth at a particular moment and the rate of change starting from then. Clients can find a smooth transition from the current trajectory to the new one where they will move back to a constant rate of change at the new rate.
* Holders of cash should earn the same interest as item owners do on the worth of their item. Having money in Zutha should emulate storing it in an internet banking account such as Netbank, where interest rates are high even though you can withdraw electronically at any time.
* Zutha should earn interest only in negative investments and the money it holds in its own savings. This is in keeping with a philosophy of truly considering item owners the deserving recipients of all money positively donated to their item, including interest on any part they donate back. This will mean items with no owner or whose owners are inactive will accumulate money donated to them by users or zutha and will also accumulate interest on this growing amount. Ultimately if items are never claimed all the benefit will be enjoyed by money handlers since the money supply will grow more as a result of money sitting around unused. All this money sitting around will act as a buffer against money handlers defaulting though because even if every active user gets their money out of zutha there will still be money sitting around that won't have to be paid for which is good because money that was issued by money handlers that since defaulted (by abandoning zutha and letting their reputation fall to zero) cannot be refunded.
* The fact that items have real money sitting in them will be a very effective way to encourage their rightful owners to start using zutha. I think zutha will benefit on the whole by granting as much income as possible to item owners since this will strengthen incentives that will grow the zutha economy which will be a more effective way to increase zutha income in the long run than imposing higher taxes and interest rates.

# Tue 15-Feb-11

* Zuthanet serving of large files
  + A video file would be broken up into multiple parts, each held in a separate Video Data item
  + Each Video Data item would have a controller group that rolls through hosts steadily like any other item
  + As soon as the new member of the controller group, for the next timeframe, is known, it is sent the Video Data item that is being managed by that controller group
  + The host that falls out of the controller group is allowed to delete the data component of the Video Data item if it wants to, but it must retain the metadata which stores the Video Data item’s ID and association with the Video item itself.
  + When a client wants to download a video, it streams in the Video Data items one by one. It knows which controller group is responsible for each Video Data item because each Video Data item has a globally known ID which can be used with the selection algorithm to determine the controller group for a particular timeframe.
  + For clients who connect to the Zuthanet through a bottleneck such as an internet connection, it may not be desirable to download many redundant copies of each piece of video data in parallel. The client should thus be able to select one or a small number of the hosts in the controller group to actually send the video data, and the others can just send the hash of the video data.
  + Actually the hash of the video data should be stored globally as part of the compulsory metadata that all hosts must hold of the Video Data item.
  + The controller group in the case of serving data can thus be treated simply as the known group of hosts that have the data itself. A client can ask any of them for it (choosing the closest one probably) and check that what it sends matches the globally known hashes for each Video Data item.
  + The fact that each part of a file will be stored on a different constantly changing set of hosts will make it extremely difficult for the file to be eradicated from the network.
* I think Encyclopaedia articles about a Topic should be subitems of the topic itself
  + This will allow competition between all kinds of internal and external sources of information about the Topic/Item
  + Items with a corresponding Wikipedia Page will have a linked item representing that Wikipedia Page, as well as a direct link from the item itself to the external URL
  + Items may also have multiple internal Wiki Encyclopaedia articles associated with it and the Zutha community can vote on the best one
  + Teams can get together to build their own encyclopaedia articles or other content. They can have their own rules and social norms for giving reputation to their members and allowing new members in. Teams can the compete against each other to produce the best content in each domain.
* There’s no reason why a group or individual user couldn’t set up a bot controlled synchronization mechanism between their own set of Zuthanet hosted wiki pages and Wikipedia.
* Currently the Zuthanet will be very inefficient at finding a route around a bottleneck because there will be a lot of links to explore behind the bottleneck that all keep trying to point towards the bottleneck.
* The path of a message through the Zuthanet towards a destination can be modelled as the flow of water towards the lowest point of a complex terrain
  + For a given destination, a node has uphill and downhill neighbours
  + A node X knows which neighbours are uphill neighbours for a destination node D because they send it messages that are destined for D. Also, when D sent out its Announcement Beacon, X would have only received messages from its downhill neighbours and it knows the rest of its neighbours are uphill neighbours.
  + A node knows the latency between a given downhill neighbour and D
  + When a node first receives a message destined for D, it will simply forward it to its lowest latency downhill neighbour
  + If a node X receives a failure response from a neighbour it forwarded a message to, X will try to send the message to its next closest (in terms of latency) downhill neighbour to D
  + If X has exhausted all its downhill neighbours it should send a failure message to ALL its uphill neighbours, even those that have not yet received the message at all
  + This behaviour emulates water rising in a valley that has been blocked at its lowest end. The water will keep rising until it finds a way down somewhere. There is no way to know where the water surface will meet a route to lower ground so the water must rise uniformly across the whole valley.
  + When X sends failure messages to all its uphill neighbours it should swap its uphill and downhill neighbours because it is now “submerged” so the path to D is by floating to the surface first; thus formerly downhill neighbours will now be the source of messages to D and formerly uphill neighbours will be candidates for forwarding messages to D.
  + Once a blockage in the valley has occurred, the only way it can be unblocked is when D sends out another Announcement Beacon and all routing table information about it is completely refreshed. If the blockage has been unblocked when this happens, nodes in the submerged valley will receive messages from D from “below” them and know that the blockage has been removed.
  + Consider water rising up two different streams from the lake over the blockage in the valley. Nodes in one of these streams will not know if nodes in the other stream have found a way to lower ground, so they will keep making the water rise in their stream (water in the Zuthanet is permitted to defy some of the laws of physics). Eventually all streams will find their own way to D. There will be a strong likelihood that water rising up a stream A will meet another stream B that has already found a way downhill. Nodes in B will notify the nodes in A that stream B is already flowing downhill and nodes in A will wait to discover the outcome of the water (message) flowing down B. If stream B actually finds D, nodes in both A and B will receive notification that stream B is the way to D. If D is not found via steam B, both stream A and stream B will keep rising, now a merged stream.
  + In terms of nodes and messages, this behaviour is achieved as follows
    - Node A has received a message (for the first time perhaps) from a downhill neighbour, indicating that the message cannot get though via the downhill neighbour.
    - A has one other downhill neighbour, B, which A forwards the message to
    - B has already received the message and has sent it to its best downhill neighbour
    - B receives A’s message and records the time it is received, but does not immediately respond to it – except perhaps to indicate that it is received
    - If B receives a response from D it will forward it onto A along with the time A’s message would have taken to get to D if B had sent it again when it was received from A. Actually a better way to do this would be for B to delay sending D’s reply on to A for a period equal to the time between first forwarding the message toward D and receiving the incoming message a second time from A.
    - If A receives a success response from B, A will update its routing table with the new latency information for B as a downhill neighbour towards D. A will then only respond to the neighbour that sent it the message, though there may be multiple, as there was with B.
    - If B does not receive a response from D after exhausting all its downhill neighbours, it will send a failure response back to A, as well as all its other uphill neighbours
    - A will then also send a failure response to all its uphill neighbours
  + Hosts/Named Nodes should periodically send out Announcement Beacons to the entire network (at different times from each other) to regularly keep the network up to date with the most efficient way to reach them.
  + I think nodes should forward messages to all downhill nodes after a failure response because A’s second closest neighbour to D may be closer than B’s second closest neighbour to D plus the distance between A and B.
  + Messages that get blocked trying to get back to a client will not go searching for the client but will give up. If a client does not receive a reply from the host it was sending to, it will have to send the message again in order to find a new route to the host.
* What routing information does a node need to store?
  + For each Named Node D as a destination, a node must store an unordered list of uphill neighbours and a list of downhill neighbours associated with and ordered by their distance from D in terms of latency
  + Each node must also keep a short term database of the closest neighbour to each client that has sent a message through them in the recent past.
  + Clients are expected to maintain their connections with hosts they are currently communicating with by sending regular keep-alive messages.
  + If a node has not routed a message from a client in a while, it may delete the associated data.

# Wed 16-Feb-11

* There’s no reason why Documents on Zutha should necessarily be written in HTML
  + Initially, when Zutha is accessible only via the web, HTML documents make the most sense as the standard document format
  + Later, when clients can be arbitrarily powerful and customizable, any kind of document could be read and edited directly from a client
  + The Zuthanet standard should define nothing specific about any kind of data formats, including HTML
  + An HTML document should be stored in the exact same way as a word document or AutoCAD file
  + For some document types, people will write software for clients that make them viewable and possibly editable directly through the client
  + For certain kinds of documents it might also be possible to write protocols for real-time collaboration in the style of Google Wave
  + All file types, including HTML documents should be managed in the same way as Video Data items
    - They each get assigned a rolling controller group
    - All the hosts in the controller group are expected to have the data of the item available
    - Once a host is no longer in the controller group of a data item, it may optionally delete the data, but must keep the metadata
  + Small files like documents will be stored in just one chunk, but larger file will be broken into parts
* Two nodes can be downhill from each other
  + This could be described as a sideways link
  + A sideways links will occur when two neighbours, during an Announcement Beacon from D, send messages to each other before either receives a message from the other. The messages will cross over along the link between them.
  + A sideways link is a special kind of downhill link
  + A sideways link will never be the highest priority downhill link because by definition a node has already received the first message from D (along what will become the highest priority downhill link) when it sends a message along a sideways link.
  + A sideways link may be the second highest priority downhill link however
* Every destination Host is the lowest point of its own independent network topography/terrain
* If a link breaks which prevents messages reaching many destination nodes through their highest priority route, each destination node’s topography will have to be updated independently when messages try to reach them through the broken link
* Actually if a node X discovers that a link has broken, it can look up all the destination hosts for which that link was the highest priority downhill link to that destination. X can then send out separate seeker messages for each of these destination Hosts.
  + There will be two kinds of messages sent out in this situation. There will be the message which X initially tried to send towards D. This message is still trying to reach D. All the other messages, though, are only to alert nodes that need to know, about X’s broken neighbour. These messages need not reach D.
  + Let’s call the messages that are purely to notify relevant nodes about a link break that affects them, remap messages (for now at least)
  + When X discovers that a link is broken, it sends remap messages to all its uphill neighbours for destinations X can no longer reach through downhill links
  + If a node receives remap messages from all its downhill links for a particular destination, it forwards the remap message to all its uphill links for that destination
  + This can be visualized as the water level rising in a valley only until it reaches a ridge over which it could flow whereupon the water stops rising. If the water level rises later on the other side of the ridge and meets the already submerged side, the water keeps rising at that place.
  + This process should notify all nodes that would have sent messages on a route that would require passing through the now broken link, to stop using that route. No nodes should receive a remap message that do not need to change their behaviour though.
* There should be a limit to how high the water level can rise in search of an alternate route to D
  + This is because the broken link may have been the only way to reach D. In fact the neighbour on the other side of the link could have been D itself – a host can hide behind an anonymous node to conceal their exact location
  + Then again, it might be desirable to notify the entire network if a Host becomes unavailable – in fact this would definitely be desirable
* The water level should keep rising until it reaches all nodes in the network if the destination host is never found
  + If this happens, by far the most likely scenario is that the destination host has gone offline
  + Once links to satellite and circling UAV nodes are prevalent, it should be very rare, but not impossible, for a major section of the network – containing a Named Node/Host – to become disconnected
  + Hosts should be rated according to how reliably they are connected to the network, not just on whether their internal systems remain up and running. Thus a host that becomes unavailable to the whole network should be punished by a reduction in reputation
  + After a host forwards a remap message to all its uphill neighbours, it will wait a sufficient period of time to confirm that the destination host was never found and then it will decide that that destination host is no longer connected to the network.
  + Actually, every host should be assigned a rolling controller group like any other entity in the Zuthanet. The controller group will be responsible for managing rewards and punishments of that host and for removing the host from the Named Hosts index if it becomes disconnected.
  + Let D be the destination host that becomes disconnected
  + If a host X in D’s controller group forwards a remap message to its uphill neighbours and times out waiting for a response, it will send a message to all other hosts in D’s controller group notifying them that D has disconnected.
  + All the hosts in D’s controller group should discover that D has disconnected as well because the “rising water” will pass over all of them in its search for a path to D.
  + Every host in D’s controller group should thus receive the same disconnect message from all the others. The standard rules will apply for deciding the timestamp and ID of the event item that will be created for disconnecting D from the network – that is, the ID and timestamp will be selected from the message with the median timestamp.
  + There will be a period between the rising water that reached every node in the network and notified them that D cannot be found, and the time that D is officially disconnected from the network. During this time, D is still on the Named Nodes list according to the selection algorithm, so it will be selected as a participant in many requests. Nodes should simply not bother sending a message to D even though D is one of the hosts in the controller group of their message. Controller groups that include D will know to exclude D in their processing of messages.
  + Once the timeframe arrives in which D officially drops off the Named Nodes list, the controller groups it was a part of will elect a new member, but not expel their oldest member since D was expelled instead.
  + In order for a controller group to be wiped out, more than 1 of their number would have to fall off the network every timeframe until the controller group ran out of members.
  + Actually it may work for a controller group to elect new members equal to the number that disconnected last timeframe because all nodes in the network know that those members disconnected last timeframe so they can know that the selection algorithm needs to be used multiple times to select multiple new members for that controller group this timeframe.
  + This will mean that literally every single host in a controller group will need to be disconnected in a single timeframe in order for the controller group to fail, otherwise it will grow back to full health in the next timeframe.
* Nodes should probabilistically forward messages to downhill neighbours according to the weighting of that downhill link.
  + The weighting of a downhill link should take into account most recent recorded latency, historic latency, how commonly the link has been broken and how commonly the link has been an uphill link.
  + There should also be a small probability of a node forwarding a message to 2 of its downhill neighbours
* It would be useful if new links could be announced to the local network such that it can be used for destinations reachable through that link
  + When a node X makes a new link with a node Y, X should ask Y for its distance from every host – that is, the weighting of its best link to each host
  + For each destination host D, X should calculate whether its distance from D is decreased as a result of the link with Y
  + If X’s distance from D has decreased, X should send a remap message to all its downhill neighbours (with respect to D), which includes X’s new distance from D.
  + Each of X’s downhill neighbours should in turn decide if X’s shorter distance to D makes their distance to D shorter. If so, they should change X to a downhill neighbour and inform their own downhill neighbours that they are now closer to D than they used to be.
  + This cascading message should reach only nodes whose behaviour will change as a result of the new link – well one layer of nodes around these nodes actually because a node has to decide it is not going to change its behaviour in order to stop sending on the message.
* The internet suffers from a vulnerability that rapidly “flapping” links can cause a huge amount of network overhead to rapidly change the relevant data in routing tables back and forth between the two states resulting from the presence or lack or presence of the flapping link respectively.
  + It is important that the Zuthanet avoid this vulnerability
  + A node next to a flapping link should be responsible for deciding when the link is reliably available again. Only then should it notify the local network that a new link is available.
  + If a link has just failed, a neighbouring node should wait a while before notifying the network of its presence again once it becomes available again.
* If a node receives a seeker message from a downhill neighbour other than its highest weighted one, it should forward it initially only to its highest weighted downhill neighbour.
  + If a blockage occurred along the route that would be followed from a node’s highest weighted downhill neighbour, then the notification of the blockage is likely to arrive first through this highest weighted downhill link.
  + If it does turn out that the highest weighted route is also blocked, the node should then send out seeker messages to all its downhill neighbours. Only if all of them are blocked should it forward the seeker message to all its uphill neighbours.
* Links are not really concretely uphill or downhill; they range on a continuum from uphill through horizontal/sideways, to downhill
  + This is because of the probabilistic nature of weights.
  + Many nodes will be horizontal neighbours of each other in the sense that they are both among the highest weighted downhill neighbours of each other.
  + When a node receives a message from a neighbour which is destined for D, it should reduce the weight of that neighbour with respect to D
  + If two nodes are highly weighted neighbours of each other, they will regularly receive messages from each other on the way to D and this will reduce their weighting with respect to each other over time
  + A weighting can become negative when a node receives a lot of messages from a neighbour and doesn’t send many messages to a neighbour when trying to reach a given destination.
  + An uphill link should be defined as one that has a negative weight.
  + Some nodes will have only neighbours with very low weights because all their links are horizontal links.
  + Two nodes that are very close together but very far away from a destination will have a highly horizontal link with each other because there will be very little difference in latency between forwarding a message to the other node, or sending it by another route.
  + If all a node’s neighbours are very close to it, it will be difficult to determine which is the best one to forward messages to, so all will have a roughly equal low weight.
* When a node receives a seeker message from a downhill neighbour, it should forward to blocks of neighbours at a time, starting with the highest weighted ones
  + Only after a node receives failure reports from all the neighbours it has forwarded the seeker message to so far, will it forward the message on to more of its neighbours.
  + A node can be imagined to cover a region of a topography rather than just a single point
  + When a node receives a seeker message it first spills “water” over the lowest ridges within its region. If the water level rises above those ridges, the node spills water over higher ridges within its region.
  + If the highest ridge is flooded, the node knows that the destination cannot be reached.
  + A ridge (neighbour) is not flooded until the node receives a reply from that neighbour saying the destination cannot be reached through it
  + A node reports that a destination cannot be reached through it (a failure report) when it receives a seeker request and has no neighbours remaining which it has not received a seeker request or a failure report from.
* I think it is possible for infinite loops to form if nodes only send seeker messages to some neighbours and wait for concrete failure responses from them before sending to more neighbours
  + Actually if anything I think there could be an endless wait because of a loop
  + But I think even this can be avoided
  + If a node is currently waiting for responses from some of its neighbours but has not yet sent out the seeker message to all its neighbours, and it receives an incoming seeker message from a new neighbour, it should send out seeker messages to more neighbours.
  + This still isn’t sufficient to block loops
  + Perhaps nodes that receive a seeker message from their highest weighted neighbour should just send out seeker messages to all their other neighbours
  + If a node receives a seeker message from a neighbour which isn’t its highest weighted neighbour it should send on the seeker message like a normal message – probabilistically according to neighbour weight with a small chance of sending more than one message.
  + If this normal message fails, it should then send out seeker messages to all its neighbours.
  + A loop is formed when a message reaches a node that was an earlier node in the path that message took. This would be a back-edge in graph lingo.
  + There is no way for the receiver of this message to know that it originated from itself, because the history of a message’s path is not carried with the message.
  + When a node X that has already forwarded a seeker message to Y for which it awaits a response receives another seeker message from Z (perhaps its last remaining neighbour), its behaviour should be to notify Z that it has received the message, but wait for a response from Y before propagating that response back to Z.
  + If X’s message went to Y, then to Z and then back to X again, X will be waiting forever for a response from Y.
  + These loops must be avoided somehow because it is possible that the destination is never found and all the loops will go on forever.
  + I don’t think a path of all highest-weight links can form a loop
  + If nodes only forward seeker messages that are not from their highest weighted neighbour to their highest weighted neighbour then I think loops will be unlikely.
  + If a node that has forwarded a seeker message to its highest weighted node does receive a seeker message from a new neighbour, it should push out the message to all its neighbours (including back to the neighbour it just received the second message from), just in case the new message is forming a loop.
* Zutha should allocate quotas of virtual gold to money handler in exactly the same way that it does for Zutha dollars
  + The Zuthaverse Gold Coin should be pinned to the Zutha dollar in value
  + If the value of the Zuthaverse gold coin falls, there is too much of it and Zutha should reduce the quotas of money handlers that manage Zuthaverse currency such that they have to pull virtual gold out of the Zuthaverse market to meet their minimum holding requirements.
  + Conversely, if the value of the Zuthaverse gold coin rises above its index value Zutha should grant additional quotas to virtual gold money handlers
  + When players pay gold to NPCs, the NPCs should hold some of it to purchase goods from players or other NPCs, but put any profit they make in a bank.
  + The gold NPCs deposit in a bank should be exchanged with money handlers for Zutha dollars and these Zutha dollars should go into Zutha’s account.
  + When an NPC is first starting out in a business, Zutha will give it a sort of loan by buying gold from money handlers and giving it to the NPC. The NPC will then pay Zutha back over time as it makes profit. Actually the NPC will continue paying Zutha forever in a similar way that real people pay taxes to governments. Real people in the Zuthaverse will not have to pay taxes though.
  + Natural resources can be mined by players. A player could even mine more gold and make his own gold coins out of it. This new wealth would be owned by that player and would not constitute a debt to anyone.
  + If virtual gold is created by players mining it, then there will be more gold in the system than is accounted for by money handlers’ debt. Thus if all players were to sell all their gold to the virtual bank, there would not be enough debt from money handlers to make all the trades automatically. Eventually all money handlers’ virtual gold quotas will fall to zero, they all will have bought enough virtual gold from players to meet their debt, and there will still be gold left over in the economy.
  + If there is absolutely no demand for virtual gold at this point, players left with the excess virtual gold will not be able to get rid of it
  + If there is any demand for virtual gold though, players will trade the virtual gold that still exists with each other for virtual goods or real money.
* I discovered a paper (<http://ghost.lesiuk.org/AdHoc/adhoc>) which describes a very similar routing algorithm to the one I’ve been developing called Temporally-Ordered Routing Algorithm (TORA). It even uses the metaphor of water flowing downhill to describe how it works.
* I think there should be a clear distinction between uphill and downhill links
  + The fact that uphill links have negative weights is an appropriate and sufficient distinction
* Horizontal links should also fall in their own category
* The behaviour of a node X that receives a remap message for destination D from one of its downhill (or horizontal links):
  + X first forwards the message to all its other downhill and horizontally connected neighbours if it has any
  + If X receives a message to D from one of its uphill links it will continue waiting for a response from its downhill and horizontal links
  + In order for this to work it must be impossible for a message to flow continually downhill but end up flowing in a loop.
    - It might be possible for an endless loop to form through horizontal links, but these are not uphill links; messages X receives through horizontal links at this point will be responses to the messages it has already sent out.
    - Actually X may receive a new remap message through a horizontal link it just forwarded a remap request of its own through. This would happen because the horizontally connected node sent the remap message to X before receiving X’s remap message.
    - If X receives a remap message through a horizontal link it will mark that horizontal link blocked.
    - The neighbour on the other end of the horizontal link may still think that X could provide a route to D.
    - It is possible that the message X receives through the horizontal link was sent by X itself, forming a loops
    - The only way I can think of that these loops can be detected is if messages carry information about which nodes they have passed through
    - This could be done without compromising location privacy because nodes could attach a number to a message they send out that is meaningless to all but themselves. If they see this number again, they know a loop has formed.
  + If X receives failure requests from all its downhill and horizontal links, it will forward the message to all its uphill links
  + If, at this point, X receives a message from an uphill link, it will reply to the message indicating failure because X has exhausted all its downhill links and a message coming from uphill is asking about the connectivity of the old downhill route through X. Such a message may also have been sent by X itself, so it needs to be rejected immediately with a failure response.
  + Tracking numbers would only have to remain on a message while it is travelling along downhill or horizontal links.
    - When a node propagates a message to its uphill nodes it need not retain or attach any tracking information, either from itself or other nodes.
    - If X receives a remap message from a horizontal link once it has propagated the same remap message to its uphill links, it will treat this horizontal link as an uphill link and thus send a failure response because X has exhausted all its downhill and horizontal links.
    - It must be assumed that no loop can form such that a message travels always uphill from X, but ends up coming back to X from “underneath” through one of X’s downhill links.
    - If this is true, then X (and all nodes “below” X) need not worry about messages going upwards from X reaching them from below
    - Thus once a message passes uphill it cannot subsequently form a loop and so tracking information can be forgotten
    - So tracking information is added to a node whenever it is sent downhill (including along a horizontal node) and all tracking information is removed when the message is sent uphill.
  + Messages that are incoming along horizontal links are considered to be coming downhill from a higher node
  + Messages that are outgoing along horizontal links are considered to be going downhill towards a lower node
  + When X is waiting for responses from downstream neighbours the direction of current through X is considered to be downhill
  + When X is waiting for responses from upstream neighbours the direction of current through X is uphill
  + If the current is flowing uphill through X and X receives a message that is going downhill, X returns a failure response, telling the sender of the message that it cannot continue through X
  + If the current is flowing uphill through X and X receives a message from one of its downstream neighbours…
    - Actually this should never happen because the current will only start flowing uphill in X once X has received responses from all its downstream neighbours that they are blocked.
  + If the current is flowing downhill through X and X receives a message from a downstream neighbour Y, X will mark Y as a blocked route, but will wait until it has received responses from all its neighbours (including upstream ones) before it replies to Y to indicate whether D can be reached through X.
    - It should not be possible that the message from Y has passed continually downhill from X because, after all, it is now coming uphill.
    - If a message passed from X downhill to Y, downhill again to Z, and then uphill to X, this does not form a loop because Z had to switch the direction of the current flowing through it and when it did so, it would have notified all its uphill neighbours, including X and Y, that D cannot be reached through Z.
    - At this point, Y may still have a downhill current and be waiting for responses from other downhill neighbours than Z, but it is not waiting on X to tell Z that X is blocked (and it isn’t necessarily) in order to know that Z is blocked.
  + If the current is flowing downhill through X and X receives a message from an upstream neighbour Y, X will do nothing until all its downstream links are blocked, unless it detects a loop. If the message from Y includes tracking information that identifies it as sent by X, then X will reply to Y with a failure response. This failure response should cascade back to X, the originator of the message. The link through which this failure response gets back to X will now also be marked blocked.
  + If X receives failure responses from all its downstream neighbours, it will change the direction of its current and send messages to all its upstream neighbours. All its upstream neighbours will then know that the path through X is blocked.
* The water level will rise as slow as the longest uphill path
* How does the destination host’s reply get to all the nodes that need to receive it after a topography remapping caused by a link break?
  + A node X that forwarded the message towards the destination host D in the normal way – to one (and with a small probability two) of its downstream neighbours probabilistically by link weight – will forward D’s reply only to the upstream neighbour that sent X the initial message
  + A node X that has a full downstream current (had sent the message to all its downstream neighbours) when it receives the reply from D will forward the reply to all its downstream neighbours except the one from which it receives the reply from D.
    - X will have kept track of when (if at all) it received a failure response from each of its downstream neighbours
    - For downstream neighbours that returned a failure response after X had sent the message that reached D, X should add a delay header to the messages it sends these neighbours that tells them how much time to add to the response time to reach D through X
    - For upstream neighbours that sent X the outgoing message after X had already received the message from a downstream neighbour and forwarded it on to all its downstream neighbours, X should similarly add a delay header to the message so these upstream neighbours know how long it would have taken to get a response from D through X had X forwarded their message when it received it instead of earlier, when it received the message from a different neighbour.
  + A node X that has an uphill current (had sent the message to all its uphill neighbours) when it receives the reply from D will forward the reply to every one of its neighbours except the (upstream) one from which it receives the reply from D
    - For X’s downstream neighbours the delay header X specifies will be negative because the messages from X’s downstream neighbours would have gotten to D faster had X forwarded them immediately to its upstream neighbours instead of waiting for failure responses from all its downstream neighbours first.
    - X will have received a failure response from only some of its upstream neighbours. For these, X should add the appropriate delay to the delay header of the reply from D that it forwards to them.
    - Uphill neighbours that sent the outgoing message to X while its current was going upstream would have received a failure response from X. They should still receive the reply from D through X though because the topology has now changed and X is now upstream of those neighbours, or horizontal to them at least
    - X should also send out D’s reply to all the upstream neighbours it hasn’t yet received a response from. This is because X’d distance from D has changed and they need to know this information. X need only report its own distance from D in terms of weight to these nodes; it does not have to add or subtract a delay to cater for when X received their message (because it never did).
  + The message X receives that has come from D will already have a delay header. X should add or subtract an appropriate amount to this header for each of its neighbours that it forwards D’s reply to.

# Thu 17-Feb-11

* If a region of the network is “submerged”, messages that originate within it will “float” to the top by all upstream links from every node.
  + This will mean that a second message to D, message2, will take the same routes as message1 – the message that has caused the region to be submerged
  + It is likely that message2 will meet the reply to message1 on the way out.
  + Where message2 meets a returning message1, message2 should start flowing down by a single route instead of being propagated up to all upstream nodes as it had been. This is because the reply to message1 will notify the nodes it passes back through of the best route to D.
* A region only remains submerged until a reply is received from the destination host D, or until the message to D times out, which implies that D has dropped off the network
  + If D has dropped off the network, every node in the network should be waiting for a response from D and all will timeout (not all at the same time though).
  + When each node times out waiting for D to respond, it will wipe the whole topology associated with D from its memory.
* When nodes whose current is currently flowing uphill receive a reply from D through neighbour A, they will make A into a downstream neighbour and give it a weighting based on the information they receive in the reply from D. They will also turn all their downstream neighbours into upstream neighbours and wipe weighting information from those links.
  + As the reply from D propagates through a submerged area, the elevation of the terrain within the submerged area will be inverted from the outside in, coming from the direction of the reply from D.
  + At all times during this process, there should be a route from any node to D. Nodes that are submerged will propagate messages through all uphill neighbours, while nodes that are above water will have a single best downhill path to follow.
* Horizontal links should have a weighting of zero
* Horizontal links should not be followed when a link with a positive weighting is available
* Tracking information does not have to be stored with messages until it is known that a blockage has occurred
  + If a node X has sent message m through downhill neighbour Y and then receives m back through an uphill neighbour Z (could be through a horizontal link), then X should send tracked messages down all its downhill links, including to Y again.
  + If it turns out that the message that arrived through Z had travelled constantly downhill from X through Y, this is a loop that X will detect with the second tracked message it sends to Y.
  + X will only receive a message from an uphill neighbour if there has been a blockage somewhere below it.
  + X needs to tell Y if the message X received through Z gets blocked (because it forms a loop), because otherwise Y will be waiting forever for at least one downhill neighbour to respond to its message – the one that carried the message that ended up back at X via Z.
  + This system will mean that Y must “overflow” a second time by propagating X’s message to all its downstream neighbours yet again. In fact all nodes below Y that still have a downhill current would need to overflow again in order to make sure X’s tracked message will get back to X if a loop has formed
  + This is very inefficient. It might be worth having all nodes put tracking information on every message.
    - This would add a lot of overhead to messages that take a long path, and this overhead would be incurred for every single message, whereas the inefficiency involved in the scenario above would only occur in the exceptional cases where a blockage has occurred AND X happens to receive a message from above
  + I just realised X could receive a message from above if no blockage has occurred because the message took an additional path from the source and is merging with another path at X
    - Messages should be marked differently once a blockage has occurred
    - Actually they already are because messages that have been sent from a node that is “overflowing” will have tracking information with them
  + Another disadvantage to storing tracking information always is that it provides information about how far away the source is – the more tracking numbers are on a message the farther away the source is
  + So I think the best way is what I started with: that tracking information is only added to a message when a node overflows.
  + If a tracked message reaches a non-overflowing node from above, it will overflow itself, causing all nodes below it to overflow again with the added tracking number.
* When a node X discovers that its best link has increased in weight, X should broadcast this to its neighbours so they can update the weight of their link to X
  + This should not happen for trivial increases in weight; there should be a minimum degree of increase that will trigger X to broadcast the higher weight
  + The broadcast of an improved route will propagate outwards to all nodes who will change their best route as a result (plus one layer of nodes who will not change their behaviour as these are the ones that stop the propagation).
* When a node X first discovers a new link, it should assign a weight that is less than what it would be purely by the latency to D through that link
  + If the link remains connected for a long time, X should gradually increase its weight
  + Once the link becomes X’s highest weighted link, X should broadcast its new “height” to its neighbours so they can adjust the weight of their link to X accordingly
  + When a link is new, the region of nodes surrounding it that will point towards it will be small
  + The longer a link lasts, the larger will grow the region of nodes around it that will point their highest weighted downhill link towards it
  + This will avoid the issues the internet has with “flapping” links because a link that flaps on and off will only be dealt with by the two nodes immediately connected to it. Only after it settles down and remains connected for a substantial period will its sphere of influence begin to increase.
* Nodes should always forward messages to their highest weighted neighbour.
  + With a small probability, they should send an additional message to one of their other downhill neighbours probabilistically according to weight.
  + These additional messages will allow the node to learn of improvements in the weight of its secondary downhill links
* When a node X receives a reply from D it should forward it to all the neighbours that sent it that message
  + If X is overflowing there may be downstream neighbours that have not sent it the message yet (which would indicate they are blocked).
    - X should only forward D’s reply to these neighbours if they do eventually send X the message (indicating they are blocked)
    - If a downstream neighbour responds with a reply from D of its own, it does not need to be forwarded X’s first reply from D
    - Actually X should immediately forward D’s reply to all its downstream neighbours as this will allow them to determine if X is now the closest of their neighbours to D
    - X may also have received the outgoing message from non-overflowing upstream neighbours. X should forward D’s reply to these also.
    - Actually X may have received the outgoing message from overflowing upstream neighbours as well. These should also receive D’s reply.
  + If X has uphill current there may be upstream neighbours that have not sent X the message.
    - X should only forward D’s message to upstream neighbours that did send it the outgoing message
    - If upstream neighbours send X the outgoing message after X has already received a reply from D, X should forward the reply to them with an appropriate delay header
    - The uphill neighbours that have not yet send X the outgoing message should all have received the outgoing message from X, so they should eventually return either the reply from D or the outgoing message to X
    - If an uphill neighbour returns the reply from D to X, X need not send it the first reply from D it received
    - Actually X should forward D’s reply to ALL its uphill neighbours (if its current is uphill) because this will allow those neighbours to discover whether the fastest path to D is through X still.
    - Overflowing nodes form a barrier outside which D’s reply will not pass – except to uphill neighbours that sent X the outgoing message
* There should be no distinction between a failure response and the message itself
  + When a node X receives a message from below it knows that link is blocked
  + If X has not yet overflowed, it should then do so
  + X will forward D’s response to any downstream neighbours it receives the message from
* Argument Maps could be built from a hierarchy of ownership such that each Assertion item inherits the reputation – or validity – of all its child (dependent) Assertion items
  + Objection child items should have a negative effect on the reputation of their parent Assertion item
  + If should be possible for the same Assertion item (Reason or Objection) to be the child of multiple parent Assertion items and pass up its full reputation
  + The flow of “reputation” for an Argument Map is already looking substantially different from the way reputation behaves normally for Zutha items
  + If no one owns Assertion items, then many of the exploitation opportunities that needed to be avoided in determining the flow of reputation, will be absent, so the flow of reputation (or “truth” perhaps) in an Argument Map could be far more flexible
  + The items used in an Argument Map should be a distinctly different type of item. Zuth should still be used to “invest” in them, and influence should determine how much users can invest, but the aggregated metric that rates the item should be called something other than reputation; perhaps “validity”.
  + It should be strictly impossible for any type of item other than an Assertion item to own an Assertion item. This is because the “validity” metric used to rate Assertion items will not be compatible with reputation.
  + Validity can apply its full influence multiple times to multiple parent Assertion items (usually from two different Argument Maps)
  + Positive validity in an Objection Assertion item subtracts from the validity of the parent item
  + Negative validity in an Objection item should have no effect on the parent Assertion item
  + Only positive validity in a Reason Assertion item should add to the validity of the parent Assertion item; negative validity is not passed up to the parent item
  + Investing negatively in an Assertion item can only counter any positive investment in that Assertion item in terms of that Assertion item’s influence on the larger Argument Map
  + There should be a container item type which falls under this same category of Argument Maps item types, which acts as the parent of a set of co-premises
  + Perhaps the container item could be called an Assertion Group
  + An Assertion Group requires all of its children (the set of co-premises) to be “true” (have positive validity) in order to have any effect on its (the Assertion Group’s) parent Assertion item
  + The validity of an Assertion Group should be the maximum of 0 and the validity of the lowest validity child
  + I’m not sure whether it would make sense to allow direct investment in an Assertion Group itself.
    - If this was possible, the validity of the Assertion Group would probably be its own worth plus the lowest validity child (provided all children have positive validity).
    - It wouldn’t really make sense for one of the co-premises to be false but for the whole Assertion Group to be true because there is direct positive investment in the Assertion Group itself
    - I think therefore that it should not be possible to invest in an Assertion Group
  + An Assertion Group should not be an item which users interact with directly. They will usually edit the Argument Map through an interface designed specifically for Argument Maps and this interface will show co-premises as a grouped set of direct children of their Assertion Group’s parent.
  + Actually the Assertion Group may not be necessary at all. Co-premises could instead be modelled with a multi-role association.
  + There need be no distinction between the behaviour of a single child and a group of children consisting of a set of co-premises
    - The rule for passing validity to a parent would be the same rule as for passing validity to an Assertion Group as described above
    - An Assertion item inherits, from each group of child Assertions (where a group can contain only a single item), the maximum of 0 and the minimum child validity in that group
    - If the child group is a Reason, the inherited validity adds to the parent’s validity, while if the child group is an Objection, the inherited validity subtracts from the parent’s validity
  + The association used to link Assertions with zero or more co-dependent groups of 1 or more child assertions should be specific to Argument Maps. This association is not ownership; it does not confer reputation from owned items to their owners, nor should Assertion items receive income from their child Assertion items.
  + Real money donation to an Assertion item (both positive and negative) should be considered a donation to Zutha. The Zutha community should determine the default amount to donate back and have the privilege of keeping whole donations on an individual basis if there is sufficient consensus to do so.
* How would a Zutha network handle being partitioned into two completely separated parts?
  + From the perspective of one of the parts, many Hosts will suddenly go offline
  + At the beginning of the next timeframe these missing Hosts will be removed from the Hosts list
  + New Hosts will be elected for each controller group to make them up to their required number of hosts
  + Once this has happened this partition can continue operating as normal
  + Provided each partition is a sufficient proportion of the original network, it is likely that all data will be kept by both parts because every bit of data should have been spread across Hosts evenly distributed throughout the network
  + While the two partitions are in the process of separating, many messages will not reach their target Hosts, but the requests these messages carry should still be able to be managed by the Hosts that do receive them.
  + If the number of Hosts in a controller group falls below some critical number, that controller group should stop accepting messages until their numbers are replenished. This rule would be enforced by all nodes in the network; they would not accept confirmations of requests by less than a minimum number of Hosts.
  + Actually the only power gained by a controller group of even just one node is the ability to select its own timestamp for the requests. It could probably block requests from specific users if it wanted to as well, but neither of these pose severe dangers to the security of the network, so I see no reason why a controller group couldn’t keep on operating with only one available member.
  + If a whole controller group was on one side of a partition, the other partition would have to wait until the next timeframe before items could be created or modified that were under the control of that controller group
  + Essentially it seems like both isolated networks would go on operating as usual. They would both start with the same data as the original whole network and would begin to deviate from each other over time.
* The harder challenge would be how to merge two network together
  + This problem would probably have to be treated the same whether the networks to be merged were once a single network or whether they arose entirely separately
  + Essentially it would be a matter of merging two very large topic maps
  + Many items could be merged seamlessly using various identifiers including Published Subject Identifiers, unofficial URL identifiers etc.
  + If the two networks used to be connected, many of their items could be merged by their Zutha item ID. If they have been separate for long enough, though, they may both have added new Hosts to which they assigned the same Host IDs as new Hosts in the other network. This would mean two completely different items could end up with the same Zutha Item ID.
  + One of the networks to be merged could be designated the master. It would take in all the events of the other network, give them all new IDs and run them in order along with its own Events, starting from the moment of separation. If the networks were never the one network then all events from both networks could be run right from the moment of their creation.
  + This process would cause a great deal of conflicts which I’m not sure how to resolve. It might do pretty well at merging sets of comments about the one item from the two different networks. It would do poorly at merging modification events. Probably the first network to change an item after the split would have all its subsequent changes accepted, while all the changes requested for that item from the other network would be rejected.
  + Perhaps the hardest thing to merge would be the ontologies. If the networks grew from the ground up entirely separately, fundamental item types could behave entirely differently in the two networks, having different fields and different allowed associations.
  + I don’t think it would really be practical to automatically merge two completely different networks that originated differently with different ontologies
  + A better approach than merging the data and ontologies of two networks would be to migrate the users from one network to the other, and then use various bots to transfer the desired data from the other network into the primary one.

# Fri 18-Feb-11

* zutha could compete with eBay and eLance/vWorker

# Sat 19-Feb-11

* Real-time auction for taxi/bus service. You post a desired trip time, origin and destination and taxi and bus owners (or anyone with a vehicle) offers their price. You see a list of offers, along with the reputation of the driver and reviews by past customers, and you accept an offer that is acceptable to you.

# Sun 20-Feb-11

* Can you maintain location privacy but still be identifiable in Z-space?
  + If your client identifies itself as belonging to you and notifies its neighbours, including physically nearby zutha clients, of its location, then people near you would know who you are. Neighbouring clients do not pass on any information about you to their neighbours though so your location would not be broadcast to the whole network. People and devices who see you could tell others where you are, including posting your location in a public location, but this is no different then what is already possible and common in daily life.
  + If you meet a person who is not broadcasting their identity and you want to know their identity you could use face recognition on imagery from your own device or nearby public cameras. It would be far easier to ask who a particular person is that you actually meet than it would be to keep track of someone continuously using public cameras and face recognition. If cameras are owned distributedly then a protocol of not actively sharing people's location could be enough to make it very hard to discover the current location of someone who isn't broadcasting their location. If an entity had enough computing power they could stream in all the public camera data to their own servers and run real-time face recognition algorithms on all the people in all the video data. It seems practical that a powerful entity like a government could use this technique to track and search for individual people, but not keep an active location database of all people. I think the most important thing - and probably the best that can be hoped for in terms of real world privacy - is that an arbitrary individual's location is not displayed publically against their will. I think this would only happen if it became the norm for 3rd parties to discover and share in public spaces like the Zuthanet, the locations of people they find in public surveillance data. It would be necessary for many 3rd parties each working on parsing the surveillance data in different areas to all share their parsed data (the locations of all the people they identify) in a unified format, in order for it to be possible to search for the current location of an arbitrary person. If it was socially unacceptable to share such data then such a scenario is unlikely to occur. Would it perhaps be worse though if it required a lot of resources to discover someone's exact location because then you could not be sure if someone powerful was watching you, but might feel more complacent then you should because you know it is relatively difficult for someone to lookup your location? It might actually be a more desirable world if everyone's location was easily accessible to everyone. Then the powerful wouldn't have an edge - that edge at least. It seems inevitable that privacy in the real world will become very difficult with the increasing proliferation of cameras. Perhaps it would be better if the complete lack of privacy in real world public spaces was unambiguous. Real privacy is going to have to be found in virtual environments anyway, I think; why not make this clear to everyone, including the naive, by publically broadcasting a real time video stream of every person on their public profile whenever they step into a public space? This would probably send people the message that there is no longer any privacy at all in public spaces.
* 3rd parties can request actions be done by your user. You are then asked to accept or reject the requested action. For example a merchant can send a request to zutha asking your user to donate a particular amount to an item you wish to purchase from them. Once you accept, zutha will make the transaction and the merchant will see it has gone through and can give you the product you bought. This functionality should be part of the core Zuthanet standard.

# Mon 21-Feb-11

* A proportion of zuth investment is permanent.
  + This will leave a residual worth in every item that ever receives investment.
  + Real money donations should also permanently add to item worth in money terms.
  + Zuth investment should have the same mechanics as money investment. Owners should decide how much of the "donation" to keep by default and would be allowed to keep all of any donation if they choose. The owner never receives any zuth though; only the reputation that comes from zuth remaining invested in their item.
  + If money investment has the same mechanics as zuth investment it will be less obvious that the investment/donation scheme has been designed to get around gambling laws because it will look like real money investment is simply introduced with the standard zutha way of doing things rather than with a different model that is clearly a contrived way of making investment look like donation.
  + The proportion of investments that owners keep expands as the value of the investment increases. Thus sellers that sell high give a greater amount to owners in absolute terms. The amount of money given to the owner is still fixed because it is just a fixed proportion of all money currently invested in the item.
* You should be able to search for items, including Search items, that contain a specific set of items.
  + e.g. a search for items that contain both PayPal and Google Checkout should find any Category items of which they are both a part (such as Internet Service, e-commerce, Company etc.) as well as user created Search items that can act as more specific categories. For example PayPal and Google Checkout might match a Search item like "e-commerce companies that provide electronic funds management" which could be constructed from a query like "type=company category=e-commerce provides-service(service=electronic\_funds\_management)".
  + The power of the ability to search for Search items based on their results is that you can find more items that fit in the same category as a set of items you know, without knowing what that category is. So you can find competitors to a set of one or more products and services without knowing how to define what category they all fall under.

# Tue 22-Feb-11

* It is essential that small startup protection agencies can compete in a market with very large incumbents
  + There might be a tendency for protection agencies to want to merge in order to gain market power. If the number of protection agencies shrinks too much, though, the system will become unstable and conflicts will be more likely between the few huge protection agencies that remain.
  + It is essential that competitiveness in the protection agency market not depend on military might. If a small protection agency needed to be able to defend itself against other protection agencies it would not be competitive.
  + The social norm should be that protection agencies are not permitted to fight each other. They must accept the decision of the court that is selected automatically, based on each protection agency’s preferences, to resolve any dispute between them.
  + A social norm like this can be very powerful provided the number of protection agencies remains large. If any one protection agency breaks this social norm and tries to bully another protection agency militarily, the social expectation would be that all other protection agencies would oppose them. They would have several incentives to do so. Firstly, their reputation would be at stake because people would criticise them with negative investment on Zutha for failing to fulfil their duty as a protection agency by policing other protection agencies. Secondly, it would undermine the social order which is fundamental to their profitability if the social norms that maintained peace between protection agencies was broken. Finally, they would have a socially acceptable excuse to cripple, or at least harm, a competitor.
  + If a startup protection agency does not have to worry about competing militarily with the existing powerful protection agencies, it can compete by innovating in customer service.
  + A small protection agency would not be as effective at defending a customer against physical threats by itself, but it can effectively outsource this task to all other protection agencies by offering to pay other protection agencies for rescuing their customers from physical harm or catching criminals who do their customer’s harm. It should be pretty easy to prove that they have done this in a society with prolific surveillance technology.
  + A small protection agency could still be effective at providing physical protection if they only operated in a small geographic area. A startup protection agency can start small in this way and be competitive in a small area, and then scale to larger geographic areas as they become more popular. They would probably have an advantage in the small area they cover because they would specialise in providing protection in that area and could come to know to specific needs of customers in that area better than a large company who spreads its resources over a large a diverse range of customers.
  + Some protection agencies would operate purely in the legal sphere and would completely outsource physical protection of their customers. Small protection agencies could be competitive in the legal sphere as well because they could specialise in specific legal areas and be better than their large competitors in those specific areas. Customers could hire multiple specialised protection agencies for each specific area that they want legal protection in.
  + Protection agencies might find it efficient to outsource their legal services to many specialized protection agencies in different legal fields. This would allow customers to pay a single protection agency who would be responsible for identifying the best specialized protection agencies to cover each specific legal area. This model provides the benefits of one large institution to the customer (less complexity, mainly) as well as the benefits of many small institutions to society (robustness, competition etc.).
* HTTP Hosts
  + #.zutha.com is an HTTP client for a single HTTP compatible host with id #.
  + Redirect users who connect to zutha.com to a random HTTP client like 0.zutha.com or 1G.zutha.com or 12.zutha.com.
  + 3rd party hosts should be able to provide their own custom http interface to the Zuthanet which will be accessible from #.zutha.com where # is the host id of their host.
  + The algorithm that selects an http host to redirect to should prioritize hosts that are geographically nearby the client device (i.e. the device running the web browser).
  + HTTP hosts should serve view requests directly, without redundancy. They must still send edit requests to the applicable controller group to be verified and timestamped. However, they may return preliminary confirmation to their http client of the success of their edit request before the request has been officially confirmed by the network.
  + HTTP hosts can have meaningful names that are aliases for their host id. 3rd parties could even set up their own domain name which their #.zutha.com address would redirect to.
* HTTP Host Reputation
  + A host should not lose any reputation if it gives sufficient warning to the network before disconnecting.
  + HTTP hosts should gain reputation by direct investment from users who like their interface.
  + I don't think the HTTP interface would be able to provide an automated way of rewarding and punishing hosts on performance. Performance will be a factor in attracting appreciation from users such that they will praise your host with investment though.
  + HTTP hosts will still be policed on correctness when they are part of a controller group but that wouldn't stop deliberate deception of users by HTTP hosts, only accidental errors. There wouldn't be a lot if point deliberately deceiving your users though because you don't know who they are; any user can choose (or have randomly selected for them) any of a large number of independent HTTP hosts.
* Only the bottom items in an Argument Map hierarchy should be invested in
  + If an Assertion item on the bottom of the hierarchy has children Assertions added to it, any existing investment in it should be liquidated

# Wed 23-Feb-11

* cloak: green (Greens), blue (Democrat), red (Republican), gold (Libertarian). Green and gold meet at the buttons.

# Thu 24-Feb-11

* Client hardware design
  + The operating system should be built in a modular way so that each layer of it can be constructed in different ways by 3rd parties
  + At the most basic level there needs to be firmware to interpret input from hardware
  + There should be a layer which is designed to hold many different conversion algorithms between the direct input from the real hardware and various arrays of virtual hardware
  + It should be possible to plug in arbitrary virtual machines into the hardware virtualization layer
  + Anyone should be able to make their own virtual machine type – such as an Android phone, Apple iPad, Xbox, Asus laptop, or any custom virtual device – and install it onto their client over the virtualization layer
  + On top of the virtual machine one would install an operating system. In many cases there would be only one operating system available, as in the case of an Xbox or iPad.
  + The Zutha Client should just be another virtual machine with the Zutha client software running on top of it as its operating system
  + A user can choose to run any number of virtual machines of arbitrary types at once
  + A virtual machine might be interacted with as a virtual device that you interact with in Z-Space while wearing HUD glasses (which would be your client)
    - For example you could create an iPad virtual machine and a virtual iPad would appear in Z-Space before you which you could move around in your field of view and interact with by pressing virtual buttons in the air or giving voice commands
  + Some virtual machines, like the Zutha client software (or a distribution of it that I’m imagining at least) would encompass the whole Z-Space interface because they would be designed to run on that hardware substrate.
    - Instead of a virtual 2D display, you would interact with 3D objects that can exist potentially anywhere in your field of vision

# Fri 25-Feb-11

* Competition items ranked by item worth in a particular attribute

# Sat 26-Feb-11

* Item owners should have veto power over edit permissions for their item
  + Different types of items should have different default permissions
  + If an item has no owner its edit permissions will simply be the default for that item type
  + If an item has an owner, the edit permissions will start as the default for that item type. The owner can then modify these settings as he/she desires.
* Division of ownership should initially be specified by a single user with sufficient privileges
  + Once owners are specified for an item, they are the only ones who can directly modify the owners and ownership allocation
  + Anyone can open an ownership dispute for an item. Ownership will then be determined by a decision item.
  + An ownership decision process could have a specified expiry time, after which the ownership allocation determined by the decision item becomes a fixed setting again which can be modified directly only by those owners.
  + It should also be possible to permanently change the method of determining an item’s ownership to be the decision item process, where anyone can vote/invest in the decision item in proportion to their influence over the decision item.

# Sun 27-Feb-11

* Make currency trade an automated combined double auction.
  + Money handlers are forced to offer trades of zutha dollars at the indexed rate - as determined by decision markets.
  + Actually zutha's base exchange rate for a given currency could be determined by a prediction market for the near-term future market exchange rate of that currency in the zutha hosted combined double auction.

# Mon 28-Feb-11

* A high reputation user should be able to freeze the funds of a user for a limited time until the whole zutha community can decide whether to release the funds or sustain the freeze. This ability would be useful if a wealthy user committed a crime and there was only a limited window in which to freeze his zutha funds before he got them out.
* The stuff someone downloads from the Zuthanet is public knowledge. An account should be required to access the material. A social norm could be developed that expects users to donate to items they download if they can afford to and if they like it enough. This social norm would not be enforced for individual items, but users who donate a lot would be rewarded by reputation and users who are known to have plenty of money but who very rarely donate to the items they read and download could be punished through reputation.
* Virtual worlds can be evolutionary breeding grounds for competitive social institutions.
  + Some discovered institutions will be applicable, either directly or by analogy, to the real world.
  + Many other inventions could occur in virtual worlds as well of course.
  + Life/intelligence on earth will have to innovate furiously in physical and social technologies in order to remain competitive in an arms race with any destructive intelligence that evolves, on earth or beyond.

# Wed 2-Mar-11

* Money handlers can buy larger quotas by paying a higher interest rate
* Interest earned by money owners and item owners is calculated as a percentage of Zutha income, of which interest from money handlers is just one source
* A user's old private keys could be stored on Zutha, encrypted with one of the keys generated from a set of their secret questions. This way, if they lose their client, their new client can download and store all the past private keys so the user can still read all their encrypted data.
* A user’s “Master key” – the one (or perhaps more) that they generate from a collection of secret questions – should be an extremely large number. It can be large enough that it takes a substantial amount of time just to encrypt or decrypt with it because this only needs to be done on rare occasions. I’m thinking in the order of a 16,000 bit number.

# Thu 3-Mar-11

* Updates for every item should only be sent to its current controller group
  + This will make it hard for anyone to get access to the entire Zutha database if zutha does not give them access explicitly
  + Since all view requests must go to the appropriate controller groups, all view requests can be reliably captured
  + The core semantic data will still have to be stored by every host in order to be able to respond to queries
  + Data that is propagated to all hosts:
    - type-instance associations
    - parent-child associations
    - item name, description, current worth, reputation
  + Data that is retrieved on demand from appropriate controller groups:
    - data component of an item i.e. an HTML document or other contained file
    - obscure associations
    - event items, including investment events
    - view items
    - edit permissions

# Sun 6-Mar-11

* duplicate names are individualized by appending a number in brackets. If an explicit qualification is added, the number may be removed. e.g.
  + John Smith
  + John Smith (2)
  + John Smith (3)
  + John Smith (Actor)
  + John Smith (Actor) (2)
  + John Smith (American Scientist)

# Wed 9-Mar-11

* Investment interface appears beneath item when a link is clicked. This can be done without JavaScript.

# Thu 10-Mar-11

* If a host is marked as being under a DDoS, only users with reputation should have their requests forwarded to it. The victim can send out a message to the whole network asking for traffic addresses to it to be filtered. The whole network will then be involved in filtering out the attacks, distributing the load.
* It would not be possible to hide one's location with remailers when connecting to an attacked host because intermediate nodes would only forward requests they can confirm as coming from a user with sufficient (probably just greater than zero) reputation.
* An attacker would need to simultaneously disable an entire controller group in order to block any content. As soon as some of the hosts in a controller group become inaccessible though, they will be replaced in the next period with new hosts that aren't under attack. Even if all the hosts in a group were made effectively unresponsive at once, the network would still automatically select new hosts to manage the content previously hosted by the now disabled group. The new group members will be able to send whatever content they have about the blocked subject and will soon obtain all of it from the old hosts once they are able to get a message out to them from beneath the barrage.
* Once a subject area is identified as being under attack, requests to any content in that subject area can be filtered for positive reputation users by the entire network.

# Sat 12-Mar-11

* Controller groups should automatically expand for data items (like videos) that are receiving high demand.
* Any node should be allowed to become a host at any time without any requirement that they remain available for a long period. They will just earn less reputation if they regularly disconnect - especially if they do so without first notifying the network that they are disconnecting.
* A client can offer to become a host and contribute to the distribution of some data items that are assigned to them for as long as they choose to do so. In this way the zutha network can act as a hybrid p2p and server based distribution system.
* Clients should send requests to a subset (specified by the selection algorithm) of an item's controller group. This is particularly relevant for items with large controller groups like high demand data items.
* An item's controller group should be called its "managers". The host group that a request is sent to should be called the request handlers of that request.
* Diplomacy board game over the Zuthanet
  + Moves could be encrypted with everyone else's key at once. When everyone is ready to continue the encrypted moved are passed around and decrypted.
  + The app that people run on their client could provide an integrated video chat tool for communicating face to face with other players.

# Sun 13-Mar-11

* Data items should have an associated set of hosts that have that data. Hosts can voluntarily join this list for any data item.
* The data part of data items is immutable. If someone modifies a video stored on the Zuthanet they will create a new data item and the semantic item that represent the video will be associated with this new data item.
* What about documents that get regular small changes? Every change could create a new data item that represents that version of the document. The semantic item would link to the data item associated with the active version. Internally hosts may represent the data however they wish - for example by defining old versions with respect to newer versions and reconstructing them on demand. The zutha standard only defines that a full document should be returned when a data item representing a particular document version is requested.
* Put a cancel button next to the praise and criticise buttons in the investment interface. This will cancel the process of investing in that item, but whatever has already been invested will not be undone. A user can then start the process of selling his investment in that item if he wants to completely extract his zuth from that item. For small investments the cancel button won't help much because the entire investment will be finished quickly. With very large investments though, it will take a long time for all the zuth to be invested so it will be possible to cancel before it has all been invested.
* How will investments have their starting h values determined when investments occur in parallel over a period of time?
* When a buy request is made, the h value of that investment will be the current h value of the item and enough height for the entire investment is reserved. If the investment is cancelled all investments above it will be shifted down.
* Making buying a continuous process with a user specific and item specific maximum rate has two main effects. It allows users to cancel large investments if they made it by accident or if the market turns against them, and it prevents a powerful user from instantly distorting the state of the market. The latter would be very damaging for decision items because a powerful user could push a decision item to his desired outcome before the opposition has a chance to notice his involvement and respond.
* The official reputation of an item should be calculated only from zuth that has already been invested, but interested parties should also be able to see the projected reputation of the item once all currently ongoing investments finish. They should also be able to see the total eventual sizes of all ongoing investments along with the current size and the owner of each of those investments.
* Comments are associated with an item over a range of versions. When an item is modified, specific comments can be marked as "resolved" and they will stop being associated with future versions of the item.
* All occurrences and associations of an item could be associated with a range of versions. This could be done by giving associations attributes specifying the range of versions for each member item that they apply to. In the underlying topic map, the version items would only have the one association with their semantic item and their other associations would be derived from the associations of the semantic item. Again, though, the implementation would not be defined by the zutha standard. It would just specify that, from the perspective of the query language, all occurrences and associations associated with a specific version should be immutable after that version is created.
* It should be possible with this standard in place to implement, on the client side, a tool to browse the Zuthanet at any historic point in time and to watch a replay of the activity at a particular place in the graph throughout any historic period.
* Items that have ongoing investments will have gaps in the vase sort of like air bubbles. The volume of these gaps should be subtracted from the volume indicated by the current height in order to determine the item's current reputation. When an investment is cancelled, all investments above it have to have their height re-calculated. Cancellation of an investment should have no effect on the value of any other investments or on the reputation of the item (except that it will stop contributing to an increase in the reputation of the item).
* When an item is "deleted" a new version item is created which has no associations or data and which is marked as deleted. The item will disappear from collections and search results. When users visit this item directly by using its id they will see only the message saying it is deleted. From there they can view the historic versions of the item though, so no content is made unavailable.
* Historic data (I.e. version items) should have very large controller groups that are mostly static and only add members when existing members leave the network or reveal that they can't supply the historic data anymore. Hosts can put historic data in long term storage where its potential availability in the network will not have any resource draining effects.
* Hosts should occasionally be expected to prove to the network that they still have a given piece of historical data. If they can't supply it they will be punished and a new host will be added to the controller group for that data item.
* Some items will be posted to the Zuthanet that are offensive to many people. There should be various criteria by which an item can be marked sensitive. Users can specify which kinds of sensitive content they want to be warned about or made unavailable. I can think of no way to provide any form of non-voluntary censorship. All data on the Zuthanet is inherently in the public domain. Even if every host in the zutha network happened to be willing to restrict access to a certain item when the community voted to do so, an independent network would always be able to use to Zuthanet technology to provide the censored content to everyone. If someone wants to make some information available, technology is increasingly going to make it impossible to stop them. I can think of no reason why zutha should try.
* To censor information that it would be destructive for anyone to know (nuclear weapons designs say) would only deprive the weak of that information. Once such information has been in the public domain, anyone resourceful, including corrupt governments and terrorists will be able to get access to it. Leaving such information in the public domain may actually be better because it will the whole world aware of the dangers posed by that information and allow the whole world to work on finding countermeasures to those dangers.
* Information that some people want and other people don't want them to have, such as porn, will be inherently uncensorable once encryption and strong anonymity technology is ubiquitous. If many people want to be able to get porn anonymously (which they already can) then there will be no way to stop anyone from getting porn anonymously. If parents want to stop kids from looking at porn they will have to do it at their end of the network using software on the client they let their kids use. This is almost the only solution already. Censorship at the ISP and country level is still possible in a rudimentary way but it won't be for much longer. Ubiquitous encryption and distributed hosting will soon make that impossible.

# Wed 16-Mar-11

* A host could form one way connectivity with a client.
* The host would broadcast occasionally to let clients know that it is in range of them. If a client wants a message from that host it will send a request to it via a longer route, telling the host in this message that it can hear the host's signal directly. The host can then broadcast the response directly to the client.
* The client should also be able to ask this nearby host to fetch the (signed) responses it needs from other hosts in the network.
* If a host does not want to host a particular piece of content (e.g. child pornography) then it should simply refuse to do so. It will be punished automatically if a user with reputation tries to access that data. But it might also be punished by users if it does host the data. Also, it can report any users with reputation that do try to access the data so that they can potentially be punished by the community. This will provide a disincentive for users to access the data with accounts that have reputation and thus the host will not be able to be punished if it chooses not to host the data to anonymous users.
* Hosts should be able to refuse to accept hosting responsibility for a data item or relinquish responsibility if they have it. Perhaps they should not be punished automatically for doing this. The act will be public so they can be praised or punished by the community depending on whether the community in aggregate supports or opposes censoring of that item.
* I think this latter system would be quite effective at retaining sensitive material that the public believes they should know, but removing material that the community opposes strongly like child pornography.
* There should be a constitution which specifies a consensus about what kind of material should be removed. The expectation should be that if there is any doubt, no censoring should occur. It is important that it not be possible for powerful interest groups to put enough pressure on hosts to get material they dislike removed.
* It should require the consensus of the vast majority of the community for material to be censored. I think such consensus would be available for child pornography but not something like pirated software or criticism of Scientology.
* Perhaps then hosts should always be punished if they fail to host material assigned to them. This way only a large negative community response to them holding the material would be sufficient incentive for them removing it.
* But a powerful interest group might have sufficient power to pressure all existing hosts into removing content they dislike, provided the content is not all that salient to the rest of the community as a form of free speech or other material that should not be censored.
* The fact that the interest group must pressure hosts publically will effectively eliminate their ability to censor content discretely. Even if all hosts do submit to interest group pressure, they can keep the content privately and re-post it later if the community discovers the censoring and decides against it sufficiently strongly to oppose the interest group's censoring influence.
* The Zuthanet should support HTTP requests as part of the standard.
* Every http request should be recorded and associated with the URL it targets. This will allow a count of page views to be calculated.
* Hosts that receive an http request should forward it to the target url and then send the response back to the user who initiated the response.
* It could be confusing to web servers to receive multiple identical http requests from different sources for each request sent by a user.
* For https a user's request only needs to be forwarded by one host because the communication will be secured from manipulation by that host by the https encryption protocol.
* I think it would be unsafe to allow http posts over the Zuthanet because such requests often include sensitive information which would be exposed to the intermediary: the zutha host. Actually such requests are insecure anyway, but still...
* Any http requests that rely on cookies to identify users would also be insecure over the Zuthanet.
* I think dynamic access to the internet through the Zuthanet would only be safe and practical for sites that support https.
* Some content from non https sites could be mirrored by zutha hosts. For example pages that require a login to view could be cached on the Zuthanet by users with an account on that site.

# Tue 22-Mar-11

* There should be a feature defined in the zutha standard that allows sufficient community support to order the network to distributedly decrypt specific messages or crack a specific private key.
* An advanced variant of the selection algorithm could be used to allow each node in the network to independently determine which part of the decryption job they are expected to do.
* Nodes that finish their part early can ask for more work to do to earn more reputation.
* Zutha's reputational incentives would be a powerful way to harness an enormous amount of computing power on demand.
* There would need to be a very rigorous community consensus requirement before the network would be ordered to decrypt someone's private message of crack their private key. One scenario that I think could obtain sufficient support would be the decryption of communications between individuals who have been demonstrated convincingly to be terrorists.
* The fact that who zutha users send messages to is in the public domain should be helpful in identifying connections between terrorists that are communicating via the Zuthanet. There are ways that such connections could be obscured by using many anonymous users though.
* The Zuthanet could also be ordered by the community to track the locations of specific users. The combination of location data and social network data (who contacted who) would be a powerful toolkit for identifying terrorist communications from a few leads. These communications could then eventually be decrypted to obtain more leads. Of course the location data and social network data would also be extremely useful on their own.
* It would be useful if users could post anonymous comments which they can claim from their primary user if the comment is popular. This would make it more palatable for people to make controversial statements. Many such statements that would be valuable information to many people might be suppresses if there is a fear of bullying by opponents of the controversial view. If the maker of the statement is later proved right, they should he able to claim credit for their prescience.
* This mechanism would be useful for aggregating knowledge using the Delphi technique - sharing expert predictions and explanations anonymously over a number of rounds and then aggregating the predictions of all the experts at the end.
* Investors should earn a dividend from their investment that comes from the interest earned by money handlers on that amount of zuth. Holding cash should not earn interest. This will create an incentive to leave money invested rather than sitting around unused. There will likely be demand for safe investments whose purpose is just to earn interest. Highly popular items that are likely to remain popular for a long time will probably be one of the safest places to leave money, along with empty items (where your investment is the first one). These are two types of item that I want to encourage more investment in so this is good.
* Someone could make a businesses by investing in science research capital and letting promising researchers use it in exchange for a proportion of the income they earn from zutha from research they do at that institution.

# Thu 24-Mar-11

* The Zuthanet could provide secure connection to a trusted node who can handle arbitrary http requests. This would allow the Zuthanet to extend robust access to the internet to anyone within range of any Zuthanet node.
* Try to seed a zutha civilisation with a culture that basis competition for status on zutha reputation instead of ability to waste abundant resources. The rich should be rewarded in reputation for the effectiveness with which they use their wealth and power to make their lives and others' lives better. They should be punished in reputation for wasting their wealth on competitions with other rich people to show off their wealth with exorbitant spending on things like (multiple) expensive cars and houses.
* When a client makes a request through an http host, it could specify how up to date it wants the results to be. If the host has a sufficiently recent version the relevant data in its cache it can return this data immediately. If it doesn't, it should send requests to the relevant request handlers according the selection algorithm. A client should be able to ask for proof of the validity of the previous or current request. This should take the form of a sufficient set of signed responses from other hosts to derive the response given by the http host. The oldest of these determines the age of the http host's response.
* Controller groups could take responsibility for every property of an item including its associations.
* Hosts would keep a private cache of a large part of the database but would not receive immediate notification on any changes except on items they are responsible for.
* A host can respond to a request based on its cache, indicating how old the response is based on the oldest bit of data it is using. If some of its data is too old it should request this data from the relevant controller group before responding to its client.
* Controller groups should propagate the most recent changes to their item since the last update at regular intervals (if there were any). In this way, hosts will have a limited upper bound on the age of their data. It will also reduce network traffic for frequently changing, high demand items because it will be more efficient for new changes to be propagated in bulk infrequently to all hosts than requested frequently by many hosts of many controller group members simultaneously.
* To get the most up to date response possible will require asking for unfiltered sets of items from multiple controller groups in parallel. e.g. to find all fantasy books would require all items of type book from the controller group of the book type item and all items of genre fantasy from the fantasy subject item. Actually only the latest updates on those two lists would be needed since most of those lists would already been in a host's local cache.
* A response should always be for a specific timestamp - the timestamp of the oldest data used to generate it in fact. This will allow any response to be verified against other hosts' responses for the same timestamp.
* The concept of asking a host to verify its response is superfluous. The client can just ask for equivalent responses (at the same timestamp) from other arbitrary hosts

# Sun 27-Mar-11

* Prediction market items reward and punish in terms of domain specific reputation.
* When a prediction market expires, the amount a user wins in zuth also contributes permanently to their reputation and the amount they lose from wrong bets detracts from their reputation.

# Mon 28-Mar-11

* Ability to create anonymous users with as much reputation as a high reputation user you own.
* This would be useful for decisions or discussions for which there is likely to be social pressure against certain views. If users could voice such unpopular views anonymously, but with the backing of their possibly high reputation, this could be a powerful way to introduce dissenting opinions into a debate.
* The reputation spectrum could be divided into bands, each of which is assigned a controller group. Each member of such a controller group would generate an asymmetric key pair. They would publically post the public key but only give the private key to users with reputation at least as much as the lower bound of the reputation band they are responsible for. A new key pair could be generated in this way each timeframe. An anonymous user could prove he has at least a particular amount of reputation, X, by encoding some message with the private keys of a majority of the controllers of the reputation band starting at X.
* A user must ask for new reputation keys each timeframe in order to use an anonymous user of that reputation in that timeframe. Users can thus potentially be limited in the frequency with which they are allowed to use anonymous users of high reputation.
* Anonymous users with reputation borrowed from an existing user in this way should be single use entities.
* A user can include half of an asymmetric key in their anonymously posted items or investments. They can then claim the anonymous action later by supplying the other half of the asymmetric key.
* Users should be able to buy anonymous zuth cash which they can invest anonymously using a single use anonymous user that inherits their reputation. A client would facilitate this by masking a number and accompanying it with a request for a certain amount of anonymous zuth cash. This would be sent to a controller group, each member of which would blindly sign the masked number. Each member would check with the others that they are signing the same masked number by comparing hashes. None would know the number itself though. Finally, the requested amount would be deducted from the users account. The user would use the anonymous cash - possibly from an anonymous user - by presenting the number that was blindly signed. This number would have been stored by their client and all these details would be handled transparently by the client of course.
* One-use anonymous users could be created in the same way as anonymous zuth cash: by having several hosts blindly sign a request for such a one-use anonymous user.
* It is important that anonymous zuth and users be only redeemable by the high reputation user that created them. Otherwise this feature would allow the direct - and, even worse, untraceable - transfer of zuth from one user to another, which would facilitate trade of zuth for money. This would undermine the authenticity of zuth and reputation as a measure of merit as opposed to existing wealth and power.
* A single-use anonymous user should be bundled together with anonymous cash. There should be some way to ensure that the only user that can claim such an anonymous user is the user that created it.

# Thu 31-Mar-11

* When a user opens and starts editing a complex object on the Zuthanet like a 3D CAD object, a new fork of revisions should be started by default.
  + If the user has permission, he should be able to commit a series of changes to the main object once he is satisfies with them.
  + An object which has restricted editing access can still be contributed to and played around with by anyone and their changes will be visible on a separate fork for anyone who wants to examine their work.
* Zutha will provide an effective model for the distribution of scientific articles. There will be no need for a long review process before the articles are published. Instead, anyone can put an article on the Zuthanet and the community will rate it on various measures like accuracy and importance. The authors can also earn money directly from investments/donations, cutting out the middle man while also allowing free and unrestricted access to the data for everyone in the world.
* Zutha should donate to all items according to reputation, even commercial ones, but the money stored in commercial items will be inaccessible until the associated product is made free and open source (if applicable).
  + The purpose of doing this would be to give commercial ventures confidence that they can earn sufficient income from Zutha to make their product free.
  + The problem is there will be many products that will never be made free, either because they are physical products like apples that have a marginal cost of production or because they are produced by companies that are too invested in the old fashioned model to change their business model. If zutha donates to such products the money will never be used so this would effectively be a donation to money handlers.
  + Zutha could donate only to digital products that have no marginal cost of distribution. It would be completely wasteful after all to donate to products that will never be free and able to access the donation. There might be a lot of money donated to software that will never be made free, but I think it might be worth it to create an atmosphere of pressure towards putting software in the public domain. If prospective developers saw that huge expensive software packages have a tonne of money available to them if they made their software free, this is likely to cause the developers to more seriously consider making their own software free from the start.
  + The wasted money will not be wasted resources, it will merely increase the buffer available to take the slack if there is a run on the distributed zutha bank. It will also make being a zutha money handler even more attractive and make it possible to charge a higher interest rate.
* Zutha is a crowdsourced distributed knowledgebase

# Fri 1-Apr-11

* Zutha dollars should be stored in accounts.
  + Multiple users can have access to a single account and a single user can have multiple accounts.
  + Every single item could be allowed to have money on it. An account would simply be a semantic construct bit would have exactly the same financial features as every other item.
  + When someone (including Zutha) donates to an item, the money is stored in that item. At regular intervals, that item will pay out the money it holds to its owners and beneficiaries. One benefit of this incremental rather than continuous cash flow system is that cycles of cash flow would not be a problem since the cash would simply flow slowly around the cycle. With a continuous cash flow system a cycle would cause an infinate loop in the algorithm that calculates account balances.
  + I want to make money donated to commercial items from zutha inaccessible until the item becomes free, but it seems inappropriate to lock up all donations to commercial items. I could distinguish between the two if zutha donated to a separate account than everyone else, but this would require having account subitems for every item. Funds could be released on a more arbitrary basis than whether an item is commercial or not. The condition could simply be that the community thinks the item deserves income from zutha. This way, a popular product that has a free and commercial version may be permitted to earn money from zutha, but something like Microsoft Windows may not. Under this system I would be more comfortable with all user donations being unavailable in locked items because the whole community is deciding to lock away those funds. Of course, it would be important to very clearly notify donors when they are investing in a locked item that their donation will not be available to the item owner until the community unlocks the item.
* The zutha id of an item should be a Name in topic map terms with the type Zutha ID. All items should also have a subject identifier of the form zutha.com/0X4G. This will make it easier to merge data from different networks - perhaps after a split and rejoining - because the parts of the database which are common to both networks will be identifiable by those items having the same subject identifiers. New semantic items can be merged by name and manually by users, while non-semantic items like comments will have different subject identifiers and will not be merged.
* How can revisions be kept track of for private encrypted objects on the Zuthanet?
  + The client that makes the change could upload both the diff and the full new version of the data item, both encrypted. For this to be practical it would require data objects to be represented as many relatively small objects. This would in turn require a lot of associations between encrypted components of objects which would provide more information to the outside world about the content of the encrypted objects, but I don't think this amount of information leakage would be significant.
  + I've already decided that every standard (unencrypted) change should cause two items to be made: a diff and the new version of the item. The event item is the diff - or maybe contains the diff. Normally only the diff information would need to be sent and the new revision would be constructed on the host. Actually hosts should not be expected to know anything about how to deal with file formats so I think it should be standard practice even with unencrypted items to upload both the diff information and the complete new data item.
  + Diff information will be important for clients to be able to display the running changes to an item without needing to download the full item at many different revisions. With diff information available the client only needs to download the base revision and the diffs to calculate any revision.
  + A client can lie about the diff by submitting a diff that does not result in the new revision that they submit. There should be a way for other clients to report such discrepancies and submit their own corrected version of the diff. They can check that the diff is correct by applying it and comparing the hash of the result with the hash of the next revision stored on the Zuthanet. For encrypted items the calculated revision would be encrypted with the same symmetric key used to encrypt the next revision on the Zuthanet (they will download this symmetric key from the Zuthanet in encrypted form and unencrypt it with their private key). The hash of this encrypted revision would then be compared with the hash of the same encrypted revision on the Zuthanet - a hash that would be calculated (or at least verified) independently by each host.
* The Zuthanet could be a reliable host for real-time updates for the firmware and software of devices that can connect to it.
  + Example: an open source automobile could have a zutha client built in that would allow security and feature updates to its software. It would verify that an update has sufficient support from the relevant development community to be accepted and then download the update from the Zuthanet, verifying the authenticity of the downloaded file by comparing the hash with multiple high reputation hosts.
  + This system should be secure enough against compromisation to be used even for a device like a car for which the compromisation of software would be a major safety concern.
  + The zutha community could create many customized versions of device software which users can download and install automatically from their device.
  + This paradigm of development is likely to produce very secure software because there will be a huge community of developers searching for bugs and vulnerabilities and any they find can be fixed quickly and immediately pushed out to all devices in the world.
* Virtual worlds hosted on zutha should be built on top of a standardized platform.
  + Objects and rules of a virtual world should be specified semantically.
  + It should be possible to upgrade the underlying virtual world engine without touching specific instances. It should be possible to do such things as increase the resolution and realism of textures and even whole trees, increase the intelligence of AIs and improve the physics engine, all via the underlying platform.
* Anyone should be allowed to host their own virtual world on the Zuthanet platform for free. Hosts that support the Zuthanet virtual world platform will allocate their resources according to the reputation of each virtual world because they will earn reputation themselves according to the reputations of the services they contribute resources to. In order to get other hosts to flag themselves as willing to support your virtual world therefore, you will need to get some initial support from other users about the value of your creation. Some hosts may even offer some resources to experimental works that don't yet have recognition. Of course, you can use your own hosts to get started as well.

# Mon 4-Apr-11

* When a node first signs up to be a host, it will automatically be assigned a few items to manage. It will become a member of these items' controller groups.
  + It may voluntarily indicate that it wants to keep up to date with any other items it chooses. It will be sent secondary updates to the latter, which means the controller group will periodically send it updates, but it won't receive updates in real time.
  + Hosts may volunteer support for categories of items as well as individual items.
* It should be possible to find an acceptable controller group for an item just from its id and the current timeframe key.
  + This could be possible if several hosts were added to a controller group each timeframe. The hosts from the last 2 or 3 timeframes would also be considered candidates for requests to that item to facilitate a smoother transition of control of an item to new hosts.
  + Now when a client first connects to the network, it only needs to ask for the host list for the last 2 or 3 timeframes, whereas, when controller groups changes by just one host each timeframe, a new client would need to know what the timeframe key and host list was for many timeframes into the past in order to know which hosts to make requests to.
* 3rd parties could offer automated data services which are activated by uploading a data file to a particular place on the zutha knowledge base that is controlled by that 3rd party. For example a service could be provided to parse a data file of a specific type and convert it into semantically structured nodes in the zutha knowledgebase.
* The Zuthanet (or potentially 3rd parties) could provide a service whereby users specify an amount of zuth or money which they want to have managed automatically. Their pool of money would be managed automatically by the network using the rules they specify. For example, they could specify that investments should be bought and sold in order to maintain a portfolio with proportionately the same investments as a user or group of users whose investment behavior they like.
  + This feature would effectively allow anyone to invest in an arbitrary portfolio of money managers with no associated fees and without any contracts or other formal transactions needing to take place.

# Wed 6-Mar-11

* It should be possible to encrypt items for access only by members of a group.
  + Every group would have and associated private key. When a new user is added to the group, he would be given access to the group's private key.
  + A category of items could all be encrypted with the public key of a group (actually the symmetric keys of each item would be encrypted with the group's public key and possibly others). If you wanted to give someone access to all the items in that category, you would add them to the appropriate group. They would then discover that group's private key and use it to unlock any of the items among the encrypted items in the category.
  + Example application: Patients store their medical data on the Zuthanet in encrypted form. They encrypt the data with one or more public keys of groups they create to represent varying degrees of access to their data. If they want to give someone access to some if their data, they simply add them to the appropriate access group. They can also give someone access to an individual item by encrypting just the symmetric key of that item with the public key of the user they want to share it with.
* Data items should not have outgoing links.
  + This way, multiple semantic items can link to the same data item without knowing about each other.
  + It will also make it impossible for the holders of encrypted data items to know what they are. This will make it much harder for hosts to sensor which data they host, which will enable controversial data to be hosted on the network more reliably. Such data may include content that the majority of people would not want to be available to the public (e.g. child pornography, weapon designs). For such data to truly be public, though, it needs to be findable by the public through a semantic item. If a public semantic item links to a data item, then both can be sensored by the public and by individual hosts.
  + The link to a data item may not be from a public semantic item; it may be encrypted in a private semantic item.
* A data item should be a link to the most recent revision item. There would thus be two degrees of abstraction between the item a user requests and the actual data underlying it. This will allow multiple semantic items to link anonymously to the same data item. When the data item is updated, all semantic items that point to it will see the changes.
* Data representing a person's medical records could be shared anonymously with the public.
  + When a doctor adds data to a patient's online medical profile, his client would create an anonymous data item and 2 semantic items.
  + One semantic item would be private and its link to the data item would be encrypted with both the doctor's and the patient's public key.
  + The other semantic item would be public and would have a public link to the data item. The public semantic item would have associations with items to categorize the type of medical data it represents, but would have no connection to the user whose data it represents.
  + When a doctor's client creates the public semantic item and the data item, it should use a one-use anonymous user which borrows the doctor's reputation. The doctor's medical reputation and status as a practicing doctor will be needed to make the change because it would be important that the database of anonymous medical data is authentic. It is important that the data item and public semantic item be created anonymously because otherwise it would be much easier to narrow down the possible owners of a particular piece of anonymous data, making it much less anonymous.
  + When adding a piece of medical data to a user's profile, a doctor's client would create duplicate associations for connections between data that should be public. A public shadow copy of the user's medical profile would be constructed which excludes identifying information.
  + The user's private medical profile would include an encrypted one-way link to the anonymous public profile. When doctors (or anyone else) are given access to the patient's profile, their clients would locate the shadow anonymous profile so that any changes made can be mirrored there.
* I don't think it makes sense to make a clear distinction between data items and semantic items. I need to think in terms of object oriented programming. There will be different levels of data items from primitives like strings and integers, through composite data types like lists and trees, to higher level concepts like "Chris Barnett's medical profile".
* A revision item should represent the state of an item at a point in time, including the associations and data items it was linked to at that time.
  + A data item has no semantic meaning on its own so it makes no sense for it to have versions. An object has versions and those versions will be represented by revision items that link to different data items and associations.
* A single fact in a medical record such as "patient has diabetes" or "blood pressure at time x was y" is a piece of semantic information rather than a data item. A lot of this sort of information will be represented by associations rather than literal data. To share such information anonymously, I think it will be necessary to create a full copy of all the associations and data with the identifying information stripped out. When changes are made to a record, they should simply be mirrored in the anonymous public copy.
  + Actually another alternative might be to make just the anonymous public medical profile and have a one way link to it from the patient's user profile.
  + How would someone be given access to only a subset of the information in a patient's profile?
  + It isn't possible to create public one way links because the link can be looked up in reverse using an index.
  + In order for it to be possible to share a subset of a medical profile, a copy would have to be made. The easiest way would be to go back to go back to the strategy if maintaining both a private and public copy of the profile.
* I don't think primitive data items like strings should have an associated zutha item. Whether primitive data items are represented as independent nodes that can be linked to by multiple nodes is up to implementers.
* Any host in the network can be asked to manage a query request. If that host doesn't have the necessary data it should retrieve all sets of data needed by the query and assemble the results itself.
* It should be possible, with sufficient public support, to reveal the reputation-holding identities behind one-use anonymous users.
  + The information required to identify the owner of a one-use anonymous user should be distributed across many independent hosts.
* I don't think it will be possible to implement reputation inheritance for one-use anonymous users. If the user is created with a specific amount of bundled zuth then the owner can be identified (or at least narrowed down dramatically) by finding a request for an anonymous user with that much zuth. Such a request must exist because the Zuthanet must know to deduct that much from the user's account. It wouldn't work to break up the zuth into uniform size chunks because there would be no way to combine them again to make use of the investment quota granted by the accompanying reputation.
  + I can't remember the specific application that lead me to want to enable one-use anonymous users, but I think the anonymity already provided from one's real world identity should be sufficient to break down the most prominent social and psychological barriers to honest and independent discourse and input.

# Thu 7-Apr-11

* Every item should have investment pools specific to attributes applicable to that type of item.
  + Each attribute should have a positive and negative pool with descriptive names for each.
  + Example: comments could have pools for interesting/uninteresting, intelligent/unintelligent, true/false, insightful/uninsightful, dispassionate/biased, etc.
  + Example: products could have investment pools for useful/not useful, reliable/unreliable, user-friendly/not user-friendly, good value for money/poor value for money, etc.
  + Items will be ranked by default on their overall reputation, which will be the sum of their scores across all attributes. Users should be able to construct their own ranking formula though, which specifies custom weightings for each attribute.
  + There should be a default attribute which is invested in by default by users who simply want to generically support or criticise an item. For example the default attribute - perhaps for all items - could just be good/bad.
  + Investment functionality and data can be managed in attribute subitems rather than being an embedded part of every item. An item would not need to have any attribute subitems until a user first wants to make an investment in it.
  + Some kinds of items can be unavailable for investment by default. For example event items need not provide functionality for investment. If a user does something that someone wants to praise or criticise, they would create an action item and an attribute subitem such as responsible/irresponsible, which they would then invest in. The action item would point to the underlying event item(s) it represents, but would be a more semantic representation of the event, targeted towards human rather than machine understanding.

# Fri 8-Apr-11

* It should be possible to post a message to a place in the Zuthanet that is addressed to users in particular region. Clients would query the network periodically for messages addressed to the region they are in. They could do this anonymously.
  + Depending on the user's settings the client would notify them when a message is addressed to their region which has sufficient credibility. The credibility of a message would be determined by the message reputation and message owner's reputation and potentially any other criteria.
  + The default credibility formula would be determined by the user community so that important messages get to most people in an area.
  + This feature would manifest itself as a sort of combination of the minimap ping and regional chat features of multiplayer games.

# Sat 9-Apr-11

* I don't think it will be possible to enable one-use anonymous investment with the backing of a high level user, but I think it will be possible to enable some anonymous actions by high level users.
* A user can get a blindly signed certificate from several hosts that verifies that his reputation in a particular domain is above a certain level. This can then be user to perform an action that is restricted to users with a minimum reputation in a domain and which permits these anonymous certificates.
* For example, a doctor's client can obtain certificates verifying his credentials and reputation as a doctor. It could then use these to update the anonymous copy of a patient's medical profile.
* Some kinds of actions would require accountability and would not allow verification of reputation by anonymous certificates.

# Mon 11-Apr-11

* Nodes with a relatively minor position in the network can keep a routing table of nodes close to them, but default all traffic addressed to unknown destinations to a specific neighbour. That neighbour will either directly or indirectly be a major node with a more complete routing table.

# Wed 13-Apr-11

* Clients register interest in items. The controller group for that item will then send the client updates about changes to that item.
  + Complex items may represent many subitems. Registering interest in a complex item would imply interest in the directly relevant subitems.
  + Items that represent saved searchers would have a controller group which would keep track of changes to the search results.
* When changes are made to the ontology/ schema new changes to data should be required to satisfy their new schema. Old data that defies the schema should be flagged so that contributors can go through the list of broken entities and modify them to satisfy the new standard.
* There should be a period before an update of the Zuthanet standard in which hosts are expected to prepare to make the changes live. During this time, it should be possible to query any host with a flag indicating that the host should use the about-to-go-live standard to determine its response. This way hosts can be checked for readiness to upgrade. If they fail the test, they can be reminded of the upcoming update.
* The Zuthanet standard should not define the protocol for communication between neighbours. Peer to peer wireless communication protocols and such will likely be developed by international standards bodies as the technology develops.
* A Zutha node should pick an appropriate set of neighbours from among the nodes it can connect to using the communication channels it has access to. For example it could pick some nodes that are close to it on the internet (in terms odd latency), as well as the closest nodes it can find (in terms of degrees of separation) that it can access via the local ad-hoc wireless network.
* The Zuthanet standard should be general enough that nodes can send arbitrary messages to each other over it in a restful style. This will allow hosts to provide arbitrary services to any nodes in the network.
* An item should be able to have a variable amount of controllers from one upwards.
* A device could be the sole controller of the complex item that represents it on the Zuthanet. For example a car could be a node in the Zuthanet which hosts the item that represents it. The driver's client would connect to the car as a neighbour. The driver could command the car by manipulating its representation on Zutha. Because the car itself is the controller of that item, the messages would go directly to the car. The car, as a Zutha host, would periodically send the commands it has received to the rest of the network (or at least the hosts interested) - probably in encrypted form. A user with the right permissions could just as easily command the car from anywhere by sending the commands to the car host via the Zuthanet rather than directly.

# Fri 15-Apr-11

* Only an item's controller group should know the worth of an unsubscribed commercial item.
  + This is to make it much harder for third parties to provide an interface to the Zutha knowledge base that undermines one of Zutha's revenue streams - subscriptions from commercial items - by showing the worth of all items regardless of whether they have paid.
  + The controller group of an unsubscribed commercial item should not send updates to the network about investment events in that item and should report its worth as zero (or maybe a special value indicating its unsubscribed status).
  + Investors in an unsubscribed item should be able to see only the value of their investment; the actual worth of the item should be hidden even to them.
  + The owner of an unsubscribed item should be able to see its worth. However, the plain text response to the owner's query should not be digitally signed by the members of the controller group that respond to it. They should encrypt the response in the owner's public key (directly) and sign that. This will stop owners from being able to prove the worth of their item without revealing their private key, which will stop owners of commercial items sending their items' worth data to third party aggregators who could then reveal it it the public, undermining Zutha's potential for revenue from subscribing commercial entities.

# Mon 18-Apr-11

* Project items should have a pool representing how much money and/or zuth is on the table for that project. A user can mark a project item with their intention to invest some amount in it if its later progress satisfies them. They can only put on the table as much money or zuth as they currently have, and only as much zuth as they are allowed to invest based on their reputation. They don't actually have to spend the money they put on the table immediately; they can spend all their money on other things after they have declared their intention to invest in the item if they want. In other words, putting money on the table is not a binding commitment but merely an honor-based commitment intended to incentivise work on the target project. The requirement that the funds be available when the commitment is made is just to ensure it is credible and prevent unrealistic promises.
  + Example application: a wealthy person wants a software tool made, so he describes what he wants in a project item on Zutha and declares an intention to invest up to, say, $15000 in a sufficiently good implementation. Other people who want the same tool made may add their own pledges to the pool or even invest immediately in the item, expecting that it will become popular once created. Software developers looking for work can search for projects using criteria including the real and promised funds backing it. Developers can post their private full solutions to the project, suggest ideas for it, and start or contribute to an open source effort to create the product. Investors can invest in specific contributions using money or zuth, invest in ownership decision items to allocate credit to those they think deserve it most, or invest in the top level project item, in which case their investment will be divided up among the owners according to the ownership decision item. Those who committed to investing a particular amount early on will be expected (through reputational incentives) to have invested about that much by the time the project has satisfied its original goals. In fact, the commitment of a sum of money to a project should be accompanied by a set of criteria which the project needs to satisfy before the funds will be invested. They should be punished by means of reputation if they don't follow through (after being reminded) once their criteria have been satisfied.
  + Project items that have a lot of money on the table for future milestones should attract immediate investment by those who, expecting the project in the foreseeable future to satisfy the criteria required to receive more funding, want to profit from the appreciation effect that later investment will have on their investment.

# Tue 19-Apr-11

* The fantasy mmorpg which involves spirits moving into a previously uninhabited world should be based on the geography of Africa. The virtual world can be a fun test case for some of the ideas I want to implement in Africa. The names of provinces in the game could become the inspiration for the renaming of the corresponding real regions in Africa after those regions become Zutha communities.

# Fri 22-Apr-11

* Responsibility for allocating host ids should be arranged in a hierarchy.
  + When a new node wants to join the network as a host, its host is should be determined by a combination of the selection algorithm and choices by hosts along the hierarchy about which of their subordinates should determine the next character in the id.
  + Hosts with an id that ends in zero could be responsible for allocating the 32 ids that begin with their id.
  + A request to join the network could be served by a controller group consisting of hosts whose ids end in zero. Each of them would select an id for the new host and, as with the allocation of an id to an item, the response with the middle timestamp is used to determine the id of the new host.
* Users should have attributes that can be invested in as a way of praising or criticizing that user for a particular kind of behavior.
  + Example: you could praise a user for contributing semantic data to Zutha by investing in an attribute item labeled something like "ontology contributor".
  + Example: you could praise someone for being good at finding and praising praiseworthy contributions by other users.
  + If you praise a user for a specific action, it should contribute to their reputation in the more general field that action is associated with. It should be possible to specify what it is about a user action you want to praise by investing in an attribute subitem of the action item. That subitem should also be made a subitem of the user's attribute subitem which represents the general form of the same attribute.
* Zutha hosts could independently keep logs of the requests they serve along with data about the clients that made them (if available) such as protocol used to connect, user-agent, ip address (if applicable) etc. Hosts could publish their logs if they chose. It would then be possible for anyone to aggregate the log data of many hosts for various statistical purposes.
* There could be different timeframe ids for different time scales.
  + Each time scale would also have an independent list of active hosts.
  + A host would have to prove itself more reliable (in uptime, computing resources and reputation) to be included in a longer timeframe host list.
  + An item would have some controllers from each time scale.
  + There could be time scales of a week, day, hour and minute.
  + The week long timeframe should switch to a new timeframe at a different time than the other timeframes, say at 12:30:30 pm where the day timeframe changes at 12:30:30 am; the hour timeframe changes at #:00:30 and the minute timeframe changes at #:#:00. This is so that at any timeframe change there are always hosts in a controller group from the other time scales that can pass data about their item to the new hosts that are introduced by the timeframe change.
  + A client that has just connected to the network would first ask for the current host lists for each time scale. From these it would calculate there current timeframe ids. When it wanted to view or make changes to an item it would calculate the controller group for that item by using the selection algorithm once for each time scale, with the item id and the host list and timeframe id for that time scale as input.
* Search queries should be parallelized
  + Any search query can be broken up into a number of index lookups followed by set operations on the results.
  + The API for querying the network should not allow the client making the query to determine the hosts that will manage it. The request handlers will be determined by the selection algorithm.
  + The request handler of a search query will be responsible for parsing the query, doing the necessary index lookups (locally or remotely as necessary) and calculating the final result using set operations.
  + Keyword indexes could be broken up into parts by having each index only store words starting with a particular letter.
  + A host would keep a cache of each index it gets from a remote source and keep track of how old its cached version is. If a query asks for results that are more up to date than a host's cached index then the host will update that index by requesting a fresh copy from the controller group responsible for it.
  + Higher reputation users should have the privilege of being allowed to ask for more up-to-date results.
* Keyword indexes should be stored in the topic map database like any other data.
  + Item ids should be just another name. All item names should be associated with some controller hosts for each time scale as determined by the selection algorithm using that name as input. This architecture will eliminate the need for an index on exact item names (but not a keyword index on item names). Actually this architecture implicitly implements the index in a distributed way. To find all items containing some word X, you just use the selection algorithm on X (the item with the name X) to find some controller hosts for the item that represents the word X. This item will have associations linking it to every item that contains the word X and every host in the item's controller group will know all of those associations.
  + If the hosts calculated by the selection algorithm to be the controllers of the item named X respond that they do not know of an item named X than there is no such item.
  + A message need only be addressed to a subset of a controller group to reach the entire controller group because every member of the controller group knows every other and will make sure the entire group receives every message addressed to the group.
* A data item could be broken into parts that each consist of perfectly random bits on their own. Meaningful data would be constructed from the parts by doing some sort of operation, like XOR between corresponding bits of the different parts.
  + Some parts could contribute redundancy to the set so that only a subset of the parts would be needed to reconstruct the original file.
  + A user could break up a file into such parts and upload each to the Zuthanet without associating any of the parts with any semantic information.
  + The uploader could then provide access to the uploaded file to another agent by giving them the ids of the data items that need to be retrieved along with the operation that needs to be performed between them to produce the file.
  + The main purpose of doing this would be to protect hosts from prosecution for holding data a powerful entity does not want to be available (like copyrighted material) by preventing them from being able to effectively self-sensor.
  + An agent that wants file X removed from the Zuthanet would have to locate the semantic representation of X and follow that to a set of instructions indicating which data items need to be retrieved to construct X. For each fragment of the file they would need to identify every host in that data item's controller group and pressure that host to remove the data item.
  + It would not be possible to get hosts to self-sensor using a broad law because there would be no way for them to determine if a given data item they are hosting can be used to construct copyrighted material.
  + Actually a government could post a public list of data items that are blacklisted and that hosts can be prosecuted for holding. That would allow individual hosts to be raided in search of data items that are illegal to hold. The biggest problem with this is not the loss of individual copies of a data item, but the fear it could instill in other hosts, leading them to self-sensor.
  + Users looking to download a sensitive file could be directed to ask a specific host or set of hosts for it. These hosts might have volunteered to be conduits for the sensitive data because they believe in helping to make it available and they feel sufficiently secure in their anonymity of identity and location. They could delegate storage of the file to any other hosts by breaking the file into parts and giving different parts to different hosts as described above. They would never tell anyone else the identities of these parts though. Instead, they would always be the middle man in transferring data from the hosts that actually store the data to the consumer of the data. All the middle man would need to actually store would be a few bytes of data specifying the id each fragment and the method used to combine the fragments.
  + It would be much better if there were some way for the middle man to connect the data hosts to the data consumer directly without letting anyone else learn of the connection between a fragment and the file it helps create. This is because bandwidth is more likely to be a limited resource than storage capacity so it isn't likely to help all that much for a middle man to delegate storage of a file to larger hosts if it still has to cope with all the data throughout required to relay the data to the consumer.
* An uploader could encrypt the instructions for how to get and assemble the parts of his file with the public keys of a number of hosts he trusts. He would publicly post these instructions in encrypted form and associate them with the semantic representation of the item.
  + A user who wants to download the item would send the encrypted instructions to any of the hosts who can decrypt them.
  + This host would then read the instructions, fetch the data pieces and assemble them and return the result to the user. It would then erase any traces that it had done any of that.
  + If this host was raided, there would be no trace that it stored or knew the whereabouts of any blacklisted data, or that it had ever accepted the request to process any encrypted instructions for the retrieval of blacklisted items.
  + The hosts that actually hold the pieces of a file would never know which file those pieces are associated with unless one of the hosts with access to the instructions leaked them.
  + If a host leaked instructions, the initial uploader could easily upload the file again in different pieces and change the instructions accordingly, removing the leaker - if known - from the set of hosts he trusts.
  + A host who processes instructions would not need to know what file they are returning. In fact there could be multiple layers of indirection such that the top level of instructions asks for an item which in turn contains another set of encrypted instructions. When a host processes instructions, it would not know which item contains the publicly accessible encrypted version of those instructions unless it looked this information up in some index mapping the hash of a set of encrypted instructions to the items that contain those instructions.
  + If a government wanted to make it illegal to process the instructions that would yield a blacklisted file it could provide a blacklist of instructions. The difference with the instruction-based system is that it would be much harder to prove that a host had disregarded the blacklist.
  + When a host returns the output of a set of instructions, it should not digitally sign the response. The receiver of the response will be able to check that it is correct by comparing it with the hash of the expected response stored with the encrypted instructions, but the receiver will not be able to prove that any particular host sent it that response.
  + If a host came under suspicion of being willing to process blacklisted instructions and felt vulnerable to some powerful entity (like a government) finding and attacking them for doing do, it could stop doing so and there would be no evidence that it ever had.
* A government could make it illegal for hosts to store encrypted instructions that can be used to obtain copyrighted or other blacklisted material in the same way they make it illegal to host torrents.
  + Users could post encrypted instructions in comments which could be dispatched by Zutha clients to a host to process. When distributed in this form I think it would become far more clear to people that a government would be obstructing freedom of speech by making it illegal to distribute such information. If it got to this point I think public sentiment would be practically guaranteed to turn sufficiently against such a law that most hosts would default to hosting such content provided they feel reasonably secure from being directly targeted. The location and identity anonymity provided by the Zuthanet already secures most hosts from prosecution, but I think many hosts would default to bowing to the law if they agree with it sufficiently. If the law started making it a punishable offense to host certain comments I think many hosts would stop agreeing with the law.
  + The main focus if this discussion is hosts in democratic countries. Hosts located in dictatorships are likely to have to be secret just to host any Zutha content at all because oppressive regimes are likely to want to keep out the Zuthanet entirely. I guess if governments in free countries started moving away from freedom by obstructing freedom if speech, hosts that want to host material the government blacklists will have to go underground as well.
  + The primary way that freedom of speech will have to be protected, I think, is by making it very difficult to find a particular Zutha host in the real world and discover entities in the real world that are Zutha hosts. Even if the latter were possible, it would still be hard to prove that a particular host had done anything wrong. A government would need to compile a database of all the Zutha hosts it deemed to have broken the law. It would then have to raid many physical entities it suspects of being Zutha hosts and extort from them their Zutha ids before knowing whether that host has actually done anything wrong. I don't think this would be realistic in a democratic country.
* How to tell an owner of an unsubscribed commercial item the worth of the item without allowing them to prove it?
  + The hosts from the item's controller group that respond to the request must not sign the part of the message that indicates the item worth. They would still encrypt the message in the owner's public key so that carriers of the message do not know its contents. Message carriers would not be able to change the message convincingly because they would not have or even know about the signed part of the message indicating that this is a response to a request for an item worth. The item owner would not be able to convince anyone else of the worth of their item because they would be able to make a message with exactly the same characteristics as the one they actually received, but with any worth value they chose.

# Sat 23-Apr-11

* Zutha could be an intermediary in affiliate marketing
  + A website owner would host a Zutha widget in their page which would show the top results of a Zutha query - either manually selected by the site owner or automatically selected based on the content if the page. The results of the query would have to be products that are set up to make sales through Zutha.
  + If users reached a Zutha query page from an affiliate site, a record would be kept of the site they had come from.
  + If a user purchases a product through Zutha the vendor would pay a commission to Zutha for the sale. This commission would be split between the purchaser (in the form of an investment in the item), the site that referred the user to the Zutha query page, and Zutha.
  + The refund received by a purchaser should be the same whether they buy directly through Zutha, or arrive through an affiliate site. This means vendors would pay a higher commission for sales made through an affiliate site.
  + Vendors would have an incentive to get buyers to buy through Zutha so that the sale can add to the status of the product on Zutha.
  + Vendors would have an incentive to pay a slightly higher commission for sales made through affiliate sites to encourage more site to become their affiliates and help advertise their product.
  + Affiliate sites would have an incentive to select categories of products to advertise that are most likely to sell to their readers. They would also have an incentive to show products in a category as ranked by Zutha rather than choose specific products to advertise because they would benefit most by pleasing users so that they are more likely to make a purchase and Zutha is likely to be better than them at picking the products users will like most.
  + Vendors would have an incentive to pay their subscription to Zutha so that their product appears high in placements on affiliate sites.
  + It might be in Zutha's interests to not take any commission for itself for sales made through this affiliate system.
  + Charging zero for the service will be far more effective at attracting attention than charging a very low, non-zero fee.
  + Zutha benefits by attracting traffic to the Zuthanet which will increase its mindshare and userbase. This will in turn increase the value of commercial subscriptions as well as the number of users who will decide to start investing on Zutha.
  + Zutha benefits from the additional funds that will be held by Zutha because it earns money from the interest made by money handlers on those funds.

# Sun 24-Apr-11

* Most view requests should be anonymous, even from high reputation users
  + If a user wants high priority for a particular query though, they should send proof of their identity with their query and hosts that receive it will have an incentive to give it high priority because they will expect to be rewarded with investment in their reputation by the sender of the query.
  + Most queries should be anonymous though because that way hosts will not be able to discriminate against users who have little or no reputation and who cannot reward them for a quick response.
  + A user's salary will include some zuth which is only available for rewarding hosts for quick responses. This will remove any disincentive for users to report host responses to the network.
  + A user reports a host response to the network by sending to the host's controller group, the message they sent and the hash of that message signed by the host. The header of the message would contain the claimed time the message was sent. The hash of the message would be combined with the current timeframe key and passed through the selection algorithm to identify the request handlers. If the host in question is one of the request handlers for the message and its response is correct, its controller group will automatically deduct a proportion of the user's host-specific zuth and invest it in the host, boosting its reputation as a host.
  + Actually a user with reputation should just send proof of his identity to the request handlers of his message upon receipt of the replies. Those hosts would then manage the process of getting their due investment from the user.
  + Hosts should sign in a single message the hash of the query and the hash of the response they give to it.
  + Some hosts will only be asked to return the hash of their response to the client for verification of the full responses of other hosts.
  + When a user with reputation receives a response from a host it should send the signed hashes of the query and response to the other hosts that handled the query along with proof of their identity. The other hosts will be able to confirm a correct response because it will match the consensus. They can then send commands to the other relevant parties to have the user invest in the hosts that responded correctly and on time.
  + The group of request handlers of a view message should become the controller group for the messages from the user that confirm correct responses from each host. This controller group should timestamp each of these messages and use the difference between this timestamp and the sent time of the original query to determine if the response is timely enough to be valid.
  + A user can discriminate against a host by not sending the confirmation that its response was received, but it will be a very uncommon occurrence for a user to be assigned any particular host as a request handler, so a user discriminating against a host will not do it much harm.
  + A user cannot target a specific host for praise since he cannot select which hosts will be the request handlers for a message. When a host he wants to praise does come along, he has no choice about how much zuth will be invested in it because this id determined automatically.
  + Users have an incentive to allow their clients to behave in the default way and send confirmations of the receipt of responses because it will improve the quality of the network infrastructure and won't cost them much.
  + Actually sending confirmations would have privacy costs because it would make view requests public.
* It may not be necessary for rewards for hosts to only come from users with reputation.
  + Any user could be able to send anonymous confirmations of the receipt of responses.
  + It would be possible for a user to send out an arbitrary number of requests but they would not be able to target them at specific hosts so this would not allow them to distort the host reward system toward a favorite.
  + Actually a user could make huge numbers of requests with different parameters and only reward the ones that have a particular target host as a request handler. This would require a lot of computing power to have a significant impact on a host's reputation - though maybe not of someone found an efficient way to manipulate the selection algorithm.
  + If rewards to hosts had to come from accountable users it would be impossible to hide favoritism toward a particular host - and in fact impossible to have an arbitrary impact because any given user only has a limited impact.
  + I think rewarding hosts will have to be a power limited to users with reputation. So the problem that remains is how to encourage enough users to give up their privacy for the benefit of the network.
  + I think the key will be that only a small number of users need to be good citizens to provide an incentive for hosts to serve all users as best they can. Since they will not know in advance whether they will receive a reward for any given response, the optional approach will be to serve all requests as fast as possible.
  + One problem might be that hosts will discriminate against requests for information that is usually sought in private because they will detect that such requests are less likely to be rewarded.
* As hosts acquire more reputation they will receive more requests to handle and more money.
  + There must be a way for hosts to be punished for not being able to keep up with all their traffic.
  + Or does there? All that matters is relative reputation and hosts that can handle more requests will gain reputation faster so they will be asked to handle more traffic.
  + Hosts should lose reputation steadily so that they must continue to perform to stay competitive.
  + Somehow, a host's relative reputation needs to increase if it is easily handling its traffic.
* Correct handling of requests that modify data will be incentivised by punishment for failure rather than reward for correctness.

# Mon 25-Apr-11

* Users accumulate tokens for rewarding hosts at a rate proportional to their reputation.
  + When a user sends a confirmation of receipt of a response from a host to the controller group for that view request, the controller group collectively chooses a random number which is used to decide whether a token will be given to that host on this occasion.
  + Because tokens are a discrete quantity, they can potentially be used as anonymous ecash to provide anonymity to users who still want to be able to reward hosts that serve them.
* Discrete priority divisions for hosts.
  + Probability of picking a host in a particular division rises and falls.
  + At its peak probability will match that of the division above at its trough.
  + At this point the highest performing hosts in the lower division (over the period from low to high demand) move into the higher division and the worst performing hosts in the higher division move into the lower division.
* Hosts should be paid per token earned.
* It might be feasible to have hosts be punished (probabilistically) if a user does not report a response from them but does report a response from the rest of the controller group for the request.
* Punishment would occur in the same way - and in fact in three danger operation - as rewards.
  + The controller group generates a random number which is used to select zero or more of the hosts in the controller group.
  + If a host is selected it will be rewarded if it is one of the hosts which the user confirms as having responded correctly, and punished if it is not.
  + In both cases, the same amount of currency is deducted from the user's account and that much is applied either positively or negatively to the host's account.
* Negative discrimination would be possible, but as with positive discrimination, it should take a lot of computing resources to come up with view requests that target specific hosts. Also, if users must make associate their identity with their confirmations (or lack thereof) of host responses, then it will be very obvious.
* For this reason it probably does not make sense to allow anonymous giving of confirmation tokens. Then there would be no way to detect discrimination.
* Host priority divisions should double as time scale divisions. Hosts in higher divisions should be selected for controller groups by longer time scale timeframe keys and remain in controller groups for longer.
  + Actually the length of time a host should be allowed to remain in a single controller group should depend on its reputation (in trustworthiness), which should be determined purely by user investment.
* Anonymous user confirmations of host responses won't work because accountability is required, so there is no particular need for a discrete currency anymore.
* I like the idea that rewards are determined randomly because this will significantly reduce users' powers of discrimination.
  + For a host to receive an award, a minimum number of hosts must be in the group of hosts that the user confirms replied to his request. This group then uses a random number, selected collectively, to determine which, if any, of the group will be rewarded on this occasion.
  + If a user only sends confirmation of a response from one host, this host cannot be rewarded. This might happen legitimately on rare occasions, but the rule to exclude hosts from reward on such occasions will do no harm. It is far more important to hinder targeted rewarding of undeserving hosts.
  + A user could only send view requests when a target host will be one of its request handlers, but they will need to send confirmation of multiple responses to have any chance that their target host will be rewarded. There will then be more chance that they will reward another of the hosts than their target. If there are five hosts in a controller group for a view request, a user who exclusively makes requests that include a particular target host in the controller group will on average provide one fifth of their reward currency to their target.
  + Since many different view requests must be tried to find ones that will target a specific host, that host will not be able to know in advance what the request will be and do will not be able to avoid having to do the full amount of work to satisfy the request. Therefore there is really no benefit in deliberately targeting that host for requests. It will be able to get plenty of work to do from the network to keep it busy.
  + Ah, but if a high reputation user manages to target a particular host, it can notify that host that certain requests are from a user with reputation and might be rewarded, allowing the host to prioritize those.
  + It will be very difficult to make a significant difference to your host's income with a fifth (or some other fraction) of the rewarding power of the users you have access to. Also, your host must serve all requests well to acquire more traffic and move up the priority divisions because there will be no way to know which requests that you don't respond to will be from users with reputation who can punish you for not responding.
* How to eliminate the incentive for hosts to discriminate against serving view requests that are usually made anonymously?
  + Even if there is a chance that some users with reputation will send confirmations of responses for a request, there is an incentive to serve all requests to avoid punishment. Actually a perfectly rational agent would still recognize that prioritizing traffic with a higher saturation of expected response confirmations would be profit maximizing.
  + A system could be set up to randomly audit the controller groups for all items. If a view request could somehow be made anonymously but with accountability by the requester, larger punishments could be given for failures to respond.

# Mon 2- May-11

* A new genre: Scifomentary
  + This is a play on the terms Sci-Fi, Foment and Documentary
  + I want to create a book and a movie – perhaps at the same time – which will be a documentary about the birth of Zutha and the goings on in the Zutha community so far as well as a projection of what the Zuthanet could become. These two themes should be blurred such that someone watching the film or reading the book in the future will have a hard time working out which parts were reality at the time of writing and which parts were projections.
  + A primary goal of this book/film will be to “foment” interest in Zutha, expand people’s minds to new possibilities and induce greater creativity among the Zutha community.

# Tue 3-May-2011

* There needs to be a way for small hosts to limit the resources they are expected to use. For example a host may only have limited space and may not be able to take on hosting of any more items.
* Hosts can use 3rd party hosting services to host illegal content without fear of being shut down because the content will be encrypted and there will be no way of finding out which physical server is hosting any given item.
* I need to start the site off with very limited features and a simple, coherent theme which people will immediately understand and relate to.
  + I can ads features ad fast as I want to the API.
  + The default site should only a limited set of features enabled. Users cab then enable advanced features as they want them.
  + Third party sites can implement any features they want using all of the extensive features provides by the API.
* I need something like facebook's poke feature, where the purpose is indeterminate and therefore open to interpretation. The feature cannot be too much like poke however or it will be scorned as a copycat feature.

# Wed 4-May-11

* The Zuthanet should provide a service that lets people store their personal data in encrypted form in the cloud.
  + The Metadata they use to organize their files, though, should be public.
  + This will mean that their encrypted files will appear in public searches for information on the same subject.
  + If you found an encrypted document that appeared likely to contain information you want, you could contact the owner and try to get them to give you access to the document - perhaps by offering them money.
  + The document owner could give other users access to the current and future versions of the document through zutha - this would be a publically visible setting. If the transaction needed to be kept secret though, the owner could just send the raw document to another user via any means, including by just sending to the target user the symmetric key which unencrypts the document on the Zuthanet.
  + A lot of intellectual property could find its way into the public domain via this channel.
  + Users searching for a particular book on zutha might find an encrypted document claiming to be that book. They could offer to buy access to the ebook from the owner. They might then choose to put it in the public domain.
  + A community of users striving to make the world's knowledge freely available could create incentives for people all over the world to help them find and create electronic versions of creative works (like books) by offering to pay various amounts for works that aren't in the public domain yet. They could also buy works that are already hosted on zutha on encrypted form, which would have the added benefit of making credible their claims to buy works that have not been uploaded yet.
  + A competitive spirit could arise among members of this community to be the first one to get hold of and make public the most highly demanded and the most obscure and hard to find works.
* Users should be able to join teams which compete with each other in various competitions, including global competitions to collectively earn the most zuth and the most reputation.
  + I don't think there would need to be any special team-specific features for this concept to be effective. I think just the concept of competing as part of a team would energize people.
  + Ideally people should be able to join multiple teams, but this might make scoring complex. Also, what should happen to a team's score when a new user joins who already has a high reputation or a lot of zuth?
* I think there should be a high level group type ("guild" perhaps?) of which any given user can only join at most one.
  + Guilds would compete based on the combined reputation or wealth (in zuth) of their users.
  + The best performing guilds would be awarded prizes periodically by Zutha.
  + When a user changes guild, he would transfer his stats to the new guild.
  + Guilds would compete to recruit the most productive users into their ranks, perhaps by offering them a larger proportion of prizes earned by the guild or even by paying them outright in real money or zuth investment. The latter would need to be more undercover because every investment in an item would be expected to be warranted on its own merit.
  + The main purpose of a guild would be to provide a support network for its members aimed at teaching them how to earn more money and reputation on zutha.
* A team would be a possibly temporary group set up for the purpose of achieving a particular goal.
  + A team would own project items. Donation/investment in these project items would be the team's source of revenue.
  + For example a team might be set up to create an open source software package, tidy up the semantic structure of a particular domain of the Zuthanet, or build an run a school in Africa. Each of these projects would be represented by a project item on the Zuthanet which users could invest in to support the project.
  + Team members alone would be responsible for allocating their earnings among members, specifying permissions and recruiting or expelling members.
  + A team would be treated as a single entity by the rest of the zutha community, receiving praise and punishment as a single entity.
  + Since teams would only have an isolated group of users available for settling disputes and allocating influence, it would be possible for a single user or small number of users to acquire complete control over the team, gaining exclusive control of membership and allocation of rewards. The only option that would be available to members who don't like this situation would be to vote with their feet by leaving the team.
* There should be a distinction between the concept of a user (authenticated actor) and that of an identity (such as a person or company).
  + A user would be tied to a particular means of identification. For example a facebook or google account, or a public key.
  + A user would be linked to at most (and usually) one identity.
  + The identity would contain all semantic information about the user's identity or the identity of an entity on whose behalf the user acts.
  + Multiple users could be linked to a single identity. For example, you could associate both your facebook and your google account to your zutha identity.
  + If two identities needed to be merged, the appropriate owning users of that identity would need to be specified as part of the merge operation. The users associated with one of the identities would thus be disconnected from any identity.
  + All actions would be associated with the user that authenticated them, but when displaying the performer of an action or owner of an item to human viewers, the identity associated with the user would be displayed by default.
  + Reputation should be stored with identities.
  + When a user is attached to a particular identity, it inherits the reputation and other privileges of that identity.
  + For the purposes of calculating reputation, identities should bee considered to own the users associated with them.
  + A user owns items that are associated with actions performed by them on zutha such as comments, investment items and other event items representing their actions.
  + Items that are semantically associated specifically with an identity would be owned directly by the identity. For example a book would be owned by the identity representing its author.
  + When a user is associated with an identity, all its assets are associated with the identity (reputation, actions, comments, zuth etc.) and it gains control over that identity.
  + When two identities are merged, all assets they own directly are placed under ownership of a single unified identity. If any users associated with any of the merged identities are deemed authentic, they are associated with the new merged identity, bringing all their assets with them.
  + Any users left over are orphaned from any association with an identity, though they should retain their connection with each other as belonging to the same entity.
  + Actually all users should be associated with an identity at all times.
  + Associating a user with an existing identity means merging two identities.
  + When users are left behind after a merger of identities, a new blank identity is created for them. Users associated with a single identity should remain associated with the same identity after they are "orphaned".
  + Users can be un-associated with an identity, this being separated from the other users it had shared an identity with. This might happen if it was discovered that someone had hacked into another user's account and added an authentication method of their own (another user) to that person's identity.
  + Companies (and other non-human identities) can now be treated as actors that can perform actions in zutha according to their reputation, make public statements and earn zuth and money as an atomic entity.
* Zutha hosts should run on code that is stored in the Zuthanet itself.
  + The standards that define hosts' behaviour should be defined formally in zutha items, with potentially multiple subitems defining different implementations of that standard.
  + The standard would define some behaviours that are available for open extension. For example there could be many different services that could act as authenticators of identity, all with different APIs and ways of operating. It should be possible for some programmers to get together and build a plugin for the Zuthanet that enables a new method of authentication and have this code be used by all hosts when they are asked to handle requests using that form of authentication.
  + This architecture would open up all sorts of security issues.
  + Of primary importance would be a way to reliably restrict the scope of functionally of code to different degrees in different contexts.
  + Code that defines the core behaviour of hosts would need access to broad capabilities on the host machine. Such code would need extensive vetting before being approved as an acceptable implementation of the standard to which it applies.
  + Code such as a plugin to allow authentication with a new service would not be implementing a standard but providing an extension allowed for by the standard. Such code should only be given the minimum powers and access to hosts' resources required to perform their specific function (authentication provider extension for example). With these restrictions on place it will be possible to allow anyone to implement a plugin for the Zuthanet and have it be available immediately without extensive peer review of the code. Peer review will happen eventually of course, and bad plugins will be noticed and disabled, but there would be no need to massively slow down the process of publishing a plugin with extensive testing if it only has minor powers anyway.
  + An authentication provider plugin, as it happens, is not a piece of code with minor powers and would need to be tested extensively before being allowed to go live, but I'm sure there would be many applications of minor plugins (games, apps?) that would benefit from low barriers to entry. I would want the zutha app market to be more like the android market that the apple app store after all.
* Hosts should nominate which plugins/ apps they are willing to support by adding themselves to a list of compatible hosts under each of those apps.
  + When choosing a host to handle a request that uses a plugin, the selection algorithm will select from the list of hosts that support that plugin.
* The entire host program (or an arbitrary implementation of the standard) should be installable on any machine simply by asking to install it using a simple Zuthanet client.
  + The client (perhaps javascript running in Firefox on Linux) would identify an appropriate program to download from the Zuthanet according to the machine it was running on. It would ask to run this little program on the user's computer. This program would then download, compile and then run the entire host suite from the Zuthanet, optionally allowing the user to specify which implementations of various parts of the standard he prefers.
  + This downloader program seems a lot like a Trojan. That's kind of spooky because such a Trojan could allow zutha to spread itself autonomously across the internet the way the daemon spreads in Daniel Suarez's books.

# Thu 5-May-11

* API should include a service to find items a user is likely to like in a particular category.
  + It already does: filter reputation by investors of items you have invested in and then browse to an arbitrary category or query.
* Query items should be generated automatically for common queries.
  + commonly used queries that include an item X as a result should appear in a quick-access list on the page for X.
  + In order for popular queries to be identifiable, every query will need to be recorded.
  + If every query is going to be recorded, every view request might as well be.
  + Should queries by hosts be recorded?
  + How would a host know if a query is from a user or a host needing more data to fulfil its own query?
  + I think only view requests that are verified by users with reputation should be stored and used to identify common queries.
  + A query counter should also only be incremented if the view request resulted in the host being praised. This would make it much harder to game the system to artificially increase the apparent popularity of a particular query.
* A user's controller group should maintain pre-calculated data which would be used in calculating results filtered on a specific set of criteria defined by the user.
  + For example a user, X, might have defined a filter on item reputation that only uses investments by users that have invested in the same items as X and weights those investments according to how much zuth is invested in common between its owner and X.
  + X's controller group would then calculate a weighting for every user on zutha based on X's specified parameters.
  + When X queries the network, the hosts assigned to the request will ask X's controller group for the weightings of the users who own investments in items in the result set and calculate the reputations of those items based on those weightings.

# Mon 9-May-11

* Agents could send secret messages that will be known to outsiders to be important by using a datapad (one-time-pad) shared previously in an encrypted message that cannot have been known to be important.
  + The strength of this cryptographic scheme is theoretically only as strong as the asymmetric encryption scheme used to encrypt the datapad, but in practice it is stronger because the message that is used to share the datapad can be obscured as a needle in a haystack of insignificant messages.
  + To find the datapad, an attacker would have to decrypt every message sent between the communicating parties until he finds the datapad. Without either party's private key, this would be extremely computationally expensive, even long after the messages were actually transmitted and technology has improved computing power.
  + It might also be very difficult to identify the message that conveyed the datapad as such because, as an arbitrary number without any pattern, it would be impossible to determine upon encountering that number, that it is the result of a successful decryption of the top layer of encryption, let alone the final message.
  + The decryption trail could be further obscured by dropping false datapads that lead to sensible but wrong interpretations of messages. Datapads can be constructed to have any message be interpreted as any arbitrary message, so there is a lot of potential for misdirection using this scheme.
  + The correct datapad to use could be indicated to the message recipient inside the final message that results from the use of that datapad. The recipient would thus be obliged to decode the message with every datapad he receives (both before and after transmission of the message itself) until he produces a message with the telltale indicator that it is the correct one.
  + If every datapad is sent using a different key pair, decryption of every message by the holder of the private key would be easy, but it would be very hard for anyone else.
  + There would be far more messages to decrypt than just the datapads. All messages to the recipient would need to be decrypted.
  + It might actually be more secure to send the indicator of which datapad to use in some other apparently innocuous message.
  + It seems clear that an encryption scheme is possible that requires arbitrarily more work to decode on the part of the attacker than the recipient who holds the relevant private keys. Is this convoluted scheme any better than simply increasing the size of the asymmetric key, though?
  + The main disadvantage of using just asymmetric key encryption is that an attacker can know when he has successfully decrypted a message. The use of datapads, however, would allow an arbitrary degree of misdirection. It would also be possible to share bits and pieces of datapads separated across various times and communication substrates. Potentially, an attacker, even long into the future, could be unable to know whether he has the right message because he would be unable to know whether he has found all the relevant communications required to identify the correct datapad to use.
  + An intended recipient on the other hand could indicate to the sender his receipt of each message with signed responses that would be practically impossible to counterfeit in real time.
  + The main vulnerability of any encryption scheme is still going to be the vast pool of unknowns that exist in the real world, beyond the elegant mathematics of theoretical cryptography. Any system of secrecy will remain vulnerable to such real-world phenomena as blackmail, betrayal, malicious software on the devices of sender or receiver etc.
* Associations can be encrypted.
  + An organisation could host their entire database on the Zuthanet in encrypted form.
  + In order to be able to perform queries on data whose associations are encrypted, the hosts responding to the query would need to be able to construct indexes on the encrypted associations.
  + For some kinds of data, an index on the actual encrypted values might be sufficient to serve a query that has been converted to be in terms of those encrypted values, but there are likely to be queries that will require the host to know more about the meaning of the data. For these cases, the owner of the data could potentially delegate responsibility of the data to certain trusted hosts. These trusted hosts would be entrusted with the relevant encryption keys so that they can build their own internal indexes over the unencrypted associations.
  + Queries that need to include encrypted results would need to be sent to the hosts entrusted with managing that encrypted data. The requesting client would also have to send proof that he has access rights to the encrypted data.
* There will need to be many different host lists for each of many different responsibilities a host may have.
  + Some host lists may be private. For example an organisation may have its own internal list of the hosts it trusts. Its encrypted data would only be sent to these hosts.
  + Such an organization could also make the host list public and assign it to items of an obscure type defined and used only by that organization.
  + Whenever the selection algorithm is used, the host list from which to select hosts should be determined by the type of the item whose controller group needs to be found. This list should be obtainable by asking the controller group of relevant the type item. This lookup should be done on the universal host list, which should be known by all nodes.

# Wed 11-May-11

* The zutha host standard will be a meme.
  + It will compete in a process of evolution, with the most competitive host standards being the ones that multiply themselves most effectively.
  + The Zuthanet itself will be a cybernetic superorganism - or maybe there will be multiple competing superorganisms, each defined by a different host standard or collection of compatible standards in the same way that organisms are comprised of a set of genes.
* I want to develop an extremely general programming language that runs on top of the Zuthanet platform.
  + Hosts could opt in to support programs written in this language, which would take the form of items of a particular type.
  + Let-s call the language Z for the sake of this discussion (even though Z is already taken as the name of a programming language).
  + A program written in Z should be executed on creation. In practice, a Z program would be run by creating a small Z program that just calls the larger one, perhaps with arguments. In this way, the process of creating and calling/querying a Z program would be unified into the same process.
  + Z should merge many programming language paradigms including logic, functional and object-oriented.
  + Z should be capable of calling code written in any arbitrary language. It should also have a general toolkit for making interfaces that allow other languages to call Z code. Both of these goals would be achieved via translation libraries that map Z code to and from other languages. Such translation libraries would need to be made for each language Z is to interact with.
  + A Z program (a single Zutha item of type Z program) would call code in another Z program by importing that program using its full unique name on the zutha knowledgebase or by qualifying a method/function/ predicate call with that full name.
  + Once someone has written a translation library from Z to some language X and put this library in the appropriate place on the Zuthanet, any Z program will henceforth be able to call code written in that language (that is hosted on the Zuthanet) in the exact same way as other Z code is called.
  + The side effects of any code run on the Zuthanet should only be the modification of the Zuthanet itself. The language Z itself should be the only one able to do this. Actually output to standard in, out and error should be mapped to default actions like the creation of items containing that output which would be associated with the action item representing that particular execution of the program.
  + I don't think it would make sense to allow programs written in other languages to call Z code. Z code would be fundamentally intertwined with and dependant on the Zuthanet itself. There would be no standard way for traditional languages to impor Z code. Instead, the interface to all Z programs should be through standardized web services including ones based on SOAP and REST. Actually the Zuthanet itself will lead to standardized interfaces on a similar plane to SOAP, which would be implemented in each language the way libraries for using SOAP are.
  + Any call to a Z program will leave a trace in the form of an item that holds the result of that execution. This will also serve as a cache of previous runs of the program so that whenever a call to a program is precisely repeated, the result can optionally be looked up from previous calls instead of recomputed.
  + The caching concept will only be applicable for calls to non-destructive (immutable) functions.
  + It would not make sense for Z to call code in other languages which depend on or modify local state. Any call to a Z program will be executed redundantly by multiple hosts in parallel. If some parts of the code depend on local state specific to each host then each will get a different answer and the Zuthanet will reject the result.
  + I was just thinking that some way would have to be devised to require that all code run on the Zuthanet be immutable, maybe it wouldn't matter. Any code that depends on local state would fail because different hosts would respond differently when executing it. There is still a problem though: nothing prevents each host from performing different destructive actions while returning the same result. The program could ask the Zuthanet to make a new item via a web service and this create item action would be duplicated by every host that executes the program.
  + There is a bigger problem with expecting hosts to execute arbitrary code: security. Hosts could theoretically run any code safely in an isolated virtual machine, but this would be messy, resource intensive and probably still not entirely safe. Ways would be found to do all sorts of nasty things if the door was left open to run arbitrary programs anonymously in a cloud-based supercomputer.
  + I'm not yet entirely convinced that it would be impossible to design a safe and practical system that allowed users to run arbitrary code in the cloud (after all this is what web hosts do). However I think this needs to be thought about separately. Such a paradigm is not compatible with the Zuthanet model of executing every request redundantly.
  + The paradigm I'm immediately interested in is how to allow arbitrary manipulations of the Zuthanet using a general purpose programming language. For such a paradigm I think Z will need to work alone.
  + A Z program should output an action which acts on the Zuthanet. The action should be encoded in the syntax of the standard API.
  + It would be cool if the API, the query language and the general purpose programming language could all be unified.
* The entire Zuthanet should be exportable as a topic map. Should topic map syntax form a fundamental part of Z though?
  + Ctm has a syntax for macros, but I don't think they generalize to doing arbitrary manipulations of input arguments. That is essentially what I'm looking for, though: ctm extended to be capable of arbitrary manipulations of data. The fact that ctm ultimately results in a set of topics and associations that would be merged into a topic map is very much in line with the paradigm I'm envisioning.
  + It would be convenient and more elegant for Z to natively support zutha-specific features like investment and zutha's model of user authentication. Then again, perhaps it would be more elegant if even zutha's most core functionality was implemented (and interacted with) in such a general way that it would leave the door open for extensions to that core functionality that are implemented within the existing framework.
* The zutha standard should define the programming language Z.
  + There may be many implementations of the Z compiler.
  + The Z language will interact with abstract neighbours. The details of how neighbours communicate will be left to each host to manage. In most cases the means of communicating will be based on existing standards like http over tcp/ip over Ethernet, but hosts can devise their own private means of communicating if they want.
  + At the lowest level, the zutha standard will define, in Z code, how to read incoming messages from neighbours and write outgoing messages.
  + Actually I think the Z code should work at a higher level. A Z program should only have the power to modify items on the Zuthanet. Perhaps a different language is required to specify the behaviour of individual hosts.
  + Z's only means of output should be to write a topic map - that is, a set of topics and associations.
  + Z could still be used to implement the ad-hoc network layer. It would take a message formatted as a topic of dedicated type as input and generate a topic as output.
  + The Z language could be used to formally define the correct behaviour. This specification would be immediately executable using the Z compiler but that wouldn't preclude the implementation of the specification in other languages. The output need not actually be a string that contains a topic. It could instead be formatted more efficiently as a packet with some fixed sized fields for the critical data used in the ad-hoc network layer. output could only be formatted this way for transmission to neighbours who support that more efficient standard.
* The ad-hoc network architecture should be an abstracted layer which simply provides a service to the next layer up to send a message to a particular node in the network.
  + The zutha standard should be able to operate on top of the conventional ip addressed internet or any other network architecture.
* The hosts who are assigned to manage the creation of a Z program item should also execute the program and store the result as an item with the same timestamp as the program item.
  + Hosts should only communicate with each other for the selection of the timestamp and item ids after they have calculated the result.
  + Hosts should fetch all the Z modules required by the program and compile them before running the program.
  + The result of the program should be returned to the caller as part of the response.
  + Hosts need not actually store the result of every program since they can recalculate it again on demand to trade a permanent cost in storage resources for an uncertain cost in processing resources.
* The Zuthanet could host arbitrary services including such things as realtime stock prices, weather data and video streaming, as well as arbitrary computational services.
  + To provide a realtime video feed, the provider would create an item which is specified to have just one member in its controller group: the provider of the service. Actually the feed could be mirrored by many hosts. The content generator would stream only to the mirror hosts who would then act as the controller group of the live video feed item and supply the feed to everyone else.

# Thu 12-May-11

* While Zutha may come to be treated as having a mind and purpose of its own, I believe this purpose will be fundamentally in line with my goal of improving the quality of human life on earth because Zutha’s nature will be such that it will succeed by creating positive value for as many people as possible.
* As long as people are the ones choosing the host standard and creating hosts that is. If other intelligent entities begin to acquire high reputations on the Zuthanet they will also have a say in selecting the successful host standards and will thus be influential in Zutha’s evolution.

# Fri 27-05-11

* Data items should store a fixed number of bits.
  + All data items should form only one part of at least two pieces required to reconstruct a meaningful piece of data.
  + A meaningful piece of data would be constructed by doing some bitwise operation (like XOR) between multiple data items.
  + A given data item may form a part of many different meaningful pieces of data.
  + When uploading a new file to the zuthanet, the file will first be broken into parts of the standard size. Each part will be "encrypted" by using a random selection of existing data items sort of like one-time-pads. The semantic item will link to a sequential list of Data Chunk Constructor items which will each contain instructions for how to construct the meaningful data chunk it represents using 2 or more raw data items.
  + Any given raw data item contains meaningless random bits on its own. It may be used to construct numerous semantic items but it cannot be said that the purpose of any raw data item is for the creation of any particular piece of data. Actually, all but one of the required data items may have been created before a given meaningful data chunk, but at least one data item has to be the one that maps other meaningless (on their own) data chunks to the desired meaningful data chunk.
  + The calculation of the (1 or more) raw data item that needs to be created in order to be able to construct a target meaningful data chunk should be allowed to be done by a client. Actually nothing stops anyone from uploading arbitrary raw data items and manually linking raw data items together to form meaningful data chunks.
  + While uploading an innocent file, someone could construct each meaningful data chunk with multiple new raw data items (as well as possibly other existing raw data items). Some of the redundant data items could be constructed such that they can form other meaningful data chunks with other items. Someone who wants to upload something illegal onto the zuthanet could use this technique to covertly upload it over time, long in advance of actually creating the meaningful data chunk constructors and semantic item that stitches the parts into the meaningful whole. By this time, many of the data items made for the illegal item may also have become essential for other innocent items. It would thus be difficult (I think) to legally justify prosecuting hosts who host them.
  + I think it will always be possible to hold the most recently created data item used by a meaningful data chunk responsible for the creation of that chunk. This would make that data item a valid target for a legal attack against the meaningful data chunk.
  + It would be possible to plant data items in advance such that a single new data item could be used to create either of two meaningful data chunks using only existing data items. I don't think the creator of this item would be able to claim innocence of intending to create both meaningful data chunks, though. It would still be possible to blame the most recent data item for the creation of both chunks.
* If any data chunk may be asked for at any time for the purpose of creating literally any meaningful data chunk, then there would be no way for hosts to discriminate against items that are unlikely to be visited openly by users with reputation.
  + There would be no risk of embarrassment in requesting an item that could conceivably be being used to construct an arbitrary piece of data.
  + There would be no database mapping data items to the meaningful data chunks they can be used to construct - indeed some uses of a data item could only be known to private groups.
  + Thus a host cannot know, when being asked to return a given data item, whether or not the requester is a high reputation host who will praise them for responding. They can then only maximise their score as a host by responding to all requests as efficiently as possible.
* It would be easy to map data items to the meaningful data chunks that use them for public files so links between data chunk constructors and the data items they use might as well be associations.
* The main benefit of promoting ubiquitous use of one-time-pad-like encryption with random data items is that it will discourage hosts from neglecting any data items because any one of them might be demanded at any time by a high reputation user who needs it to construct some private file. Such a user can reward hosts for responding promptly as well as voluntarily punish them if they fail to serve.
* A major benefit of data items serving multiple purposes is that it allows users to maintain some degree of privacy about their browsing activity while still being able to reveal their identity to praise hosts for returning individual data items.
* The main target of legal action to take down an item might be the semantic item that contains the top level information about how to get hold of an item, including links to the data chunk constructors.
  + Semantic items also don't have the advantage of having unspecified meaning the way data items can.
  + Users with reputation cannot praise hosts for sending them a semantic item without revealing that they asked for an item with a very definite meaning.
  + Hosts will be able to discriminate against semantic items that users are unlikely to want to admit to having viewed (such as one representing an illegal child porn video).
  + Ideally there would be some way to allow and encourage hosts to discriminate against such things as child pornography while refraining from doing so for such items as incriminating documents leaked by a whistleblower.
  + Maybe this incentive is already present. Users who reveal that they like child pornography are likely to be severely criticized and will struggle to obtain a high reputation. A user that openly views politically radioactive material, on the other hand, is not likely to incur the wrath of the general zutha community. They might actually receive general praise for helping to promote openness and this would help to counteract any reputational attacks from parties adverse to the document being public.
  + Users who have a real world identity might be discouraged from revealing their identity when they visit politically sensitive content, but provided there are enough anonymous users with high reputation there will still be an incentive for hosts to retain the content.
* It would be very hard to eradicate the semantic items representing files (or any other semantic items) from the zuthanet because, being small and a necessary presence in various indexes, they would be extensively cached.
* They would also be very easy to re-post both on the zuthanet and in various forms all over the internet because a user could simply copy and paste a block of text in topic map form to any number of text hosting mediums. Any Zutha client could then read that topic map content and discover which data items need to be downloaded from the zuthanet to construct the file.

# Mon 30-May-11

* Make a simulation of the prisoner's dilemma which incorporates reputation.
  + Use this simulation to experiment with different rules for determining reputation in order to gain insights for the design of Zutha's reputation system.
  + Agents in the simulation should choose at the start of each round which other agent they want to play prisoner's dilemma with. This would mean every agent would play 1 or more game per round.
  + An agent would not be able to refuse another agent's invitation to play.
  + Agents would also be forced to select a partner each round. They could elect to be assigned a random partner from the whole pool of agents or select randomly themselves from among the agents they know about.
  + Scoring rules
    - If both agents defect they should both score slightly negative (say -1).
    - If one agent defects and the other cooperates the defector should gain a lot (+2) and the sucker should lose a lot (-2).
    - If both agents cooperate they should both gain a little (+1).
  + Strategies should be designed in a modular way such that new strategies can be formed by mixing rules and varying parameters.
  + Agents should participate in an evolutionary process whereby agents can reproduce voluntarily with other agents and form a child agent with a slightly error-prone mix of their rule sets.
  + Agents should die after a perioi of time (rounds) or if their score drops below a certain threshold.
  + Actually I should also try allowing agents to share rules with each other voluntarily and modify their own rule set subject to what they see working in practice. In fact I like this style of strategy evolution better as it is closer to the style of evolution relevant to Zutha - that is: meme evolution as opposed to gene evolution.
  + The simulation should be run with several different degrees of information availability.
    - 1: agents can only remember the previous game they played.
    - 2: agents can only remember the previous game they played with each agent.
    - 3: agents can remember all previous games they have played.
    - 4: agents can ask other agents for their opinion about the agent they are about to play in the form of a single number from -10 to 10.
    - 5: agents can offer to mutually share their full game history with other agents.
    - 6: every agent contributes their opinion about their game partner to a central database after every round. That is, they update their rating for that agent each time they play them. Agents could opt out by always rating their game partner 0. Only an agent's aggregate score is listed in the public database.
    - 7: all game history is available to all agents.
  + My hypothesis is that, as information becomes more available, niceness coupled with informed rejection will become an ever more competitive strategy.
  + I wonder whether altruistic punishment would arise.
  + I think it would be very difficult to program the meta-strategies that would evolve their own game strategies by learning from other agents and their own experience.
    - Instead of trying to do all this programming myself I should set up an ongoing public competition whereby real people can do the learning, invention of new algorithms and sharing of ideas (memes).
    - New entrants should be allowed to join the real time simulation whenever they want.
    - In order to better mirror the scenario that zutha must operate in, contests should be allowed to enter anonymously. This would mean, as on the internet, that a virtually unlimited number of agents in the simulation could be controlled secretly be the same contestant.
    - In the internet style scenario agents should have the option to not accept a challenge (to prisoner's dilemma). This is in keeping with a major difference between cyberspace and reality - that you can't be physically forced into a situation in cyberspace whereas in reality you can (for example, by being held at gunpoint, or simply by ending up in a social situation where vanishing is not an option.)
    - New agents/players with low or zero score may find it difficult to get any older players to play with them because their opponent would know they have nothing to lose. New players would probably have to build up their score by playing with other newbs before they would have the credibility to verse higher ranking players.
* I think it would interesting to contrast the prisoner's dilemma simulation in a traditional biological evolution scenario (where each agent cares for its own survival and evolution occurs through genetic mutation) versus an internet-like scenario (where memes/ideas are the foci of evolution and they evolve and propagate outside the domain of the simulation).
* In the scenario which allows agents to rate their game partners I don't think there would be much incentive to actually do so accurately. If the ratings are only accumulated as a single reputation value for each agent, there would be no way for other agents to know if you gave false ratings.
* There are much bigger problems than this though. With the ability to create an unlimited number of agents, competitors could create many throw away agents for the sole purpose of being mean to everyone except selected agents owned by the same competitor, to which they would be suckers. These throw away agents could also give their allies an unrealistically positive reputation and sabotage the reputations of competitors.
* It seems clear that many of the same challenges I have to overcome in designing the zutha reputation system apply here too if I want to allow unrestricted entry.
* Perhaps I should simulate the full zutha reputation system in the simplified context of a prisoner's dilemma tournament.
* I need to find some may to incorporate into the prisoner's dilemma tournament the main principle that protects zutha from abuse by armies of throw away users - namely that new users have no initial power to affect others.
* How can agents in a prisoner's dilemma tournament start powerless but be able to earn influence over time?
* Perhaps the metaphor of the prisoner's dilemma is not really applicable to the paradigm of the internet.
* The prisoner's dilemma problem is dependent on each agent being able to do good or harm to the other. A good parallel would be trade, where both parties could cheat the other by giving them counterfeit wares. If both cheat both gain nothing. If one cheats he gains a lot and the other loses a lot. If both trade fairly they both gain by the exchange.
* In cyberspace, a virtual agent has nothing of value to offer by itself and certainly 1 million copies of the same virtual entity have nothing more to offer than the first. The intelligent being that controls the 1 or 1 million virtual agents may have something of value to offer but it is not magnified by making the 1 million copies.
* The difficulty in an online tournament of unrestricted entry is that the measure of value could not be abstracted away (by defining it as a number stored in a computer) in a realistic way that does not admit arbitrary creation of value by players. The only way to do so would be to associate the value with the real intelligent beings behind the arbitrary number of virtual agents they control, but this couldn't be done for an open internet competition with anonymous participants.
* If value cannot be abstracted away then the only way to represent value that I can see is through the subjective opinions of real people. If people have to create real value in order to play in the tournament though, then practically the whole zutha framework would be needed to facilitate it.
* What I'm really trying to find a way to simulate is the evolution of cooperation in memes in the context of cyberspace.
* It seems that this problem cannot be substantially simplified beyond the complexity of zutha - zutha itself will have to be the simulation.

# Sun 5-Jun-11

* Controller groups could share secret keys so that they can produce randomness (from the perspective of everyone else) but still be able to all independently come up with the same results.
  + Whenever a new member joins a controller group, every member would send a randomly selected key to every other. Once each has received every other key, it will send the whole collection of them to every other host, along with the master key it derives from the collection.
    - This will ensure that anyone who sends a different key to different hosts will be punished.
    - It will also allow every host to be confident that every other host has the same master key.
    - Another benefit will be that it will allow slightly isolated hosts to receive the whole bundle of messages from a host that is able to reach it, even if some other hosts in the group are unable to get messages to it.
  + Controller groups could use their shared random number to generate randomness in events they are responsible for. For example, they might be responsible for determining the weather in a particular cell of a distributed MMOG. They would need to generate randomness, but all of them would need to generate the same random values so that all their responses match without them having to coordinate on every single user request.
  + It would be useful if clients in a MMOG could pre-download graphics and other data-rich components of a game before they actually need to use them. Sometimes, however, it should be kept secret that a particular entity may enter the client’s view in the near future. For example there may be some kind of rare unit hiding nearby. Players should not be able to learn of this rare unit’s likely presence nearby by watching the data being pushed to their client by the relevant controller group.
    - The controller group responsible for the region could encrypt the graphics data they push to the client using a key generated from their shared random number
    - At the moment when the client needs to know about the presence of the rare unit, the controller group would simply send it the key to the encrypted data it already possesses
    - The controller group could also host the secret data in encrypted form on the Zuthanet so that other hosts would share the load of serving the data to game clients
      * In order to maintain the secrecy of the data, it would be necessary to upload new encrypted versions frequently – that is, before the word could spread of the identity of a given encrypted item.
* Keyword indexes could be divided into shards that are each managed by a different controller group
  + E.g. there might by a controller group responsible for maintaining the index of all words starting with “st”
  + The index should actually be stored as a data structure within the topic map simply by making associations between words and the items that contain them
  + Whenever a new item is created the controller group that creates it will make associations from it to every word that it contains
  + In order to find the ID of a word item (if it isn’t cached), the parent of the word item (such as the “words starting with ‘st’ index item”) will be asked.
  + The IDs of all index items will be expected to be cached by all hosts
  + Hosts can find out the current controller group of a given Index item by using the selection algorithm with the Index item’s ID and the current timeframe key.
  + In order to reduce the number of network requests that must be sent, it should be possible to give commands destined for individual word items to their parent Index items
    - A controller group creating an item X containing the word ‘stock’ should be able to send a message to the ‘st’ Index item (well its controller group actually) indicating that an association should be created between X and the Stock word item.
    - The ‘st’ Index item would forward the request to the Stock word item, which would create the association between itself and X, filling in only its own ID
    - Simultaneously, the ‘st’ Index item would reply to X’s controller group, telling it the ID of the Stock word item. X’s controller group would then create the identical association created by the Stock word item by filling in the same ID in the association (that of the Stock word item).

# Wed 8-Jun-11

* Nodes in the Zuthanet should use public keys as their IDs
  + This would allow nodes to generate new IDs at their leisure while preventing the spoofing of others’ IDs
  + A client can create multiple client IDs as well as a separate node ID which it uses to identify it to its neighbours.
  + For an open message such as a view request, a client would make a hash of the message which it would encrypt with a public key of its choosing. This public key would be included with the message and act as the sender ID. The encrypted hash would act as a signature that proves ownership of that ID. The creating client would then take on the role of a node, wrapping up the main message in a header which includes the hash of the main message, the node ID and that hash encrypted with the node ID.
  + When a node receives a message addressed to a particular ID (some public key), it must look up this ID in its dictionary of known nodes to discover which neighbour to forward the message to. It replaces the message’s existing header with its own (including the hash encrypted with its own public key, which is also included).
  + Host tables should include in each entry the host’s public key, which is used to identify it in the network layer, as well as its item ID prefix, which begin the IDs of all items it creates.
  + It might be beneficial to require that all messages be encrypted with the public key of the next node such that the only information visible to anyone who intercepts a message between Zutha nodes would be the destination address – maybe an IP address or some other identifier for different kinds of networks.
    - The original message could still be open to all the Zutha nodes who carry it to the destination. This would allow intermediate nodes to respond to view requests if they have an up-to-date cache.
    - The sending client would simply sign a hash of the open message as usual
    - The sending node (the same machine as the sending client) would still sign the hash of the message with its own private key, but would then encrypt this whole bundle with a symmetric key and this with the public key of the neighbour it is going to send to. It may possibly include this public key in the message to identify which private key the receiver should use to decrypt the message, but this would be optional (that is, only if that requirement is specified by the receiver).
    - Intermediate nodes would decrypt the symmetric key and re-encrypt it with the public key of the neighbour they are going to send to
    - Intermediate nodes would know the ID of the sending client and that of the neighbouring node that forwarded the message to it, but even the neighbour of sending client would not know that it received the message from the sending client itself.
    - Interceptors of the message would not even know the source address (IP address spoofing could be used to enforce this principle over IP networks), only the destination address – and that’s it.

# Fri 10-Jun-11

* It should be possible to create arbitrary calculated fields
  + Reputation would simply be a predefined calculated field
    - It would be calculated by adding up the values of investments in each of an items Attribute subitems plus the reputation of items it owns.
    - Actually calculating reputation would be a bit more complicated because adjustments would have to be made to disqualify investments by the item’s owner and investors who have exceeded their cap on contributing to this item’s reputation.
  + Calculated fields should be able to take arguments
  + A calculated field could be used to calculate the similarity factor of a user UserY with a specific user, UserX, passed to the calculated field as an argument.
    - It might be defined as follows:
      * Define the *shared\_interest\_factor(UserX, UserY)* of item ItemX as the product of UserX’s investment in ItemX and UserY’s investment in ItemX
      * Define the similarity\_factor(UserX) of UserY as the sum of the shared\_interest\_factor(UserX, UserY) of every item invested in by UserY
    - This calculated field could be used to find users with similar interests to you (or any other user)
      * The query would be something like: “type=user orderby:similarity\_factor(chris\_barnett\_(user))”
  + Items could be ranked according to a personalized scaled reputation as follows:
    - Define the scaled\_reputation(UserX) of an item ItemX as the sum of each user’s investment multiplied by that user’s similarity\_factor(UserX)
    - Orderby:scaled\_reputation($user)
    - This scaled\_reputation value would be very big because it would be in the order of the product of 3 reputation values
    - The calculation can easily be tweaked though and many variations of this calculated field can be created by many different users with the best ones being voted on using the Zutha reputation system.
* How to enable a smooth transition between a single server hosting Zutha and the distributed model
  + The first server should be allowed to continue making items after the distributed system is started
  + Requests can either be sent to multiple 3rd party hosts as specified by the selection algorithm or to the First Host
  + Once items have been created and given an ID, they will be assigned to a controller group according to their ID and the current timeframe key
  + The First Host, for a while, will be part of the controller group of every item
  + At the moment when the first batch of 3rd party host join the network, the controller groups of all existing items must be determined
* There might be a need for two kinds of controller group – one that knows about a given ID and another that is actually responsible for it
  + The first kind of group needs to be the one that is selected by using the selection algorithm with the raw ID
  + Some hosts may not be able to manage some kinds of items, though, so there may need to be a more deliberately selected controller group for actually returning the contents of an item and accepting requests to modify it
  + An alternative would be that any request for an item needs to include the item’s type – either explicitly or with enough contextual information that its type can be inferred.

# Sat 11-Jun-11

* How to enable different controller group selection rules for each item?
  + I think there will need to be two layers of selection
    - The first layer should only require the item ID and the current timeframe key
    - The host group selected in this phase will know the selection rules for finding the main controller group of the item
  + There are two main scenarios that I can think of where specialized selection algorithms would be needed
    - Items that represent interactive machines in the real world should be hosted only by that machine itself
    - Items that come under high demand should be able to have their controller group scaled up in size to handle the load
      * In order for the larger group to actually reduce load on each member, it would also be necessary to have the selection algorithm select a randomized subset of the controller group for each requester
      * This could be done by defining the selection algorithm as follows:
        + Let N be the number of candidate hosts
        + Let n be the size of the controller group
        + Let q be the number of hosts that should handle each view request
        + Let H be the hash of the item ID and the current timeframe key
        + The controller group is found by doing H mod N, n times to select n hosts
        + Let H2 be the hash of the requester’s ID (a public key) and the current timeframe key
        + The set of request handlers is found by doing H2 mod n, q times to select q hosts
      * It would only make sense for view requests to use a subset of the controller group since the entire controller group would always need to know immediately about any changes to their item
      * Load will already be distributed for view requests via caching
      * There might be need for a larger controller group for an item that is evolving in real time and which has a huge viewership
        + E.g. real-time streaming of a sporting event
  + The capacity to have a more refined selection rule should be designed such that it can be seamlessly turned on for specific items without it needing to complicate the process of retrieving ordinary items
  + The set of hosts selected by the standard selection algorithm should be considered an item’s main controller group
  + An item, once retrieved, may contain details of a specialized selection algorithm for locating an expanded controller group for itself and possibly also for other items related to it
    - For example the semantic item representing a (possibly live) video feed may define a special selection algorithm for locating a set of hosts that can return a specific part of the video, defined by time elapsed from the start of the video
      * Users seeking to watch the video would first request the semantic item from the controller group selected according to the semantic item’s id (they would have found this ID from search results)
      * The semantic item would likely be cached by many hosts, probably including the one that returned the search results
      * The semantic item would contain a description of the algorithm that should be used for finding the auxiliary controller group for a particular chunk of the video
      * The selection algorithm might specify that to find the auxiliary controller group for a part of the video starting at time X from the start of the video:
        + Let Y = X/10s (rounded down)
        + Let H be the hash of the ID of the semantic item followed by Y followed by the current timeframe key
        + Let N be the number of hosts in a possibly custom set of candidate hosts
        + The entire auxiliary controller group for part Y of the video is found by repeatedly doing H mod N to select hosts from the set of candidates
        + The requester’s subset of hosts to send the view request to would be determined by using the selection algorithm over the auxiliary controller group and using a hash derived from the requester’s public key
    - How could the host reward system be extended to cater for custom selection algorithms?
      * It would not work to allow arbitrary selection algorithms or arbitrary sets of candidate hosts because that would allow exploitation of the reward system to target rewards to specific hosts
      * Anyone should be allowed to propose a new selection algorithms for use in some specialized circumstance, but it would need to be reviewed before it would become active.
      * For most circumstances, selection algorithms should be defined at the level of an item type such that it wouldn’t even be necessary to retrieve the item itself to know the selection algorithm for the auxiliary controller group.
      * The example selection algorithm defined above would be applicable to any video file (or other kind of file with a duration), so it should be defined as the default auxiliary selection algorithm for items of that type
      * Video files will vary in their demand and so the auxiliary selection algorithm would need to have a parameter specifying the size of the auxiliary controller group
      * Parameters for auxiliary selection algorithms would need to have restrictions placed on them because otherwise someone would specify an absurdly large auxiliary controller group size.
      * The allowed auxiliary controller group size could be limited by the reputation of the item. It should be specified by a decision market though because a high reputation may not necessarily be in high demand at any given time.
      * The auxiliary controller group size parameter, as specified by the decision market, would be updated at a known interval so that clients would know when the auxiliary selection algorithm they have been using may be out of date. If they have been following the item continuously, though, they will have been notified that the size parameter will change in previous requests.
* It should be possible for any valid Zutha ID to discover whether it has been allocated publically, privately or not at all
  + An item ID contains the ID of the host that created it
  + An ID’s creating host should know whether an ID in its namespace has been allocated publically privately or not at all
  + If an ID has been allocated publically then this should be discoverable by using the selection algorithm on it and sending a request to the controller group it returns
  + If the controller group that should be responsible for an ID does not know about it, then the ID will either by private to the creating host or not allocated at all. To find out which, one must ask the host that owns the ID’s namespace
* A private ID could be used by its creating host however it chooses
  + One application would be for interactive machines that use the Zutha standard as their interface
  + For example a car could be a Zutha host.
    - It would allocate private items for representing various aspects of its state.
    - The current speed, orientation, location etc. could all be represented by a private item
    - Some properties of the car could also be made public, but their editing privileges would remain locked exclusively to the car itself
  + A host could make a private item into a public one by making a special create item request, only available to name hosts, where the ID is pre-specified
    - The creation timestamp of the newly public item would be determined by the normal process though
    - The item would be considered by the network to have officially come into existence at the time it is made public
* Clients should be able to send requests that don’t include the target hosts in the request
  + A client may want the data item containing the part of a video (semantic item itemV) starting at timeX, but the client does not know the ID of that data item or the size parameter of the auxiliary selection algorithm for videos.
  + The client should specify the view request in terms of itemV and timeX and send it to… who? Some default neighbour? All neighbours?
  + Normally view requests are sent to a set of request handlers who then take responsibility for finding and processing the query
* I need to distinguish between requests made by clients and those made between hosts
  + Clients should only ever make view requests or edit requests
  + Both view requests and edit requests should always be addressed to a set of request handlers which are determined by the sender’s ID and a hash of the message itself
  + The controller group of an item is only of concern to other hosts
  + View requests between hosts will be targeted at specific hosts that are expected to have the data requested
  + The query language available to hosts should be much more restricted than the one used by users, including such things as
    - To the CG of the ‘Av’ index item: return the set of items containing the word ‘Avatar’.
    - To the CG of the Avatar.mp4 movie file item: return the full contents of the calculated field containing the entire movie file
      * The CG of the movie file item would then be delegated the responsibility of retrieving all the data blocks that make up the movie file and combining them using the appropriate bitwise operations to form the meaningful data.
      * Of course, it is likely that most members of the movie file item’s CG have already cached this calculated field
    - To the CG of a particular data item: return the data item
  + When a client requests a particular block of a video file, each member of the request handler (RH) group will do as follows:
    - If it has the requested video data in its cache it will immediately return it to the requesting client.
      * The message it sends will begin with the hash of the entire message
      * This will allow the client to read the hash and, if the hash matches the hashes of the other responses it has already received, ignore the remainder of the message
      * The client, upon receiving the hash, may even send a reply to cancel sending the rest of the message, which will have some effect at reducing the load on at least some intermediate nodes still carrying the message, and will possibly allow the RH itself to stop sending partway through.
    - If it doesn’t have the requested data it will request this data from the controller group that owns the data item or the auxiliary controller group that holds the particular block of the video file that is desired
      * The RH should include in its request, the view request that it is responding to
      * The controller group can then send out the needed data to all the hosts in the RH group, preceded of course by the hash so that they can cancel it if they don’t need it.
      * If a member of the controller group will be sending the same message to multiple hosts through the same neighbour, it can send the message to the neighbour just once and include multiple addresses in the header. This multi-addressed message can then be split up into separately addressed messages as needed along its path.
    - Once a RH has the needed data it will send it to the requesting client, preceded by its hash so the client can cancel the rest of the message as needed
* What is the incentive for an intermediate host to pass on messages?
  + Request handlers responding to user requests benefit from messages passing successfully between them and their client
  + It would thus be in the interests of an RH to reward its neighbour upon successful transmission of a message in both directions
  + If a node knows it may be rewarded by the RH for successful transmission of a message, then it has an incentive to reward its neighbour for transmitting the message as well
  + The client also benefits from his neighbours being cooperative in relaying its requests – well the user that owns the client does
  + A user can tell his client to randomly praise its neighbours with reputation for successfully returning the results of requests
  + A given intermediate node along a message path will not know how far it is from the client, and will have varying degrees of uncertainty about its distance from the target RH of the request. It will also not know whether its neighbour will happen to reward it on this occasion so its best strategy is simply to relay every message
  + Actually a node will build a reputation for how much it rewards its neighbours. Nodes that reward their neighbours more often will, in general, get better service from them. Again, though, if a node has plenty of resources available, it may be worth its while to serve even neighbours that reward service extremely rarely.
  + If your neighbour noticed that you never reward its message relaying services it might decide to stop offering that service.
  + Users with reputation could adopt a strategy of punishment for poor service instead of regular reward for service.
    - This would be particularly applicable to users who want to remain anonymous
    - It might not be an immediate neighbour causing the problem, but the user could punish whichever neighbour is not returning messages, under the expectation that that neighbour would pass on the punishment (as it sees fit) to whichever of its neighbours is causing the problem
    - A user who wants to remain anonymous might not want to reveal his identity and location to punish a neighbour for poor service. He could punish the poorly performing neighbour much later, though, after he no longer requires anonymity of identity/location
    - A node, faced with a neighbour that is not rewarding it at all for transmission of its messages, would not be able to be confident that that neighbour is not a high level user in disguise who may punish it severely for poor service later on.
  + A user would not have to reveal his identity for his client to punish and reward neighbours
    - A user’s client could earn its own reputation by offering network services to its neighbours
    - Whenever a user changes his client’s ID, his client would have to start again from zero reputation
  + Nodes would be able to find out the reputation of their neighbouring nodes so they would know how much power each has to reward and punish
    - However, for any message between two nodes, both nodes may benefit from the message passing because one might be the client and the other might be the RH.
    - If the other node might be the RH then it might praise you with its host reputation, which is unknown to you.
    - Similarly, if the other node is the client making the request then it might praise (or punish) you with its user reputation, which is unknown to you.
    - Hosts and users may want to praise their neighbouring nodes without revealing which neighbour they are
    - They could do this by rewarding/punishing neighbours after they have abandoned a node ID.
    - Someone (or some organisation) who owns multiple hosts or multiple users in different locations could praise/punish the neighbours of one host with another host they own.
  + There is going to be a need for a lot of Game Theoretic analysis to get the incentives right with regard to hosting and network services
* There is going to be an incentive for high reputation users to reveal their identity when they make view requests in order to get better service
  + Hosts will have an incentive to prioritize high reputation users who reveal their identity because there will be a much higher chance of them getting a payoff in that scenario than for an anonymous request that has a small chance of being a high rep user and an even smaller chance of being a high rep user who will identify themselves after the request.
  + This will only be a major problem when hosts are overloaded since if they have resources to spare it may still be worth the small cost of computing resources for the small expected payoff of an anonymous user

# Sun 12-Jun-11

* Intermediate nodes would have an incentive to respond to a request if they can, just on the chance they might be rewarded with reputation
  + High reputation users who have a stake in Zutha doing well would have an incentive to reward intermediate nodes that are generous
  + Actually any user would have an incentive for rewarding good behaviour in intermediate nodes, especially those close to them which will likely act as intermediate nodes for many of their requests
  + High reputation users would even have an incentive to praise other users who praise intermediate nodes for good behaviour – especially those that make the request anonymously
    - Actually it wouldn’t be possible to know whether a user was acting anonymously in any given situation, so it would be risky to act on this assumption
    - That is, a user could always communicate his identity to a given node prior to, or at the same time as a request to, without anyone else knowing
* There should be a special kind of host reputation which is only attainable via the automated reputation system – that is, hosts can only earn this kind of reputation when they are selected as request handlers by the selection algorithm, the requesting user happens to have reputation and host rewarding currency available, and a random throw of the dice determines that the host should be rewarded for this request.
  + This special reputation would be used to rank hosts within a level category and select hosts to move up or down a level at the end of a ‘round’
  + It would just be a domain specific reputation that no one can invest in manually
  + The purpose of using this special kind of reputation is so that entities with a lot of standard reputation cannot rapidly advance their hosts to high levels
* Data items should be named by the hash of their contents
  + Since names must be unique, this will prevent duplicate items containing the same data
  + When a user tells his client to upload a file, his client can divide up the file into parts of standard size, make a hash of each, and check whether any of those data blocks are existing data items on the Zuthanet.
  + This is similar to what dropbox apparently does to avoid users needing to upload files that someone else has already uploaded to Dropbox’s servers

# Tue 14-Jun-11

* The process of allocating Zuth according to reputation is going to be complicated once the database is distributed
  + Practically any item can have reputation, so it will probably be necessary to assign responsibility for paying each item to that item’s controller group
  + Unlike with Zutha’s donations of real money, there is no central starting point from which to allocate zuth to a finite set of top level items to whom the responsibility of distributing payment of zuth further can be delegated.
* For payment of real money by zuth to users with particular kinds of reputation a more centralized model will be possible
  + Zutha will have an explicitly defined policy of which items it donates to
  + These might be items which represent categories such as Philanthropic Causes or Zutha Development
  + Each of these items will then be delegated the responsibility of paying its own beneficiaries
  + Thus the payment of real money by Zutha - that is: Zutha’s donations - are just like any other donations. They start at a specific entity and propagate outward sort of hierarchically
* On what basis should Zuth be paid?
  + Should it be paid to all items according to the total reputation of each?
    - Each item would then be responsible for paying its owners
    - The problem with this is that it makes obsolete the limitations on contributions to a user’s reputation from a given source. A powerful user UserX could invest in many intermediate items who would all be owned by a target user UserY. UserY may be limited in the total allowed reputation he can gain by investments of UserX’s, but UserY would still get the full payment of zuth that he would get without any restrictions on UserX’s allowed contributions to UserY’s reputation.
    - I think it best if the concept of items paying their owners is applied only to real money.
  + Should it be paid only to Active/Sentient entities (those that can take actions such as Users or Organisations)?
    - This seems like a difficult constraint to apply unambiguously. Some items may begin as “insentient” (as it were), but become sentient when some real world entity claims them.
    - Then again, there is a pretty solid semantic distinction between the kinds of entities that could conceivably be considered sentient and those which could not
    - There should be a Type which represents a Sentient item, of which only subtypes are allowed to become sentient
      * Thus User and Organisation would subtype Sentient Entity (or maybe Active Entity or Agent or something)
    - Maybe only “Agents” (I haven’t decided on the best term yet) should be considered to have reputation.
      * I started with this model – Users would have reputation, which would be the sum of the worth of all the items they own
    - Agents would be paid in Zuth according to their reputation
      * This would allow rules that restrict reputation to apply prior to this value being used to determine an Agent’s income in Zuth
    - If there were a single type of item of which only its subtypes can be paid in Zuth, a centralized model of payment would be conceivable
      * This model would make it realistic to specify a specific amount of zuth to inject into the economy and then divide that up among all agents according to reputation
      * Of course, this model would still allow a distributed model of Zuth payment, where a rate of pay per unit of reputation is defined and the payment of each Agent is calculated independently by their controller group
* Zuth could be paid at a constant rate according to an Agent’s reputation at each instant of time
  + This would require a lot of calculus on the part hosts, but also clients - to show the smooth change of zuth over time
  + Payment could occur at regular intervals instead, but this would require many event items
  + If payment was infrequent, but occurred at the same time for all Agents, it would create a very significant cyclic effect in investment activity

# Wed 15-Jun-11

* If zuth was paid at discrete intervals – especially long ones – it would enable an exploit where users invest lots in a target user just before payday and then sell their investments again afterwards.
  + If the unreasonable investment never stayed in the target user for long, it would be difficult for the offenders to be effectively punished because they could avoid being caught in the actual act of the crime. Thus their crime would be less salient and fewer users would be willing to punish them for it.
  + If zuth was paid continuously, this exploit would not work because the spike in a user’s reputation would only increase their rate of income for the period that it lasted.
* If Zuth is paid continuously, there would need to be a mechanism to cope with errors – or at least, delays in information reaching a user’s controller group regarding that user’s current reputation.
  + Things could get very complicated if a user invested using zuth he should not have had and some of the investment needs to be revoked retrospectively.
  + The level of reputation on which the current rate of pay is based could be some period of time in the past (a minute or so? Maybe even an hour?) – enough so that there is a high degree of confidence that a correct value of reputation (for that moment in the past) is being used.
  + Alternatively, or in addition, there could be a limit on the percentage of one’s zuth that is allowed to be used (say 90% or something). This buffer would only be allowed to be used up for cases when the controller group didn’t have the latest information regarding the user’s current zuth balance.
    - I don’t like this idea. It’s messy.
    - And how to decide what precisely is a context in which the buffer is allowed to be exceeded?
    - What if different members of the controller group have different data for a user’s current zuth balance?
  + If rate of pay (in zuth) is based on reputation at a sufficient, fixed time in the past, then it will be possible to be confident that a user’s current zuth balance is accurate.
  + A user’s zuth balance should only be able to change via their salary in proportion to reputation (positive or negative), by voluntary investment or by voluntary or involuntary sale of investment.
    - Involuntary sale (or reversal) of an investment is the only one of these that can occur at any moment. However, since it increases a user’s current stock of zuth, it is not a problem.
    - In order to be sure that a user is allowed to invest a certain amount right now, the controller group must only be certain that his stock of zuth cannot possibly be less than they currently calculate it to be.

# Fri 17-Jun-11

* Listening to the start of The Master Switch, it strikes me that there is a real risk for some kinds of technology that a single corporation could gain control of the fundamental infrastructure and remain in a monopoly position because of huge barriers to entry thereafter.
  + The main problem would be that once the infrastructure of the new technology has been built once, network effects would make it more valuable to customers than even a much better implemented rival which doesn’t yet have the same coverage.
  + I guess this scenario is what economists would call a natural monopoly.
  + What struck me was that this scenario seems to be a case in which it would be desirable for a society to have some system in place to prevent the infrastructure for the new technology from being dominated by a single corporation. A more desirable outcome would be for a set of standards to be developed and incentives created such that it is in the interests of market incumbents and entrants to make their infrastructure interoperable with their competitors over the standard interface.
  + This seems to me to be an example of where some kind of coordinated force could be valuable for bringing about a more socially beneficial outcome.
  + It’s difficult to see how anything like a conventional government could do more good than harm in this matter. As soon as there is central control, that central power will face incentives (whether through the greed or power-hunger of individuals, or lobbying by powerful corporations) to propagate the centralization of control (for example by creating a government granted monopoly) – there will certainly be no incentive for centralized power to deliberately work towards real decentralization of power.
  + But I used the word coordinated instead of centralized to refer to the kind of force I think might be needed to prevent natural monopoly. A coordinated force need not be centralized. That is one of the game-changing ideas I think Zutha will demonstrate.
  + I think it would be possible for a coordinated policy to arise organically out of the enlightened self-interest of a dispersed minority of society. It would arise out of their self-interest, but would be in line with the interests of society because the environment of reputational incentives created by Zutha would align their self-interest with the interest of society as a whole.
  + An effective anti-natural-monopoly policy could, I think, take the form of as apparently intangible a phenomenon as a cultural norm that condemns companies that do not provide open interfaces for interoperability with their competitors.
    - Granted, the feasibility of creating a culture powerful enough to overwhelm the commercial benefits of lock-in is a long shot, but the increased power of reputation created by something like Zutha may make such it a realistic possibility.
    - An anti-lock in culture may not even need to be powerful enough to force all firms to cooperate; even if it is a powerful enough force to induce a few small competitors to try working together, it might create a powerful enough cooperative group to take on a powerful incumbent.
    - There is already an incentive for small competitors of a near-monopolist to join together to compete with it, but this too often takes the form of mergers between the second tier of competitors with something like duopoly being a more likely outcome than a return to vibrant competition. Often when 2nd tier firms in a market merge to compete with a single dominant 1st tier monopolist, they are liable to ignore the smallest players in the market as of too little importance to worry about.
    - What would be powerful about a reputational force that encourages firms to adopt open standards of cooperative competition (think ISPs), is that it would make standards a primary unit of natural selection in the market. If a few medium sized firms adopted a standard, they would automatically – without any extra effort on their part – allow new entrants to join them to compete with the bigger players. New entrants would join the standards they think look most competitive. Incumbents would have an incentive to switch to the most successful standards to benefit from the network effects. Open standards (including many independent firms) would compete at the same level as individual closed firms.
    - Linux versus Microsoft is an example of this war between an open standard with many independent members and a monopolist with its own private standard. My hypothesis is that in an environment like the one Zutha will create, where reputation is more powerful and more ubiquitous, and with a concerted effort by some influential people to create a cultural expectation of openness, the battle between Microsoft and Linux would swing in Linux’s favour – or at least, in the favour of some open-standards based competitor.
* Facilitate the creation of an industry for verifying people’s identities and vetting people for trustworthiness and perhaps other character traits.
  + If there were many independent firms competing for the best reputation in accurately assessing trustworthiness, I think the best of them would do a far better job than a typical government security vetting process.
  + Such a service would be valuable for setting up an open source intelligence network among other things.
  + People could potentially ask a private vetting agency to be vetted and then ask them to associate the vetting results with their anonymous Zutha user.
  + Actually incentives would be aligned better if the agent (X) paying the vetting agency was the one who wanted to know the trustworthiness of the anonymous Zutha user (U). X would be able to select the agency that he wants U to be vetted with. When the agency posts the results, X would pay the agency. X would never need to know U’s real identity, but if X trusted the selected agency, he would be able to trust that U is trustworthy.
* One of the services that will require trust that the Zutha network will need is the creation of client hardware and eventually bots of various kinds for defence and other services for Zutha communities.
  + It would be essential that reliable ways be found for such hardware to be vetted for authenticity – that is, free of devices for surveillance or sabotage).
  + One way this might be achieved is by batches of devices to be randomly sampled by a variety of vetting agencies, each of whom stake their reputation (and thus income) on their honesty and the thoroughness of their inspections.
* It should be possible to search for currently active conversations by various criteria including comments per minute, total rate of investment in the parent item and the comments combined etc.
  + Cyberspace can feel effectively deserted even when there are millions of people currently visiting the same (or similar) pages as you.
  + I think this must have a detrimental effect on an individual web surfer’s tendency to contribute to web conversations. It is clear that most of the comments on most pages are created near to each other in time, indicating that people are more likely to add their input when they feel that they have company.
  + I think it would be powerful to set up a mechanism which attracts people to currently active conversations. I think this would lead to a net increase in user input as well as, probably, much higher variability in activity across pages/items due to network effects.
  + While a mechanism that broadcasts current user activity would undoubtedly boost the amount of activity at popular sites/pages/items, I think it would also substantially increase activity at relatively unpopular ones; it would “fatten the tail” of the activity distribution, as it were. This is because such a mechanism would facilitate the congregation of a small and dispersed set of interested individuals who might not have discovered their mutual interest at all otherwise, and may thus have never been inspired to start a conversation at all.
* Users should be able to search for other users who are members of many of the same sets as they are – sets being things like membership of a club, being a fan of a sporting team, living in a city etc.
  + Martin Nowak suggested in SuperCooperators that social sets like these are fundamental to cooperation – that cooperation forms around mutual membership of sets.
  + People are more likely to trust someone who is a member of many of the same sets as they.
  + If Zutha makes it easier for people to find others who are members of the same sets as they, it will facilitate the creation of a greater number of cooperative groups
  + This is one of the powerful effects of social networking. It seems to me though, that mutual set membership discovery could be facilitated far more effectively than it is with current social networking services like Facebook. Facebook (as far as I am aware) has no feature for identifying people who share membership of many sets with you. It allows you to filter search results by a single criteria at a time such as High School or City, but does not (for example) find people in your city who are fans of the same football team as you, like many of the same movies as you, share a passion for hiking, have the same political affiliation as you and attended the same karate club as you 10 years before you did.
  + There are many compelling sources of potential for bonding other than mutual friends, and many that would be more likely to create productive relationships, but as far as I can tell, current social networks do not exploit these – not effectively anyway.

# Wed 22-Jun-11

* Zutha queries should be a smooth extension of standard search engine syntax as well as Lucene Syntax
  + Item fields should be searchable with the syntax
    - author:”J.K. Rowling” philosopher
    - birthdate:2001-11-17 type:person
  + The text immediately before a colon may be interpreted either as an Occurance or a Role Type (in Topic Map terminology)
  + An explicit Association Type should be specified when referencing a Role Type by putting it before a forward slash as follows
    - Authorship/author:”J.K. Rowling” philosopher
    - birthdate:2001-11-17 type-instance/type:person
  + Actually I want it to be possible to specify multiple other Roles within an Association, but this slash syntax only permits one
  + What about
    - Authorship(author:J.\_K.\_Rowling) philosopher
    - Authorship(author:Janny\_Wurts, author:Raymond\_E.\_Feist)
    - Marriage(date:[1989-12-25 TO 1990-01-01], place:Melbourne)
* The Zutha standard should not require the underlying data model to be based on topic maps
  + The standard should simply define an abstract set of rules which need to be followed regarding item relationships and allowed actions
  + This means topics and associations used as part of Topic Map Constraint Language should not have Zutha IDs. TMCL can be used optionally to implement the constraints specified by the standard and extended by item types, but it should not be exposed to the public interface that TMCL is used to do this.
* The Zutha API should be built on top of a tree of Action items
  + An Action item will describe an atomic action which can be performed on the Zutha knowledgebase
  + Every Event item will be associated with exactly one successful Action item
  + Action items will specify the format of the input they require and will describe a set of rules for verifying the validity of that input
  + High level Action items delegate actions to lower level Action items
  + The lowest level Action items will be inbuilt and will define actions like
    - Create Association
    - Create Item
    - Modify Property
    - Add AssociationRole
    - Remove AssociationRole
  + Higher level action items will be built on top of these such as
    - Create Type
    - Create Item with Properties (perhaps with JSON input)
    - Invest in Item
  + Users will be able to make their own higher level Actions with their own custom validation rules
    - Since user actions must use lower level Actions to actually cause any changes, this does not pose a security concern since the low level Actions still validate themselves according to low level permissions and other validation rules.
    - High level Actions only allow a subset of the functionality of lower level actions – they are like macros, with extra restrictions rather than diminished restrictions
  + Custom Actions would be used to build the functionality of custom Applications on top of the Zutha architecture
    - For example in the Zuthaverse MMOLG there would be actions like
      * Start Moving (in given direction)
      * Cast spell
      * Send chat message to player
    - Hosts would be free to implement the execution of Actions however they want, as long as they are able to respond to any request as expected by the standard
    - Some hosts might implement specialized handlers for complex Actions (like game actions) which increase the efficiency of their execution.
      * For example they might have custom data structures for representing the game state which they modify directly so they can quickly return the time-critical response to the user.
      * After completing the time critical component, the host could then take its time performing the lower level actions on which the high level Action is built
      * Low level Actions that might be delayed could include the creation of new events (of which a host must participate in), notifying other items of their inclusion in a new Association, or modifying a property of an existing item. This latter could even be omitted altogether; the Zutha format item could simply be calculated on demand from the custom internal data structure.
* Use Zutha.net as domain name
  + Zutha.com, zutha.org (and eventually Zuthanet.com, Zuthanet.org, Zuthanet.net) should redirect to zutha.net

# Sun 26-Jun-11

* Seed Zutha communities in poor neighbourhoods
  + Provide Zutha clients to many members of the community so they can start to build a stronger framework of reputation
  + Invest resources in mentoring individual children and perhaps whole families at a time
  + The ability to tap into a planet-wide information economy could provide even relatively uneducated people with a powerful lever to lift themselves out of poverty.
    - I sense that people, particularly children, in disadvantaged neighbourhoods who find creative avenues through which to display talent and dedication will attract the admiration of many watching members of the Zutha community and may be able to earn more money and reputation through Zutha than similarly skilled people under less disadvantaged circumstances.
  + This framework could be an effective way to launch my idea for viral, self-perpetuating education where teachers mentor students for free with a (possibly implicit) expectation that they will repay the investment over their lifetime in proportion to how much they earn. Some students would become teachers themselves and their teachers might even continue to invest resources in them until some of their students start paying back their debt.

# Wed 29-Jun-11

* Item attributes should be able to be assigned to inherent reputation from specific types of subsidiary items.
* For example an action item representing a user making additions to the zuthanet database could contribute to that user's reputation in the attribute "zutha contributor".
* Actually this feature is already better provided by the concept of users having reputation in multiple domains. It might just overly complicate things if reputation in an attribute can come from both direct investments and reputation inherited from owned items.

# Thu 30-Jun-11

* Set up a system that allows people to invest in Zutha early on and enter a pool of special status early investors who will become beneficiaries of Zutha’s income later on.
  + Some percentage of Zutha’s income would be assigned to be divided up among that group of crucial early investors according to amount invested

# Fri 1-Jul-11

* Items should define a max cache age for zero-reputation users and a function should define how that value decreases as a user gains reputation
  + A cache age is how old the data used for a response is permitted to be
  + Example: a user makes a request for all audiobooks. The request handler made that request 23 hours ago and the Audiobook item defines that its max cache age for zero-rep users is 24 hours. So the request handler simply returns the same results it returned 23 hours ago.
  + The max cache age for the Audiobook item might be 12 hours for users with >0 rep; 6 hours for users with >10 rep; 3 hours for >100 rep; 1 hour for > 1000 rep; 30min for >10000 rep; 10 mins for >100,000 rep; etc.

# Fri 8-Jul-11

* All communication between nodes should be done by sharing topic map fragments.
* Clients should then query their local database to extract data in whatever form they want it in for display (or whatever other purpose).

# Sat 9-Jul-11

* It would be cool if encrypted programs could be passed around the Zuthanet and run in distributed fashion by randomly selected hosts
  + There should be a generalized architecture for passing around code that seamlessly integrates encrypted and publically accessible code in the same system
  + The system should scale to the task of running full distributed MMORPGs
  + The only code that would be runnable using this system would be in a language that is only capable of sending commands to the Zuthanet - that is, its only IO would be Zutha messages; apart from that the language would be functional in nature and would simply transform input to output.
  + The kind of system I’m envisioning would be capable of doing a large part of what the Daemon does in Daniel Suarez’s novels
  + Consider a simple distributed daemon whose task is to send a particular message over the Zuthanet upon some event occurring.
    - To keep things simple, let’s assume that the event is simply a specific predetermined time in the future and the message is a predetermined static message.
    - The simple script defining these rules would be stored in a Zutha item of type Zutha Program (or something to that effect)
    - As with any other item, the Zutha Program item will have a controller group that will change over time
    - Each host in the Zutha Program’s controller group will be responsible for registering (in whatever way it chooses) the event triggers specified by that program
    - When the event occurs (in this case when the predetermined time is reached) the current set of hosts in the Zutha Program item’s controller group will be triggered to run the script associated with that event.
    - In this case the script will specify that a prewritten message should be created as a Zutha item
    - Each host in the controller group will execute the script and they will coordinate with each other such that a single message will be created with all the hosts specified as the creators.
  + The scripting language will be fundamentally integrated with the Zutha environment such that a command to send a message will be transparently transformed into a coordinated multi-host action whereby a single Message item is created with the event being signed by all the hosts that successfully participate in the multi-agent action.
  + There will need to be a dedicated kind of message in the Zutha protocol for coordinated actions like this. The process will be similar to that undertaken by a group of request handlers when they coordinate the selection of an ID and timestamp for a user action for which they are the selected request handlers.
  + A Zutha Program could be encrypted
    - The item contents containing the code would be encrypted and the decryption key given to each host that enters the controller group
    - The code of the program would thus become less secret over time as more hosts are given access to it
    - A specialised selection algorithm could be used that is specific to a particular Zutha Program and which only includes hosts trusted by the program writer as candidates
  + Much more complex event triggers than a simple time trigger would be possible
    - For example the trigger could be an arbitrary condition on the value of a particular kind of message sent by a certain category of users to a particular target
    - Call the conditions specifying the message to watch for the Message Conditions
    - Let GroupX be the controller group of the Zutha Program
    - Let GroupY be the controller group of the target of the message to watch for
    - GroupX could send a request to GroupY asking to be notified when it receives a message matching the Message Conditions
    - The ability to request notification of updates related to a particular set of items should be a standard part of the Zutha standard
    - GroupX would delegate the task of detecting messages that might match the trigger conditions of their Zutha Program. Each time GroupY sends them such a message, they would check it’s value against the Value Condition specified by the program to see whether the trigger should be executed.

# Mon 18-Jul-11

* Events should be associated with the earlier events that they depend on
  + When a request is made to undo/reverse a past event, all the events that depend on it will be reversed first in order of most recent to earliest
  + Some example rules for determining the events that a new event depends on
    - Adding a topic type
      * Let event EX be an Add Topic Type event which makes the topic TT a type of topic TX
      * EX depends on the event that created TT because TT needs to exist to become a type of TX
    - Adding an association
      * Let event EX be an Add Association event which adds an Authorship association between author topic t\_author and book topic t\_book
      * EX depends on the event that created t\_author.
      * EX also depends on the event that gave t\_author the type Person, because there is a constraint on the Authorship association that requires the topic playing the Author role to be one of a constrained set of topic types of which Person is one.
      * EX depends on the t\_book creation event and the event that made t\_book a Book
      * EX depends on the Authorship association type topic
        + The Authorship association type topic will in turn depend on the events that created the role type topics it uses among other things

# Wed 20-Jul-11

* Should an item’s worth include the worth of the items it owns?
  + It would be confusing if items have more than one score: both the amount of zuth directly invested in it and the net worth of adding up the worth of all owned items
    - It is already going to be necessary to have more than one score per item though: there will be a score for both zuth and real money, as well as a worth comprised of the investments in each attribute
    - I don’t think it will be a multiplicity of scores per item per se that will necessarily be confusing, but it would be important to make sure each score has a comprehensible meaning based on a solid mental model.
    - One thing that would be crucial is that investors are easily able to discover the value that represents the current total size of the investment pool they are investing in.
    - Perhaps investment pools should be made independent concepts from semantic items. Actually I had already begun to think in that way when I devised the idea of items having attributes which would be invested in independently.
    - Investment pools could be thought of as the leaf nodes of the reputation hierarchy, where the reputation of items and users would be derived from the reputations of the items beneath them in the hierarchy.
  + In some conceptual domains it makes a lot of sense for the value of an item to be derived from the value of subsidiary items
    - Argument maps are an example: a node’s worth should be derived from the worth of the arguments for and against it
    - What about the kind of ownership relationship that exists between a character in a book and the book itself? If people like the character, should this automatically contribute to the derived worth of the book?
  + I think it would be more difficult to calculate reputation – particularly domain specific reputation – if every item’s worth were derived from a hierarchy of subsidiary items
* How is domain-specific reputation calculated?
  + An investment pool is the fundamental unit which contributes to a user’s reputation
  + A single investment pool (just call it “pool” from now on) may contribute to the worth of many items because items can be owned by more than one parent item
  + A pool may contribute to the reputation of an item via more than one route through the reputation hierarchy
  + The amount of reputation contributed by a pool to an item higher up in the hierarchy will be scaled by some factor which will be derived from the product of multiple scaling coefficients contributed at different points in the chain from pool to item.
  + A scaling coefficient is applied to each contributing pool when a parent item only owns a percentage of the child item. The coefficient applied is the percentage of the child item owned by the parent item.
  + Consider the scenario of calculating the reputation of a user U in domain X
    - Domain Y is one degree of separation from X.
      * Y is one degree of separation from X if there is an association between Y and X which is a subtype of the Related Domain association (I might think of a more appropriate name later)
      * For example, Y could be a parent domain of X or a child domain of X or some other domain related to X in a way that merits a flow of reputation across that boundary
    - The association between Y and X specifies that p% of U’s reputation in domain Y should contribute to U’s reputation in X
    - All the coefficients of the pools contributing to U’s reputation in Y should be multiplied by p before being combined with the pools from other domains that contribute to U’s reputation in X
    - When merging sets of pool/coefficient pairs, if a pool appears in multiple sets, in the merged set it will be associated with the highest coefficient it had in any of the original sets.
    - Eventually a single set of pools should be derived where each pool is associated with a coefficient and appears just once.
    - The set of pools should be unpacked into a set of investments, each of which will be associated with the coefficient of its parent pool.
    - The set of investments should be put into groups according to the investor that owns them
    - Each investor’s contribution to U’s reputation is calculated by first finding the sum of the product, for each of that investor’s investments, of the investment value and its associated coefficient. If this value is greater than the investor’s allowed contribution to U’s reputation, then the investor’s contribution equal’s his maximum allowed contribution.
    - U’s allowed contribution to himself is zero, as is the allowed contribution of any users suspected of being controlled by the same agent as U.
    - Every other user’s allowed contribution to U’s reputation in domain X is their own reputation in domain X
    - If a user’s reputation depends on other users’ reputation in the same timeframe then an infinite loop can be created. Therefore, when calculating the maximum allowed contribution of an investor V to user U in domain X during timeframe t, V’s reputation in domain X during timeframe t-2 should be used. (see below for a discussion of calculating reputation and item worth in discrete timeframes).
  + Example scenario: Einstein’s reputation in special relativity
    - Einstein will have reputation in physics in general from contributions such as his work on the Photoelectric Effect
    - Physics is a pretty closely related concept to special relativity (it might even be a direct parent in the reputation hierarchy), so a reasonable amount of the reputation Einstein gains from his work on the Photoelectric Effect will contribute to his reputation in special relativity.
    - The influence of investments in Einstein’s work on the Photoelectric Effect will be diluted at least twice though. They will be multiplied by one coefficient in the transition between the Photoelectric Effect and physics, and multiplied by another coefficient in the transition from physics to special relativity.
    - An investor V has high reputation in the domain of the Photoelectric Effect and low reputation in the domain of special relativity. He wants to increase Einstein’s reputation in Special Relativity as much as possible.
      * If he invests in items owned by Einstein in the domain of the Photoelectric Effect his influence will first be diluted by the separation factor between the Photoelectric Effect and Special Relativity, and then capped at his reputation in Special Relativity.
      * If he invests in items owned by Einstein in the domain of Special Relativity (Einstein might be a partial owner of the topic of Special Relativity itself for example), then his influence will not be diluted, but it will still be capped at his small reputation in Special Relativity.
    - An investor W has high reputation in the domain of special relativity and low reputation in the domain of the Photoelectric Effect.
      * If W invests in Einstein in the domain of the Special Relativity, his investments’ influence will be undiluted and the cap on his influence will be high because W’s reputation in Special Relativity is high.
  + Roughly speaking, an investor V can only influence a user U’s reputation in domain X in proportion V’s reputation in domain X and a particular investment’s influence on U’s reputation in domain X is proportional to the degree of separation between the parent item of the investment and X.
* How is item worth calculated?
  + If item X owns only (investment) pool items, then it’s worth is calculated as follows
    - Define item X’s “pool set” (this term will be used later) as a set of pool/coefficient pairs, one for each pool directly owned by X, where the coefficient in every case is 1 (a pool can only be owned by 1 item).
    - …
  + Essentially, item worth is calculated the same way as user reputation except that relevant pools are only aggregated through ownership links and the only dilution to an investment’s influence that occurs is when an item is only partially owned by a parent in the relevant hierarchy.
  + An investor V’s maximum allowed contribution to the worth of an item X is V’s reputation in domain X.
* Calculating an item’s worth will be potentially extremely computationally expensive
  + It would be impractical to expect every item’s worth to be calculated in real time every time a result set needs to be returned sorted by worth
  + The worth values of items should be updated at regular intervals – probably in alignment with timeframes
  + If a view request is made in timeframe t, the worth values of the results should be based on all data up to and including timeframe t-2 (that is, up until the start of timeframe t-1). During timeframe t, hosts should be calculating the effect on item worth values of investments that occurred during timeframe t-1. They will need to have finished by the time timeframe t+1 starts.
* A user U can invest as much currency as she has in any pool, but the influence of that investment on the worth of an item X is limited by U’s reputation in domain X.
  + This means that the only thing that needs to be calculated to check whether a requested investment is valid is whether the user has sufficient currency (zuth or money) in their account. Since this is easy to keep track of in real time, investing should not incur significant delay.
* What is the computation complexity of calculating the reputation of every user in every domain each timeframe?
  + So far I have been working under the assumption that practically every item counts as a domain – even things like comments. That means there is an independent reputation calculation for every user/item combination, so (items\*users) calculations.
  + Each reputation calculation (U’s reputation in domain X) takes as input the investments in all items connected by any number of degrees of separation to X. A worst case scenario approximation would assume the number of investments to be the total number of investments in the database.
    - There would probably be some log factor here to account for the fact that many pools of investments have their coefficients multiplied by something for each degree of separation between them and X.
    - A calculation needs to be done on each investor that contributes to U’s reputation in X to limit their influence to their reputation in X. A worst case scenario approximation would put that at the total number of users in the site.
  + So in the worst case scenario the time complexity would be something like: users^2 \* items \* investments
    - Taking values of: users=10^7; items=10^9; investments=10^10…
    - Time complexity would be: (10^7)^2\*10^9\*10^10 = 10^31
    - I have no idea how to estimate the constant factor, but let’s just say it is something like 1000 floating point operations for every one of the 10^33 calculations
    - If a timeframe is, say 2 minutes long, then there are ~100 seconds to complete 10^36 floating point operations, so the total computing power required for this task by the Zutha network would be 10^34 FLOPS.
    - The most powerful supercomputers today operate in the range of 10^15 FLOPS. Hmmm…
  + I don’t think the worst case scenario would accurately reflect the actual time complexity in a real life scenario. There would probably be more log factors, for starters. For example, the number of investments connected to a particular item would probably not scale linearly with the total number of investments in the site, since a large part of the expansion would be horizontal – that is, into new, independent domains.
  + A great deal of the calculation has been done in the previous timeframe.
    - The contribution of investor V to U’s reputation in domain X in timeframe t is just what it was in timeframe t-1 plus the value of each investment of V’s that is connected to X.
    - Only investors that actually invested this timeframe need to be taken into account
    - This brings the time complexity back to users\*items\*(users that invested this timeframe)\*(investments this timeframe)
  + Every item need not be a domain
    - Items like comments should not be their own domain
    - Domains need only be as specific as “Harry Potter (fictional universe)” or “Bayesian Probability Theory”
  + Only users that have been active recently need to have their reputations recalculated immediately
    - Users that invested recently need to have their reputations updated because this information might be needed to calculate the worth of items or the reputations of users.
    - No this is wrong: passive users can have their reputation change because of the investment activity of other users and this will in turn change the influence of their existing investments on other users’ reputations.
  + For each investor V that contributes to a user U’s reputation in domain X, only V’s investments this timeframe need to be examined for new investments that need to be added to V’s total contribution to U’s reputation in domain X.
    - So we only need to multiply by the average number of investments per user per timeframe
  + The time complexity is now (users with reputation)\*(domains)\*(users that invested this timeframe)\*(avg. investments per investing user this timeframe)
  + Let’s use values of
    - Users with reputation=10^6
    - Domains=10^4
    - Users that invested this timeframe=10^3
    - Avg. investments per investing user this timeframe=1
    - Time complexity=10^6\*10^4\*10^3\*1\*10^3/10^2 = 10^14 FLOPS
    - There! I just need a 2008 grade supercomputer. In a decade or so, a distributed supercomputer like the Zuthanet will probably be able to do this with ease.
  + There should be a maximum number of degrees of separation between a domain X and domains that can influence a user’s reputation in X
    - This will dramatically reduce the degree of dependency between different user’s reputation values and I think will make lazy evaluation of domain-specific reputation more viable.
  + Use of indexes will probably be able to reduce the time complexity much further

# Sun 7-Aug-11

* Zutha should use an abstract data model that can be implemented with topic maps
  + In the abstraction, associations should be allowed to have literal value properties. This would be implemented in topic map format as a reifier on an association which has the properties as occurrences.
  + Items which represent types should have fields which specify the constraints on their instances.
    - These fields should map to TMCL templates
    - In the abstract zutha data model, the constraint fields will be single associations which include literal properties for cardinality and regexps.
    - For example, the topic type Author would have an association of type “plays-role” with fields role-type: Author, association-type: Authorship, card-min: “0”, card-max: \*.
      * i.e. plays-role(role-type: Author, association-type: Authorship, card-min: “0”, card-max: \*)
      * note that plays-role is the name of the TMCL template which generates the TMCL constructs that define this contraint
  + Constraint associations like “plays-role” should be real associations in the zutha abstract data model and in the topic map representation of that data model
  + “Association type” items which represent constraint associations like “plays-role” should be linked to topics which specify and document this association’s representation in various implementations of the Zutha abstract data model.
    - For example, there would be an item for the Topic Map implementation of this constraint association which would specify how to map associations like “plays-role” to TMCL.
    - Basically, the topic map implementation would simply be to execute the CTM template associated with the constraint association (like “plays-role”)
    - The TMCL associations and topics created by executing this template would be included in the topic map representation of the zutha knowledgebase and would be passed around between hosts that use this implementation, but these constructs would not have Zutha IDs (ZIDs) and would not exist in the abstract representation of the knowledgebase. This means topics of type tmcl:topic-role-constraint and such will not be accessible through the Zutha API orquery language and will not have their own webpage dedicated to them in web portals to the Zutha knowledgebase like Zutha.net.
  + There will be a need for some constraint associations that do not map to existing TMCL templates
    - For example, there will need to be an “Association Property Constraint” which specifies the literal properties as association is allowed to have. The property type will be specified in the same way as occurrence types and will be drawn from the same pool of typing topics – that is, Property Type items are Occurrence Type topics.
    - The Topic Map implementation of this constraint would be a CTM template that specifies that the association must have a reifier with an occurrence of the type given for the property type.
  + Reifiers - both the concept of reifiers as well as topics which are reifiers in a topic map implementation of the Zutha Abstract Data Model (ZADM from now on) - will not be present in the ZADM.
    - The topic map implementation of the ZADM should specify a TMCL constraint which allows delegates constraints on reifying topics to the constraints specified for the topic they reify
    - For example, associations may have an association-property-constraint which allows a literal property in the association (such as a cardinality value). This will map to a topic map constraint on the reifier of the association.
    - The reifier itself should not have a constraint specified in its type because there may be multiple constraints on different super-types of the association type which affect the properties its reifier is allowed to have.
    - For example, many constructs in the ZADM, including associations, will have a constraint specifying they may have a literal property specifying the permission level (in the form of reputation) required to modify that construct.
  + The “Item type” item (which will have a subject indicator tmcl:topic-type) will specify the constraint fields that are allowed in Item Type items

# Wed 10-Aug-11

* I want users to be able to learn how to use the query language by seeing the equivalent query string for the results page they have browsed to
  + For example when a user is on the page: “/item/07F5/Janny\_Wurts/wrote-works/written-works” they should be able to see that this is equivalent to searching for: “written-work:written-by(author:Janny\_Wurts)”
  + I’ve been trying to think of some way to unify the concept of being on an item’s Role page (such as “/item/07F5/Janny\_Wurts/wrote-works/written-works”) and being at the equivalent search results page. This way, the equivalent query string could be put in the search box when a user browses to an item Role page.
  + I can’t think of a way to make this work elegantly though, because there are some fundamental differences between an item Role page and a search results page. Specifically, an item Role page for X has X’s summary stub at the top of the page, and for each item Y in the list, includes other properties from the association linking Y to X.
  + I think the best way to satisfy my goal of having users learn to use the query language through browsing, is to show a link somewhere (maybe between the control bar and the main item) that is labelled “equivalent query” or something and shows the query string – and a link to the results page – that would result in the same set of items as shown (though without the additional context specific properties).

# Fri 12-Aug-11

* Binary associations should appear on item details pages in the same format as occurrences
  + The name of the association will therefore be omitted along with the accompanying link to the association-table page.
  + It will still be possible to access the association-table page by manually navigating to the url, but association-tables for binary associations would be of very little value so I see no reason why anyone would want to
  + I am having second thoughts about this idea – primarily because the association name offers good documentation about the meaning of the association.
  + The association name should be a verb phrase that is customized for each role’s perspective.
  + There may be some cases where the role name for two different associations is the same e.g. contains/place; born-in/place
  + Perhaps binary associations could be displayed in the form: instance-of/type: Person, Character

# Sun 14-Aug-11

* Roles need different names depending on scope just like associations do.
  + E.g. In parenthood association: parent would be called "parent" from the perspective of the child role, but "conceived with" or something from the other parent's perspective. The association name might be “has parents” from the child perspective and “has children” from the parent perspective.
* Required roles in an association need to be allowed to accept a value indicating the value is unknown. This could be achieved in topic map representation by a dedicated topic for the "unknown" value. This topic should not exist in the ZADM. Thus when a role is played by "unknown" it should not show up as a link but as a special value, perhaps represented by the plaintext "<unknown>".
  + Literal properties that are required should have an analogous special <unknown> value that is a reserved keyword with special meaning that is allowed in properties of any data type.
  + The <unknown> value should be usable anywhere, even for optional properties to indicate that the property should have a value, but the value is unknown. In this way, the distinction between knowledge of absence and absence of knowledge can be kept clear.
  + Some properties, such as required names may not permit the <unknown> value.
  + Properties that are required by the semantics of a represented concept should automatically be assigned the value <unknown> if a user does not fill it in when creating the item.

# Tue 16-Aug-11

* The prefix “http://psi.topicmap.zutha.net/” should be used to identify topics which will appear in the standardized topic map implementation of the ZADM but not in the ZADM itself.
  + For example, there will be a need for an association type to represent permissions on topic map constructs. This association type topic could have identifier “http://psi.topicmap.zutha.net/has-permission”.
  + When hosts pass data to each other in topic map format, they would represent permission information about constructs using associations of type “http://psi.topicmap.zutha.net/has-permission”.
* I’m going to define the term property to be, in the context of the ZADM, a parent term for occurrences, subject indicators, subject locators and associations associated with a particular item
* When an item type defines a property, it should specify whether it may take on the <unknown> value.
  + The min and max cardinality bounds encompass the logic of whether the property is required or not (i.e. is min>0)
  + If the boolean property “can-be-unknown” of the property definition association is true, then the <unknown> value is allowed to be used to fill a required value
* If an item type has a required plays-role constraint (card-min>0) which allows the <unknown> value (can-be-unknown=true), then an association of the specified type should be created with any required roles (that can be unknown) being filled with the <unknown> value.
  + E.g. the required conception association
    - A person item should have constraint:
      * plays-role(role:child, association:conception, min-card:1, max-card:1, can-be-unknown:true)
    - The Conception association should have constraints:
      * has-role(role:child, min-card:1,max-card:1,can-be-unknown=true)
      * has-role(role:biological-mother, min-card:1,max-card:1,can-be-unknown=true)
      * has-role(role:biological-father, min-card:1,max-card:1,can-be-unknown=true)
    - If a person item $x is created whose parents are unknown, an association of type conception should be created with roles:
      * Conception(child:$x, biological-mother:<unknown>, biological-father:<unknown>)
    - Now let’s say it is independently specified via the creation of a new conception association that a male person item $y was the biological father of $x. i.e.
      * Conception(child:$x, biological-mother:<unknown>, biological-father:$y)
    - This new association would not be allowed because $x can only play the role child once in an association of type conception. However, this conflict should automatically be resolved in this case because the only differences between the conflicting associations are the replacement of unknown with known values.
    - When a user attempts to create or modify an association such that it conflicts with the constraints of one of its players, the processor should attempt to resolve the conflict by merging associations that differ only by the replacement <unknown> with a known value.
* How would complex parenthood relationships involving adoption and artificial insemination be modelled?
* How would one model a person who had their sex changed?
  + One way to model people of different sexes would be to have two disjoint subtypes of Person: Man and Woman
  + For a person that changes their sex, should they be assigned both the Man and Woman types? No. They are not both a man and a woman now, but used to be one and is now the other.
  + I don’t know of any type system that allows objects to have types scoped by time period, but that is essentially what’s needed to model this scenario
  + Once you introduce the idea of time scoped associations though, that concept needs to spread throughout the entire type system
  + If an association defines a role of type Woman it can only be played be an object that was a woman for the entire period that the association is scoped to
* If the Zutha knowledge base was to support time scoped properties, this would be in addition to the framework that keeps track of edit history
  + Some kinds of history that I have been considering would be managed by the edit history framework, such as item worth history, would instead be modelled by time-period scoped properties.

# Wed 17-Aug-11

* I don’t think I will include name types as a part of the ZADM
  + I find the concept of names having different types somewhat confusing and I can’t think of a use case for it.
  + I want the concept of a name to be simple and universal
  + Every item should have a name in the unconstrained scope and optionally additional names in other scopes
  + I see no reason to have multiple name namespace types (i.e. both scope namespaces and type namespaces)
* I think I will use the term “property” to cover the topic map concepts of “occurrence”, “subject identifier”, “subject locator” and “name”
  + Occurrence will not be an immediately understandable term for people who have not studied topic maps
  + The terms “subject identifier” and “subject locator” are somewhat more intuitive than occurrence in their meaning, but are still very topic map specific terms
  + The concept of property is already a widely understood idea and I think it makes sense for it to cover all kinds of literal data associated with an item
  + Some property types will specify that they should be implemented as specific topic map constructs in the topic map implementation of the ZADM
    - The property type “name” should be implemented as a Name of the default type in topic map form
    - The property type “External Identifier” should be implemented as a Subject Identifier in topic map form
      * This property will allow no items in its scope
    - The property type “ZID” should be implemented as a Subject Identifier with prefix <http://zutha.net/item/> in topic map form
      * This property will allow no items in its scope
    - The property type “identifier” should be implemented as a Subject Identifier with prefix <http://psi.zutha.net/> in topic map form
      * This property will allow no items in its scope
    - The property type “URI” should be implemented as a Subject Locator in topic map form
      * This property will allow no items in its scope
* I will use the term “field” as an umbrella term covering properties and associations
* I have already elected to replace the topic map term “topic” with the term “item”
* The “identifier” property type (or ZSI) is primarily intended for use by software to refer to items with a permanent and semi-human-readable identifier
  + For example, I will use the identifier “item-type” to refer to the Item Type item in the code that implements the logic of the zutha.net application.
* There will need to be a standardised universal syntax for transferring Zuthanet data
  + All hosts will need to send data to clients in the same syntax in order for clients to compare their responses by hash value
  + The syntax cannot be topic map syntax because there are some substantial differences between the ZADM and topic map data model that can be translated into topic map form but not elegantly
    - For example, in ZADM, subject identifiers will be standard properties that have permission data associated with them. In topic map form, they will need to be represented as occurrences so that they can have a reifier that contains the permissions data
* Zuthanet data should be able to be translated into topic map form but every construct need not be one-to-one mappable with a topic map construct
  + ZIDs, Identifiers and External Identifiers, for example, should all be standard item properties in the ZADM.
    - When translating these constructs to topic map form, they should be mapped to both an occurrence and a subject identifier.
    - The occurrence will have the type of the respective Item Type (ZID/Identifier/External Identifier) and will have a reifier storing the permissions data
    - The accompanying subject identifier will essentially duplicate the value of the occurrence, though possibly with a special prefix prepended such as <http://zutha.net/item/> for ZIDs
    - Actually ZIDs need not be duplicated as occurrences because they will never have item-specific permissions and so will not need a reifier in topic map form. ZIDs will also never have scope.
    - Then again, it might be convenient to store the non-prefixed form of ZIDs as an occurrence of type ZID to make it easier to lookup all ZID’s of an item. If ZIDs were only implemented as specially prefixed subject identifiers, then, in order to find the ZID’s of an item, all its subject identifiers would have to be retrieved and filtered to those starting with the ZID prefix.
    - I think the Zutha standard for topic map implementations should specify that when Zuthanet data is exported in topic map form, ZIDs should appear as both subject identifiers and occurrences of type ZID. Implementers can do as they please internally, but they must generate both forms of ZID per item when exporting in topic map syntax.
  + The URI property should similarly be represented as both an occurrence and a subject locator in topic map form
  + The Name property should be represented as a name of the default type in topic map form
    - There is no reason to duplicate the name property as an occurrence of type Name because names in topic maps have a superset of the features of occurrences, including reification and scope.
  + Schema Associations in the ZADM should be translated into topic map form as both an association of the ZADM type as well as the analogous TMCL constructs.
    - There will probably be a need to devise custom TMQL-defined constraints in order to capture all the logic of the ZADM schema
    - Such custom TMCL constraints will be defined in a set of Items dedicated to describing the standardized rules for representing the ZADM in topic map form

# Thu 18-Aug-11

* There needs to be a way to refer to properties, associations and roles directly in Zutha requests
  + I think I need to reify all constructs and give them all a ZID
  + When a Zutha client sends a modification request, it must refer to the construct it wants to modify by its ZID
  + Some constructs, like date created, date modified and ZIDs, will not have reifiers, because they are auto-generated properties that have no associated permission level
  + The ZADM should model reified constructs just as topic maps do: as a link from the construct to a reifying item
  + Reifying items should be instances of the type of the construct they reify.
    - E.g. an item that reifies a property of type “Date of Birth”, will be an instance of the Property Type Item “Date of Birth”
  + Reifying Items will also be instances of Item like everything else (by virtue of the fact that all Item Types, Property Types, Association Types and Role Types are kinds of items)
    - They will have auto-generated, non-reified names of the form “property-0D5FG” or “role-07F3J”
    - They will have auto-generated, non-reified descriptions of the form “Date of Birth property of item Albert Einstein” or “A role of type Author, played by item Robert Jordan in association 0K76E”
    - The name property may be, and the description property should be dynamically generated when requested because it must look up the current names of the items mentioned.
    - Reifying Items will have auto-generated created and modified date properties like every other item
  + Reifying Items will not actually contain the value, scope or players data stored in the construct they reify, but Zutha Clients may choose to provide an interface to this data from the perspective of the reifying item.
  + The primary purpose of Reifying Items will be for managing permissions on modification of the construct they reify and for resolving disputes about such modification
    - A construct reifier may define a permission level and optionally a reputation domain
    - If the reputation domain is present, a user must have at least as much reputation in that domain as the value in the permission level field in order to modify the construct (or the permission level itself)
    - If the reputation domain is absent, a user must have the required reputation in the global reputation domain
      * Actually perhaps the reputation domain field should be required, but may be filled by a special Global Reputation Domain value
    - If the permission data is missing from a reifier, then the permission rules are inherited from specifications in the Item Type of the parent item of the construct which enables the parent item to contain this construct. Example:
      * The Item Type item declares that its instances may have a “item-property-constraint” or “Has Property” field (an association)
      * It also specifies that a permission level of 1,000 in the Zutha Administration domain is required by default to create or modify fields of that type in its instances
      * The Person Item Type has a “Has Property” field declaring that its instances may have a “Date of Birth” property
      * This particular “Has Property” field has a reifier which is empty; that is, it does not override the default permission rules specified in the Item Type item for fields of type “Has Property”
      * This means that, in order for a user to delete or modify the cardinality of this “Has Property” field of Person (that allows people to have birth-dates), the user must have at least 1,000 reputation in the Zutha Administration domain
    - If default permission rules are missing from the declaration (in an item type) of a field, then default permission rules are taken from the field type item
    - Reifying items should allow comments to be attached to them to allow discussion about the value its reified construct should take
    - Any user can create a Conflict Resolution item and attach it to a construct reifier
      * A Conflict Resolution item will be a Decision Market which allows users to invest Zuth up to their reputation level in the Reputation Domain applicable to the Decision Market
      * Users may add candidates to the Decision Market at any time.
        + The candidates allowed will be constrained by the type of value the Decision Market is trying to decide upon.
        + For example, a decision market to decide the date of birth of Person will only allow candidates of type Date (the data type of the Date of Birth property type)
      * The Conflict Resolution item will only become “active” once one of its candidates accrues as much value (in invested zuth) as the permission level required to modify the construct in question
      * Once a Conflict Resolution item becomes active, the construct it is about may not be modified by any user until the Conflict Resolution process is complete
        + Actually, I think the construct should take on a special <in conflict> value until a concrete value is decided upon by the decision market
      * Once a Conflict Resolution item becomes active it has a fixed period of time before it completes, during which time investment in the decision market continues.
        + When the Conflict Resolution item completes its countdown, the candidate with the highest value wins and determines the new value of the applicable construct
        + The new permission level of the construct is the closing value of the winning candidate
        + The countdown period is defined with the definition of the field by an Item Type item of the parent item of the field construct; if not there, by the Field Type item defining the type of the field construct.
    - I just realised that it probably doesn’t make sense for the relevant reputation domain of an item to be specified manually for the permission rules
      * The Reputation domain as applicable to permission rules should be inherited from the reputation domain of the parent item (in the case of properties) and - ah… possibly something more complex in the case of associations. I’ll get back to that later.
      * The only data that needs to be stored by a construct reifier then is the permission level - a simple integer property.
  + I need a FieldSet item type to reify the collection of fields of a particular type in an item
    - This reifier is needed to specify item-specific permissions on modifying a field set (including adding new fields to it)
    - For example, consider the Person item “George Clooney”
      * Imagine that a meme has been spreading which is causing lots of users to add themselves to George Clooney’s list of spouses
      * The default permission level on the field type “Has Spouse”, defined in the Person item type, is not sufficient to stop the tidal wave of fake George Clooney wives
      * There needs to be a way for a high reputation user to increase the permission level required to add “Has Spouse” fields to George Clooney and only George Clooney.
      * Topic maps have no way of reifying a set of associations of a particular type involving a specific item, but they could emulate such a reifier with an association between the AssociationSet reifier, the target item and the Association type.
      * This AssociationSet reifier would have the same essential features as other reifiers in terms of providing fields for specifying permission level and allowing comments and conflict resolution/decision market items.
    - The FieldSet item would have two subtypes: AssociationSet and PropertySet
      * PropertySet would be just like the AssociationSet described above, except for a set of properties of a particular type instead of associations
* Item Types must be both instances of Item and descendants of Item
  + In fact, the Item Type Item Type must be both an instance of Item and a kind of Item
  + Damn, but bizarre things happen at the top of a type hierarchy
* Roles in optional binary associations should not be reified; to change such a field, the whole association should be removed and a new one created with new role players.

# Sun 21-Aug-11

* Association Set items should apply to all associations that involve a particular target item playing a specific role
  + i.e. the parenthood associations connected to item X with X playing the parent role should be in a different Association Set than the parenthood associations connected to X with X playing the child role.
* Some constraints (defined in item types) should be overrideable by subtypes
  + The overriding constraint should be stricter than the constraint that it overrides
  + reifier-constraint example:
    - Reifiable-Construct specifies that its instances may-have-reifier()
    - Association overrides that constraint with the refined constraint that its instances must-have-reifier()
  + item-role-constraint example
    - Item-Type specifies that its instances play the role constrained-item-type in association item-property-constraint 0 to infinity times
    - Role-Type overrides that constraint with the stricter constraint that its instances play the role constrained-item-type in association item-property-constraint exactly 0 times.
    - Property-Type and Field-Set-Type would do the same
    - This is basically saying that Item Types by default can declare that their instances have properties, but Role Types, Property Types and Field Set Types cannot declare new properties for their instances.

# Mon 22-Aug-11

* If a subtype is only allowed to refine a constraint in a supertype (that is, make it stricter) then there is no need for an explicit indication that the subtype declaration is overriding anything.
  + The rule simply needs to be that all the constraints must hold, in which case the stricter constraint in the subtype is the effective constraint. The weaker rule in the supertype still holds though.
* There will need to some need for override logic actually – when it is declared that a type must have a certain field exactly 1 time and multiple types in the inheritance chain have that field, then the field that applies to instances is the one in the nearest type to the instance.
  + auto-property-generation-rule example:
    - auto-property-type declares the property auto-property-generation-rule for its instances
    - auto-property itself does not define the field property auto-property-generation-rule – it doesn’t have to because it is abstract
    - auto-name defines property auto-property-generation-rule as “\”item-\” + zid”
    - auto-property-name defines property auto-property-generation-rule as “\”property-\” + zid”
    - The declaration in auto-property-type: “has-property(zsi:auto-property-generation-rule,1,1);” applies to the whole inheritance tree of each of its instances.
      * To satisfy the card-min=1 condition: an instance of auto-property-type must either have the property auto-property-generation-rule or have a supertype which does have that property
      * To satisfy the card-max=1 condition: every definition of auto-property-generation-rule in an auto-property-type instance overrides any definition of this property in a supertype of that instance.
    - So the definition of auto-property-type in auto-property-name overrides the definition in auto-name.
* How does automatic override detection handle declarations in item types that allow 2 or more fields of a type in its instances?
  + I think an explicit override declaration would be required to handle this situation
  + Let’s say item type $itemTypeX declares that its instances may have from 0 to 2 properties of type $propType
  + $itemTypeA isa $itemTypeX
  + $itemTypeA defines a property $prop1 of type $propType
  + $itemTypeB ako $itemTypeA (inheritance relationship)
  + $itemTypeB defines a property $prop2 of type $propType
  + Is $itemTypeB’s definition of $prop2 overriding $prop1 or defining a second occurrence of $propType?
  + In order to resolve this ambiguity, $itemTypeB needs to be forced to declare explicitly that $prop2 overrides $prop1 if that is the desired behaviour. Otherwise, $prop2 will be assumed to be the second definition of $propType allowed by $itemTypeX
* In situations where it can be inferred that a field definition overrides a definition in a supertype (such as when the max cardinality of the field is 1), I think an explicit overrides association should still be created between the reifiers of the two fields, even if this association is created automatically.
* Should refinement of constraints require explicit override declarations?
  + reifier-constraint example
    - reifiable-construct-type specifies for its instances: “plays-role(zsi:constrained-reifiable-construct-type,zsi:reifier-constraint,1,1);”
    - reifiable-construct implements this required field by specifying: “may-have-reifier();”
    - auto-property also implements this required field by specifying: “cannot-have-reifier();”
    - unless auto-property’s implementation of this field overrides the implementation in reifiable construct, auto-property has effectively implemented it twice because it inherits reifiable-construct’s implementation, yet reifiable-construct-type specifies that it must only be implemented once.
    - In this scenario, I think an explicit override declaration should be used, even though the constraint logic would work even if both reifier-constraint fields were considered active.
  + Item-property-constraint example
    - Item-type implements item-role-constraint with “plays-role(zsi:constrained-item-type, zsi:item-property-constraint,0,\*);”
    - role-type implements item-role-constraint with “plays-role(zsi:constrained-item-type, zsi:item-property-constraint,0,0);”
    - item-role-constraint is allowed to be implemented any number of times so the presence of both of these implementations in role-type is not a violation of the constraint rules
    - The desired logic for item-property-constraint in role-type also works as desired when both implementations of it are considered valid in role-type
    - It is thus optional whether role-type’s direct implementation of item-property-constraint overrides its inherited implementation of it from item-type
* When $itemTypeX declares that its instances have a property of type $propTypeA, this declaration does not imply that instances of $itemTypeX may have properties of type $propTypeB, where $propTypeB is a subtype of $propTypeA.
  + If a subtype of $itemTypeX, say $itemTypeY declares that its instances may have properties of type $propTypeB, then its instances may have both properties of type $propTypeA and properties of type $propTypeB
  + If an instance $itemY of $itemTypeY has a property of type $propTypeB, this is also a $propTypeA so it contributes to $itemY’s count of properties of type $propTypeA and is thus constrained by the min and max cardinality of properties of type $propTypeA specified in $itemTypeX.
  + Example with the Name property
    - item declares “has-property(zsi:name,1,\*);”
    - role declares “has-property(zsi:auto-role-name,1,1);”
    - role is a subtype of item
    - auto-role-name is a subtype of name
    - A standard item (say “Robert Jordan” the author), is not allowed to have properties of type auto-role-name or any other property which is a subtype of name
    - A role item must have a property of type auto-role-name, and this property is automatically created when the role item is created because this is the behaviour for properties whose type is a type of auto-property.
    - Since auto-role-name is a subtype of name, this automatically generated property satisfies the constraint declared in item that all instances of item must have at least 1 name
    - A role item is also allowed to have additional name properties like any other item, but it must have exactly 1 auto-role-name.
* A Field-Set should only be attached to a field type which is declared to be allowed an optional number of times (min-card < max-card)
  + The conceptual idea of a field-set should only apply in these cases
  + A field-set should be required to reify/be reified. This just means that all situations where the conceptual field-set is relevant, an actual field-set item should be associated.
  + The permission-level value associated with a field-set determines the reputation required to add an element to the field-set and gives a lower bound for the reputation required to delete an element from the field-set.
  + This is essentially the only purpose of the field-set construct – thus it makes no sense for a field-set to be associated with a field-type that is only allowed a precise number of times by the schema.
* The is-abstract declaration is not inherited, but most other schema declarations in item-types are inherited
  + Schema-rule associations should have a Boolean property is-inherited that defines whether they apply to all subtypes of the declaring Item Type or just the Item Type itself.
  + Actually I think inheritance behaviour of constraints should be defined (at least for now) as part of the hard-coded meaning of the constraint. This is because it will probably be difficult to design a system for specifying define which role (or roles) of the constraint association the is-inherited condition applies to.

# Sat 27-Aug-11

* An AssociationFieldSet is a set of Associations of a particular type with the parent item of the AssociationFieldSet playing a specific role.
  + An AssociationFieldSet is reified
  + It contains, conceptually, a set of AssociationFields, which are Associations as seen from the perspective of a particular item playing a particular role
  + A single item may play multiple roles in a single association, in which case that item will have an AssociationField for each role it plays
  + An item has an AssociationFieldSet for every role type it plays in every Association type in may participate in
  + There is a conceptual AssociationField for every role-player in the database
  + AssociationFields are not reified
* I don’t think Roles should be reified
  + There are two main reasons why it might make sense to reify role items
    1. To allow assignment of role-specific permission-levels
    2. To refer programmatically to specific roles in an association via a ZID
  + I don’t think it makes sense for individual role-players in an association to have their own permission level.
    - Changing any role in an association fundamentally affects all the players in that association.
    - The association as a whole represents an atomic piece of information; changing any part of it changes what the whole association means
    - A higher permission on a single role in an association would make it harder to remove that role, but changing any other role in the association still changes the meaning of the high permission role
  + A role can be referred to precisely by the triangulation of an Association, an Item and a Role Type
    - This is assuming that an Item is not allowed to play the same Role Type more than once in a single Association.
    - I don’t think it could ever make sense to do this so such a scenario can safely be banned.
    - If a role is defined within the ZADM to be the triangulation of those three constructs (Association, Item, Role Type) then it would never be possible to assign an Item to the same Role Type twice in a single Association; the only action available would be to replace the existing role defined by those three constructs.
    - The most common way to add and subtract players of Associations through the API would be to make commands such as:
      * associationX.removePlayer(itemY) --this would remove all roles from associationX where itemY is the player
      * associationX.removePlayer(itemY,roleTypeZ)
      * associationX.addPlayer(itemY,roleTypeZ)
  + An association will have a single permission level associated with it which applies to adding or removing roles from the association, deleting the association itself, or modifying the association’s permission level.
* Roles should behave somewhat like traits in Scala
  + An ItemType can declare that its instances are allowed to play a certain role. For example, the Person ItemType may declare that its instances are allowed to play the Role Author.
  + Specific instances can then declare whether or not they do play any given Role that their ItemType declares they are allowed to play. The J.K. Rowling Item would, for example, declare that it plays the Author Role.
  + Role Type items (such as Author) declare the Associations they may participate in (via the same constraint associations that declare which roles an Association has). For example, the Author Role Type might declare that it participates in the “Works Written”, “Works Edited”, “Series Written”, “Notable Works Written”, “Writes in Genre” and “Member of School or Movement” associations.
  + Role Type items may also declare Properties that their players may have. For example the Author Role Type might declare the properties: “Open Library ID” and “Pen Name”
  + This behaviour of Roles is similar to Scala’s traits (and traits and mixins in other languages) because items “mixin” the properties (and associations) defined in a Role without actually subtyping the Role. The “plays-role” association between an Item and a Role Type is similar to the Mixin relationship between an object or class and a trait in Scala.
  + An Item Type may also be a Role Type
    - For example Person is an Item Type because it can be instantiated, but Person is also a Role – in “has occupation”, “place of birth”, “has nationality” say; and Person has properties like “age”, “height”, “date of birth” etc.
  + An Item Type that is not also a Role Type should only be able to define properties and specify the Roles that it is allowed to play
* The item called “Role” should be the Item Type which Role items like “Author” instantiate
  + There will no longer be both a Role and a Role Type item
  + An Author is a Role, it is not a kind of Role
  + I think the reason Topic Map terminology distinguishes between Roles and Role Types is so that the work Role can be used to describe the actual programmatic construct that links a player, a role type and an association.
  + Actually perhaps this is a valid way of defining Role. After all, we say things like “Bary’s role in the case was as a witness”. The fact that the ownership of the “role” is attributed to someone and the use of “is a” before the word “witness” makes witness look like a type of something – a type of Role. So Bary’s Role as a witness in the case is a specific instance of the Role Type “witness”.
  + OK, I’ve changed my mind. Role Type gets to stay.
* I have now strayed so far from the structure of Topic Map Constraint Language that I don’t think there is any point trying to figure out how the ZADM constraint system might be implemented in it. If it can be – good; if not – I don’t think it will be such a big deal given that I no longer intend Topic Map syntax to be the standard syntax of Zutha data.
  + Some people may want to export Zutha semantic data in Topic Map syntax just as they some people may want to export it in RDF format. Both should be possible because Zutha data will be well structured data.
* The association-role-constraint should now specify the cardinality of both players of the relationship (the association type and the role type)
  + association-card-min will specify the minimum number of associations of the constrained-association-type the constrained-role-type requires its players to play.
  + association-card-max will specify the maximum number of associations of the constrained-association-type the constrained-role-type allows its players to play.
  + role-card-min will specify the minimum number of roles of the constrained-role-type that must be present in associations of the constrained-association-type
  + role-card-max will specify the maximum number of roles of the constrained-role-type that may be present in associations of the constrained-association-type
  + Example: the Authorship association
    - The association-role-constraint between the Authorship association and the Author role would define the following cardinality values:
      * Association-card-min: 0
      * Association-card-max: \*
      * Role-card-min: 1
      * Role-card-max: \*
    - This means that Authors may be the Author of zero or more books and that an Authorship association includes 1 or more Authors
    - The association-role-constraint between the Authorship association and the Written-Work role would define the following cardinality values:
      * Association-card-min: 1
      * Association-card-max: 1
      * Role-card-min: 1
      * Role-card-max: 1
    - This means that a Written-Work must be in an association of type Authorship exactly once and that an Authorship association applies to exactly one Written-Work
* A Role Type items can be declared abstract
  + Instead of disallowing instantiation of this item the way is-abstract does with Item Types, the is-abstract declaration on a Role means it cannot be played directly. Only subtypes of the Role can be played.
  + I can’t think of a Use Case, though
* If an item X is an instance of both Item Type and Role Type, its instances are implicitly declared to play the role X
* I’m leaning towards renaming “Role Type” to “Role” again.
  + My convention has been to end item names with “Type” if their instances can be instantiated
  + The items that will be instances of Role Type/Role, such as Author, CEO, Parent etc. are thought of as being “Roles” no “Role Types”
  + There needs to be an Item called Role because Role is a concept that people use to categorize certain kinds of concepts (like Author, Teacher, Parent)
  + The current meaning of the Role Type item is pretty much exactly the same as what most people understand the meaning of the concept Role to be
  + I won’t use the terminology “Role” the way it is used in topic map terminology to refer to the construct that connects a Role Player, Role Type and Association.
  + Associations will be considered to have Role Players each of which plays one or more Role in the association (rarely more than 1).
  + In Zutha terminology, an Association’s Roles will mean the same thing as an Association’s Role Types in Topic Map terminology.

# Sun 28-Aug-11

* The zsi:item-plays-role association is not inherited
  + It applies to an item, not to an item type, so it is not applicable to the item’s subtypes (which it may only have if the item also happens to be an item type)
  + This is similar to the behaviour of the type-instance association.
* If an Item Type $type declares must-play-role($role) then the zsi:item-plays-role association between $type’s instances and $role is implicit and doesn’t actually have to exist
  + Or maybe this association should just be created automatically on item creation
  + If the zsi:item-plays-role association didn’t exist, this would be appropriate for the behaviour of permissions, because it would not be possible to remove this required association. Even if it did exist though, the requirements of the schema override the editing privileges of any user or users.
  + I think it would be more appropriate if the required zsi:item-plays-role association is actually created because it would be more consistent. It would be strange if a query for all Roles played by an item returned Roles that couldn’t be found by browsing the relevant associations manually.
  + Use Case: Item declares must-play-role(zsi:instance)
    - Whenever an item is created, an zsi:item-plays-role association is automatically created between it and the zsi:instance role
  + Use Case: Item Type declares must-play-role(zsi:supertype)
    - The zsi:item-plays-role(zsi:supertype) association will automatically be created for every new Item Type
    - This does not necessarily mean that all Item Types must participate in the supertype-subtype association as a supertype (i.e. have subtypes), only that zsi:supertype must appear in their list of Roles Played
* The zsi:item-type-is-abstract-constraint prevents an item from playing the role zsi:type
  + Item Type should declare must-play-role(zsi:type)
  + The zsi:item-plays-role(zsi:type) association will automatically be created for every new Item Type, except those that also declare is-abstract()
* Users should not be able to instantiate Item Type directly; they must subtype an existing Item Type
  + When an Item Type X is subtyped, a kind of Item Type K is automatically assigned as the type of the new subtype S as specified by a zsi:descendents-have-type-constraint between X and K
  + Examples
    - subtypes of Association will always be of type Association Type
    - subtypes of Property will always be of type Property Type
    - subtypes of System Property will always be of type System Property Type
  + The zsi:descendents-have-type-constraint association is inherited
  + Example: Hybrid Property does not have its own zsi:descendents-have-type-constraint so it inherits the one from Property meaning its instances must be of type Property Type
* I think there is need for a concept like a java interface – that is, something that can be subtyped but not directly instantiated
  + A Role might be like a Scala trait in some ways, but it isn’t really the same thing as a trait/interface
  + I think I’ll call this concept a Trait, because I think that term will be more meaningful to the average user
  + Traits should be able to declare roles and properties just like Item Types can
  + Use case: “mortal” trait
    - The trait “mortal” declares the roles “killed” and “murdered” and the properties “date of death” and “age”
    - Human and Hobbit both subtype the trait mortal
* I just realised that there are Roles that are played by some items in only a subset of the associations in which they appear
  + For example the role “victim” could be played by a mortal in the association “murderer-murders-victim”, but a mortal wouldn’t play the role “victim” in an association like “victim of hoax”
  + Roles also might have different cardinalities for different players. For example, a phoenix could play the role “died” or “killed” more than once, but a normal mortal could only play that role at most once.
  + I think the first problem could be avoided by requiring a Role to be defined more specifically than Victim is. i.e. the victim of a murder should be a distinct role to the victim of a hoax.
  + The second problem seems more intractable
* The reason I wanted to define Roles as packages of allowed associations is in order to do the equivalent of declaring a Person as able to play the Role Author and letting the Author Role encapsulate the logic of what Authors do.
* Perhaps the concept of Traits can replace Roles as the packages of allowed associations and properties
  + Author could be both a Trait and a Role. As a Trait, it would declare all the associations an Author can participate in along with the cardinality of those associations (the number of associations of a particular type allowed by a single Author). This is just like the plays-role() constraint used by Topic Types in TMCL.
  + Item Types should also go back to declaring plays-role() constraints the old way (TMCL style)
  + An Item Type should be able to subtype a Trait in which case all its instances are “instances” of the Trait as well
    - Use case: the Item Type Human is a subtype of the Trait Mortal
  + A Trait cannot be an Item Type’s only supertype
    - In this way, an “instance” of a Trait cannot be created unless the instance also has the type of an Item Type that is compatible with that Trait
  + An Item Type should be able to declare Traits that its instances may take on
    - Use case: the Item Type “Person” may have Trait “Author”; Robert Jordan is an instance of both Author and Robert Jordan
  + Actually I think Items should “Mix In” Traits rather than instantiate them
    - The whole point of Traits is that they cannot be instantiated.
    - It would be inelegant (and difficult to implement I think) to allow Traits to be instantiated only conditionally on the instance also having a compatible Item Type as a type.
    - Should the schema specify that Trait is allowed to play the role zsi:type or not?
    - A zsi:item-mixes-in-trait association should be treated the same way as type-instance for the purpose of identifying the types of an Item
    - In order to implement this in Topic Maps (so that inbuilt transitive logic can be used), wherever a zsi:item-mixes-in-trait association appears, the Item should instantiate an anonymous item that subtypes the Trait(s)
* The Zutha Data Model (ZDM) is to the Topic Map Data Model (TMDM) as Scala is to Java
  + ZDM has scala-like traits; TMDM only has standard inheritance with abstract types
  + ZDM items can mixin traits the way scala objects can mix in Traits directly.
    - In scala (I think) this feature is mapped to Java by creating an anonymous class that implements the corresponding interfaces for each mixed-in trait; the object then instantiates this anonymous class
    - When ZDM is mapped to TMDM, Items that mix in Traits directly (the way Robert Jordan mixes in the Author trait) should be transformed to instantiate an anonymous Item Type that subtypes the Traits (and the original Item Type) of the Item.

# Mon 29-Aug-11

* The Type-Instance association cannot be reified because it is used by reified constructs
  + At a minimum it cannot be required to be reified because that would cause an infinite loop
  + I think the concept of Traits will make it possible to avoid reifying Type-Instance associations because it will be possible to enforce that every item only has one type (via the Type-Instance association). Additional Traits can be mixed in to achieve the effect of multiple inheritance without reusing the Type-Instance association
  + In this way, the Type-Instance association can be made immutable; once an Item of a particular type is created, its type cannot be changed.
* I think it might be possible to merge the concept of Trait into Role after all
  + To “play” Role would be analogous to “mixing in” a Trait
  + An item could be an Item Type as well as a Role by instantiating Item Type and playing “Role”
  + Role would be an Item Type and play the Role “Role”
    - This would be the strangest relationship of all because Role is only allowed to be “played” (or mixed in in Trait terminology) if it *is* a Role and it is only an (effective) Instance of Role after it Plays (Mixes In) Role.
  + An instance of Role would have the same behaviour as instances of Trait described above
    - A Role can be played but not instantiated
    - A Role can be subtyped by an Item Type but the Item Type must also be a subtype of another Item Type
      * This behaviour is enforced by the zsi:item-descendants-have-type-constraint(zsi:item: zsi:item, zsi:item-type: zsi:item-type) association
        + This constraint also implicitly requires all instances of Item Type to be descendants of Item via the ZDM specification level ban on directly instantiating an Item Type.
        + New Item Types may only be created by subtyping an Item that is an instance of Item Type.
        + When an Item Type X is subtyped with Y, the required type of Y is inferred from nearest ancestor of X (including X) that specifies the zsi:item-descendants-have-type-constraint
      * If only this constraint is relied upon, Item itself would still be allowed to subtype a Role
        + Since this is a special case and Item will be (effectively) sealed for modification once the schema is finalized (via an extremely high permission level) this shouldn’t be a problem
  + If an Item is an instance of a Role it implicitly plays that Role
    - Example: Person is a kind of Mortal (a Role) which is a kind of Killable (a Role). Hitler is an instance of Person (and, by inheritance, is an instance of Killable), so Hitler implicitly Plays the Role Killable. Thus Hitler can participate in the killed-by association as the Killable. Hitler would have to be manually given the Killer Role in order to play that Role in the killed-by association though.
      * This example is causing me to consider keeping Traits because Mortal is more appropriately a Trait than a Role; it should declare that it plays Roles such as Killed and Murder Victim.
    - Example: Name (an instance of Property Type which subtypes Role and plays the Role Role) is a Property Type and so Name implicitly plays the Role Property Type. Thus Name is allowed to participate in the property-declaration association as the property-type.
* I think I might have been confused by thinking about the Author example into thinking that a Trait and a Role is (or at least can be treated as) the same thing
  + Author is both a Trait and a Role
    - The word Author is used to associate a Trait with a Person (“Robert Jordan is an Author”). This is Author acting as a Trait.
    - The word Author is also used to describe the Role played by a Person in the Authorship association between the Person and a Written Work.
  + Murder Victim is only a Role
    - The term “murder victim” is only used to describe a person with reference to a particular context where they played that Role (the context of their murder)
    - This is distinct from the way a person can be described as an Author without particular reference to any specific context in which they played that Role
  + Mortal is a Trait
    - You say that someone is Mortal without reference to some context in which they played the Role of Mortal
    - I can’t think of any association where Mortal is a Role – it is more a description of the nature of something than a Role played by someone
    - Having the Trait Mortal carries with it some implications about the Roles that one can play, though. For example, a Mortal can play the Role “killed” (or maybe “victim” or perhaps “killed-victim”) in the killer-kills-victim association.
* How would things look if I kept the concepts of Trait and Role distinct?
  + A Trait would be the container that holds allowed associations and properties
  + A Role should just be a non-functional label for the Role played by an Item in an association
  + Roles will still be associated with the associations they are allowed to participate in, but the cardinality of the associations a Role may play should not be associated with the Role.
  + Both Item Types and Traits can declare that they play a particular Role a constrained number of times in a particular association
  + Both Item Types and Traits may declare properties
  + Role needs to be able to be instantiated as well as mixed in. This is so that Traits like Author can be instances of Trait, but mix in the Trait Role (remember we cannot have multiple types – via the type-instance association – per item).
  + Role therefore needs to be both an Item Type and a Trait
  + Role must be an instance of Item Type and mix in Trait
  + Trait must therefore be both an Item Type and a Trait (so it must mix in itself)
  + But if Trait can be mixed in… then Items can be created which are of an direct type but which can be mixed in as Traits.
    - Well Role and Trait itself are both use cases of needing such functionality.
    - A think a high permission level should be assigned to mixing in Trait because it is a strange thing to do, but it may well turn out to be a useful thing to be able to do in other cases than Trait and Role.
* I think it would be elegant if all instances of Item Type were required to have exactly one supertype (via the supertype-subtype association) in the same way that all instances of Item are required to have exactly one Type (via the type-instance association)
  + There should be an association called mixes-in which subtypes the supertype-subtype association
  + An Item Type is a subtype of exactly one supertype, but may mix-in multiple Traits as pseudo-supertypes
  + A separate item-has-trait association should subtype the type-instance association
  + An Item is an instance of exactly one type, but may have additional Traits that act as pseudo-types
  + Traits should use the mixes-in association to mix-in “supertraits”
  + Association mixes-in has role: supertrait, which subtypes supertype, and role: subtype
  + The supertype-subtype association could potentially be non-reified and therefore immutable after creation
    - I’m not sure this would be desirable though, because if there was a need to insert a new type between two existing types in the hierarchy, the immutability of the supertype-subtype association would require deleting and recreating… well, potentially the entire type system – LOL – we don’t want that!
    - I guess a special editing function could be added to the API for inserting types between two existing types in the hierarchy.
    - No, I think there needs to be the possibility to add permissions to specific supertype subtype associations and allow modification of them (provided the modification leaves the database in a valid state according to the schema)
  + If the item-has-trait association is going to subtype type-instance its roles need to subtype type and instance
  + I’m having second thoughts about subtyping the type-instance and supertype-subtype associations
    - Item-has-trait does not mean the item is an instance of the trait, it means the item has that trait
    - Similarly, mixes-in does not actually mean subtypes
    - I think the main reason I came up with the idea of subtyping supertype-subtype and type-instance was because it would be a somewhat more elegant way to get the semantics of item-has-trait and mixes-in to work in topic map software.
    - However, I’m almost certain that topic map engines like MajorToM and TMQL4J only treat associations that are direct instances of type-instance/supertype-subtype as type-instance/supertype-subtype associations
  + Item Types should still be constrained to having exactly one supertype
  + Item types should mixin traits via the item-type-mixes-in-trait association
  + In topic map form, for every item-type-mixes-in-trait association, an anonymous topic should be created that subtypes the trait and is subtyped by the item type
  + Traits should have multiple inheritance
  + Items should be able to declare traits directly via the item-has-trait association
  + In topic map form, for every item-has-trait association, an anonymous topic should be created that subtypes the trait and is instantiated by the item
* Reified constructs should have an optional permission-level-lock-level property so that the permission level can be locked to a low level, requiring a high reputation to increase it.
* non-reified field sets:
  + attribute:attribute-has-investments
* field sets do not need to be reified to be referred to in the API. Adding a property is done by specifying the type, value and parent item. Adding an association is done by specifying the type and role/player pairs. A field is removed by referring to its reifier's zid.
* Investing item structure
  + Users make investments by creating an investment item with an initial value. Thereafter the value field is controlled automatically.
  + Investment items participate in an attribute-has-investment association with an attribute item (which will in turn be associated with a single item).
  + Users make extraction requests which cause the zuth in their investments to be extracted over time and added to their account.
  + Empty investments are marked depleted but retain their association with the parent attribute item.
  + Actually all attribute items should have one positive investment pool and one negative investment pool. These will contain the investments.

# Tue 30-Aug-11

* The concept of reifying field sets is the most inelegant aspect of the schema at the moment I reckon
  + I think it would be far better if permissions on adding fields to a field set were specified by a reified association, perhaps called “field-set-permission-level”
  + This way, the field set permission can be absent (as it very commonly will be) without incurring the overhead of an obsolete reifier (which is not even really a reifier but an item associated with the field set via several anonymous associations.)
  + Reified constructs (properties and associations) can now be allowed to have field sets (such as a flexible number of description fields) because a field-set is no longer a reified construct so it will no longer cause an infinite loop as it would have if all reified constructs contained a reified construct.

# Wed 31-Aug-11

* It should be possible to mark data on the zuthanet as private while still providing it unencrypted to hosts so that they can perform calculations on it.
  + Hosts holding private data should only provide access to it to requesters who can sign with the associated signature.
  + In this way, users can host such things as text documents on the zuthanet and be able to use the services of Zutha hosts to search and perform other computations on their documents while still retaining a reasonable degree of privacy. The hosts holding their documents (which need not all be held by the same host groups) would not know the identity of the documents' owner beyond an anonymous public key. An attacker targeting a particular user would have no way of discovering which hosts are holding their documents without the key used by the selection algorithm to determine the documents'controller group. This key would be held privately by the user and be included in the privately-marked data along with the documents.
  + Only hosts with a sufficient trustworthiness reputation would be allowed to host unencrypted private data.
* Functions could be constructed from a library of primitive functions that use only Zutha items and associations as a programming language.
  + Inputs could be simple value objects; complex objects built from primitive items connected by associations; function closures, possibly partially applied with other inputs which the closure is still actively linked to.
  + Output could be a primitive that can be assigned as the value of a property such that the property can have its value specified as the result of a function which is called whenever the property is requested. The output of a function could also be a reference to an item of a particular type that is identified, or possibly constructed on each call, by the function.
* If the zuthanet is going to start being used for general purpose distributed programming, there is going to be need for garbage collection.
  + Some items should be marked to be garbage collected if they are not requested for a particular period of time.
  + Owners of Items should be allowed to delete them permanently
    - Items requested to be deleted should be kept for a period of time so that the action can potentially be reversed by a user or users with sufficient reputation
  + I’m also beginning to reconsider whether the benefits of keeping a full history of all actions performed in the Zuthanet would outweigh the costs
  + I think I’m deceiving myself about the value of keeping all information. Sure there is going to be value in keeping useful information about historic events, but keeping everything simply out of reluctance to let any information be destroyed would be foolish and impractical I think.
  + I think a lot of thought will need to go into how to identify information that is worth keeping and what can be harmlessly destroyed
  + One obvious flaw in the concept of keeping everything is that it leaves the network vulnerable to deliberate or even just unintentional clogging. Even if the network is capable of handling enormous quantities of data, I think it would always remain possible to generate junk a great deal faster than valuable content is generated.
  + So at a first assessment, what information would be worth keeping?
    - I think the full modification history of all undeleted items should be kept indefinitely
    - When an item is deleted permanently, its modification history should be deleted, but an event marking the deletion event should be kept
    - No items that are accessed commonly by users with reputation should ever be allowed to be deleted
    - Items that have been popular in the past but are now superseded by a replacement may be deleted. This is particularly applicable to large files such as videos that may be replaced by a better version.

# Tue 27-Sep-11

* The “trails” left by high reputation users in their exploration of the Zutha knowledgebase should be captured so that other users can follow in their footsteps.
  + This idea is inspired by something I just read about Hypermedia and some ideas of Vannevar Bush
  + This trail capturing concept could take the form of a ranking of outgoing links (associations) from an Item based on the frequency of their use, weighted by the reputation of each user that uses them.
  + The ranking of outgoing links could even be dependent on the prior path that has been followed by the currently browsing user. So if a user X has just followed a link from A to B, the outgoing links from B will be ranked according to the weighted (by reputation) frequency of their use by users that have just come to B from A.
  + In order to do this, detailed information about user browsing habits would need to be captured, but that conflicts with my plan to have view requests be anonymous by default so that hosts do not know whether they are serving a high reputation user or not and will therefore have an incentive to serve all requests.
* I don’t think it is realistic to expect high reputation users to conceal their identity when making view requests when it would benefit them (in terms of priority response times) to disclose their high reputation to hosts
  + Instead of relying on this to create incentives for hosts to serve low reputation users, the incentive could be based on a social expectation that low reputation (and zero reputation) users be served on a best effort basis.
  + Hosts that are found to discriminate significantly against low reputation users can be punished by high reputation users wishing to uphold that expectation for the good of the Zutha community.
  + High reputation users will of course be able to voluntarily hide their reputation when making requests in order to test hosts for discriminatory behaviour. Hosts can then be rewarded or punished accordingly by that high reputation user.
  + For hosts with excess capacity (which should be any host seeking to move up the rungs of the Zutha host ladder), it should be worth their while to serve users that don’t disclose their identity even though the expected reward from them would be lower (a request from an unidentified user is more likely to be coming from a user with low reputation and thus lower capacity to reward the host) because any reward they can get is beneficial and they have enough capacity not to need to discriminate.
  + It is likely that, for many hosts, it will not be efficient to lookup the reputation of every user that makes a request, because the reputation data will probably have to be looked up from another host anyway, which is a lot of overhead just to decide whether to process a request or not. It wouldn’t be much more work to just process the whole request. For requests that are clearly going to take a lot of processing effort, however, it would be worth the while of hosts to discriminate against low reputation users, and this would be quite acceptable; the ability to make processing-intensive requests should be a privilege of high-reputation users.
* How to capture user browsing behaviour while still permitting privacy and incentives for hosts to serve all view requests
  + High reputation users have an incentive to disclose their identity so that they can receive priority service.
  + It is primarily the high reputation users whose browsing data is desired since it has a higher likelihood of being authentic (it would take vastly more resources to deliberately fudge browsing statistics when every drone has to have a high reputation).
  + Users that want to browse anonymously may still do so though they are likely to receive somewhat lower priority service from hosts
  + Hosts can be encouraged to still provide adequate service to low reputation and anonymous users through a social expectation enforced by high reputation users (who care about the issue) that they do so. These high reputation users will browse the network anonymously or as low reputation users, and praise or punish hosts according to how well they respond to requests.
* Users should be able to create associations that they mark private which will cause them to only appear to that user (by default). The associations will be unencrypted and any user may view the private associations of others if they explicitly request it.
  + This functionality would be useful for such tasks as annotating a document for information relevant to a particular project you are working on, or simply annotating content that interests you. The annotations would not be secret, but they are not universally applicable in the way that something like an authorship association is.
* There should be a kind of item called Anchor, which has various subtypes specific to various kinds of items.
  + An Anchor would point to a particular part of another item, such as a section of text in a document or a position in a music track.
  + When an Anchor plays a role in an association, it acts as a proxy for the part of another item it points to
  + Zutha clients could show anchors in documents or music tracks or other kinds of items in the form of links or annotations or some other interactive control relevant to the kind of item, the kind of anchor and the associations the anchor plays a role in.

# Sun 2-Oct-11

* Can Bitcoin satisfy Zutha’s decentralized currency needs?
  + I don’t think it can because Zutha needs to be able to take control of the allocation of currency according to the dynamics of investment in items (and some other processes). Bitcoin fundamentally leaves control in the hands of the sending and receiving party. It doesn’t help in the pursuit of creating a decentralized intermediary with decision-making power over the allocation of currency.
  + There is no way for control of Bitcoin to be dependent on consensus; if you have the private key of a Bitcoin address (public key) that has money stored with it then you can send that money to another Bitcoin address with or without the consensus of other people that also know that private key. Those other holders of the private key will be able to detect that one of their number has used the Bitcoin associated with it, but I can’t see how this helps much.
  + Bitcoin can still be exchanged for Zutha dollars of course – via money handlers that accept Bitcoin that is.
* The Zutha Dollar will provide about the same level of potential anonymity as Bitcoin, but will probably involve much less anonymity in practice.
  + The convenience of being able to send money to a person or company by just looking up the item that represents them on Zutha and sending money to it will probably lead to a great deal of completely transparent transactions.
  + If someone wants anonymity though, they can create as many temporary Zutha identities as they like and transact using them.
  + Money launderers could have money sent to one of their dummy accounts and send it on to whomever they like without the transaction being traceable to them. The trail the money actually takes would be publically visible, so if illegal activity is detected at one endpoint of a money trail (an endpoint being a sale or purchase of Zutha dollars), all the other endpoints of that money trail can be easily found. So, for example, if a criminal’s US bank account is identified and that account has been receiving money from the Zutha network, the trail of that money can be traced back arbitrarily far. Unless the criminal’s financier has earned all his/her money on Zutha, this money trail should lead to some meaningful entry point(s), such as the purchase of Zutha Dollars from a bank account (that can then be investigated by traditional anti-money laundering means) or to some Zutha user with an identity that can be traced to the real world.
  + Bitcoin allows money trails to be traced as well, but the information which is in the public domain does not permit the identification of transactions which are trades of Bitcoin for other currencies. This information will be available on Zutha (most of the time) because users buy or sell Zutha dollars by making public trade requests which are then randomly assigned to a money handler to process. A trade request can only be made by a user who has associated with his user account, the details of some account to which money can be sent by money handlers, or which can be connected to the user when he uses it to send money to a money handler. Such an account might include a bank account number and routing number, PayPal account, Bitcoin address etc.
  + Trades could be arranged that involve Zutha Dollars which do not involve a public trade request on the Zutha network. For example, user A could communicate directly with user B and arrange to send him 1 million Zutha Dollars if user B gives him $1 million cash in person. Such trades are already possible with traditional financial institutions however and their effectiveness for money laundering depend on the cleanness of user A’s reputation.
  + One of the benefits of Bitcoin to money launderers is that it allows tainted trails of money to be cleaned by purchasing Bitcoin at a Bitcoin Exchange. The money trail can be traced to the Bitcoin exchange, but the exchange then needs to be coerced into providing the Bitcoin addresses to which they sent Bitcoin in exchange for the money before the trail can be traced into the Bitcoin network. I suspect that regulations about such disclosure by Bitcoin Exchanges are ambiguous at the moment which means there will probably arise a proliferation of semi-legitimate institutions with enough trust that traditional financial institutions will (and will be permitted be government to) approve transactions with them, but who also have an implicit arrangement with money launderers to destroy the data linking transactions with Bitcoin addresses. I’d say regulation of Bitcoin Exchanges would be the most important thing for governments to look at if they want to prevent money laundering over the Bitcoin network.
  + It will always be possible for a criminal to trade currencies (such as USD for Bitcoin or USD for Zutha Dollar) directly with an individual, but then the interaction must be personal and that individual becomes complicit in the money laundering process. This kind of intimate trade has to be dealt with by whatever means such trades are currently dealt with by existing anti-money-laundering strategies. The problem entities like Bitcoin Exchanges and potentially the Zutha network introduce is the provision of an impersonal agency that money launderers can use to clean their money. Criminals don’t need to go through the difficult process of cultivating personal relationships with reputable entities who are willing to turn a blind eye to law-breaking – instead, they can just purchase anonymous currency (Bitcoin or Zutha Dollars) at a public exchange, which (unless measures are taken to prevent it) may then destroy any record of the transaction.
  + What measures can Zutha take to inhibit money launderers
    - Record the details of every trade of Zutha Dollars for other currencies that goes through the standard public currency trading system. This system will be robust because the money handler that processes each request will be selected at random, so it will not be possible for a corrupt money handler and money launderer to arrange a transaction with each other through this system whereby the money handler falsifies the record to conceal the true source of the laundered money.
    - One service that I think would be valuable, and which I think would likely arise naturally, would be an automated auction-style currency trading system that allows buyers and sellers to find each other, but allows the trade itself to happen externally. Such a system would be a boon for money launderers because it would allow them to get lots of innocent suckers to launder they money for them. Thus it will be important to try to prevent innocent people from getting sucked into such a scheme.
    - A disincentive already exists for innocent users to make trades with strangers externally and that is that there is then no guarantee that both sides of the transaction will occur – that is, one party might accept the other party’s money and not send the currency that was purchased. This danger should be made clear to users, as well as the fact that they may be held accountable for complicity in money laundering, especially if they cannot provide verifiable records of the transactions they’ve made.
    - It would still be desirable for auction-style currency trading to be possible in order to allow market information to be generated about the appropriate value of the Zutha Dollar. The important thing would be that all transactions be declared publically with the details of the foreign currency account included. I just remembered that it should be possible for the transfer of foreign currency to be proved without a need to trust either of the Zutha users involved. For example, if user A uses his PayPal account to send money to user B’s PayPal account, PayPal will produce a signed message verifying that that transaction has taken place. This signed message can be published to the Zutha network and processed by hosts to verify that the agreed upon amount of Zutha Dollars should be transferred from user B’s account to user A’s.
    - All transactions of Zutha Dollars (call it ZUD from now on) should require an explanation to be provided.
      * A standardised automated system should be set up to allow the purchase of goods and services with ZUD. This system would automatically assign an explanation for each transaction such as “purchase of X device”. The trustworthiness of a transaction through this standardised payment system would then depend on the trustworthiness of the recipient. So payments sent to trusted sellers would be trusted, but payments of this kind going to low reputation users would be suspicious.
      * Another kind of valid explanation would be that the payment is in return for a foreign currency. For such an explanation to be valid, the transaction must be accompanied by proof of the foreign currency exchange. For example for a trade involving the transfer of $100,000 USD from user A to user B, valid proof of the transaction could be a signed message from PayPal verifying the transfer of $100,000 USD from an account associated with user A to an account associated with user B.
      * It would also be acceptable to explain a transfer of ZUD as a donation, but this explanation would be suspicious if the target is a low reputation user or item.
      * I can’t actually think of many legitimate use cases for sending money to another user on Zutha other than to sell ZUD for other currencies, pay reputable entities for products or services, or donate to reputable entities. One might want to send ZUD to family or friends who don’t have a high reputation user on Zutha, but that is unlikely to involve enormous amounts of the kind relevant to money laundering. Thus I think it will actually be practical to filter out a relatively small set of transactions that are suspicious and need to be investigated for money laundering.
    - It should be made very clear to Zutha users that, whatever the legal consequences of assisting money laundering in their country, if they send large amounts of ZUD to low reputation users without providing proof of receipt of another currency in return (such as USD via PayPal or BitCoin), their reputation on Zutha will suffer. It might even make sense to institute a system whereby users can be punished with a fine for making suspicious transactions that are found by consensus of interested and qualified Zutha users to be insufficiently justified.
    - Why not just ban the transfer of ZUD to users below a minimum reputation without proof of a reciprocal transfer of another currency?
      * Actually I think this makes a lot of sense.
      * It would make it impossible to provide financial aid to a friend or family member through Zutha if they didn’t have a sufficiently high reputation user on Zutha, but I don’t think this is a particularly important use case for Zutha and it is a small price to pay for what I think would be a significant reduction in the usefulness of Zutha for money launderers.

# Mon 3-Oct-11

* Zutha money handlers are required to accept currency trades through the automated trade system at exchange rates with a spread around the current market value of ZUD.
  + Auction style trading can go on within that spread for users that can be bothered waiting for a better deal. The currency auction will provide information about the accurate market value of ZUD that will be used to automatically adjust the official price of ZUD (and the spread centered around it).
  + Trades that go through the automated (random money handler selection) system will be counted in the calculation of the market value of ZUD. So if lots of users start selling their ZUD at the guaranteed price provided through the automated system then the system will know that the price of ZUD is too high.
  + Actually, I wanted to peg the ZUD to a bundle of goods (including foreign currencies). If a foreign currency becomes more expensive, the official exchange rate of ZUD with that currency should change accordingly to keep the spread provided by money handlers centered around the actual auction price. If the foreign currency in question is in Zutha's bundle of goods then an increase in its price will raise the price of ZUD by its proportion of the bundle in order to keep the price of the bundle constant as denominated in ZUD.
* The ZUD would be as robust as a traditional currency if money handlers were just given currency at a steady rate with no strings attached.
  + This is how BitCoin works – BitCoin miners are given money that is generated out of thin air. The work they have to do to get it is in many ways wasteful; its primary purpose is to restrict the creation of new BitCoin. Zutha can restrict the creation of ZUD just as effectively without as much wasteful work.
  + Traditional fiat currencies are even less robust than BitCoin because new currency is created at will, with varying degrees of irresponsibility across governments.
  + One of the concerns that I think people will intuitively have about buying ZUD is that they won’t be able to sell it again because money handlers might refuse to pay up when asked. One of the errors made in his concern is a common one among non-economists; the treatment of something’s degree of scarcity as binary – that is, either in sufficient supply or insufficient supply such that there will be shortages – when actually degree of scarcity is a continuum mediated by changes in prices. If holding ZUD became less desirable such that it becomes harder to sell it, then its price will fall. Even if all money handlers simultaneously ceased purchasing ZUD, there would still be some people who want it in order to make trades on Zutha, so you could sell ZUD to them even if you had to sell at a lower price than what you bought for.
  + When you buy USD, nothing guarantees that you will get back your money’s worth when you try to sell it to someone else – that is, there is plenty of risk of inflation.
  + If ZUD was just created out of thin air, it would behave like a traditional currency. It’s price would rise and fall according to demand and supply which would depend on the rate at which it is created and the desirability of having ZUD for such purposes as investing on Zutha and as a way to store money securely.
  + The ZUD will actually be more robust than a traditional currency however, because money handlers will be required (by reputational incentives) to buy and sell ZUD within a strictly controlled price spread anchored to a bundle of goods and currencies. This is equivalent to having a fiat currency system where the Reserve Bank will both give new currency to banks to inflate the money supply and confiscate money from banks to deflate it, and will do so with perfect judgement in order to maintain zero inflation.
  + Zutha will treat money as what it really is: an IOU that is redeemable from anyone. Where with fiat currencies, the IOUs are created by the government without ever being redeemable from the government (effectively meaning their creation is a tax on everyone else in the country), Zutha will complete the cycle and actually require the initial creators of the IOUs (the money handlers) to repay their debt. ZUD will thus be a medium through which complex multi-party trades can take place where everyone is treated fairly. In a fiat currency system, the creator of the currency is getting something for nothing so it is like a multi-party trade where a small amount of value is subtracted from each trade and given to a party (the government) who gives nothing in return. The fact that the government gets stuff for nothing is not a problem in and of itself, since the government (in theory) represents all the people in the country. The problem is that the tax is hidden, unpredictable, and unevenly and arbitrarily spread across users of the currency such that when someone uses the currency as a medium of exchange, they cannot be sure that it will allow them to make a fair trade, or indeed even how unfair the trade will end up being for them.

# Wed 5-Oct-11

* Dynamics of association type inheritance
  + A subtype Y of an association type X may define additional roles and properties. When associations of type Y are viewed from the perspective of X, the new properties and roles defined by Y are invisible. X thus acts like a projection of Y.
  + Subtype Y can refine (make more restrictive) the cardinality constraints on properties or roles defined in X. To do this, the refined role/property must be overridden by a new declaration in Y involving the same property type/role but with more constrained cardinalities. The overrides declaration is required in this scenario.
  + Subtype Y can refine a property type or role declared in X with a subtype of that property type or role. In most circumstances the refined declaration would override the declaration in X. I’m still trying to decide whether it would ever be useful to declare a refined (subtyped) role or property type in Y that does not override the declaration in X. This would mean instances of Y would have the option to use the refined role/property type, but would still be allowed to use the more general role/property type defined in X. This seems like an extremely messy concept and it seems very unlikely to ever be useful so I think I’ll just require that any declarations in Y involving a role or property type already defined in X must override the more general declaration.
  + It is possible that Y will declare multiple roles or property types that subtype a role or property type declared in X or that a role/property type declared in Y is a subtype of more than one role/property type declared in X. This is why explicit override declarations should be required.

# Thu 6-Oct-11

* Zutha should have a single standardized serialization format for the purpose of comparing host responses (like Topic Map Canonicalization format). All hashes of messages should be hashes of the message in this format. Hosts may respond in a different format, perhaps one more optimized for the particular kind of data being returned, but the hash they sign should be of the data in standardized/canonical format.
  + New data formats can be designed by users. They should have a well-defined reversible conversion algorithm into the standardized syntax.
  + Clients will specify the data formats they support in their requests similar to HTTP.
  + The standardized format should be called ZCF – Zutha Canonical Format

# Sat 15-Oct-11

* It should be possible for items to be deleted non-permanently.
  + This would be important for issues of privacy and defamation.
  + Deleted items should not appear in search results unless specifically requested.
  + The importance of deleting such content would be to prevent it being widely distributed and believed rather than making it inaccessible.
* If a user wants to create a public profile with identifying information, such as an advertisement with a contact number, the user must authenticate themselves as the individual identified. For example, they could provide a phone number which several high reputation users would need to call and vouch for before the phone number would be allowed to appear formally on the profile. Perhaps unauthenticated information could just be marked prominently as such.
* A good use case to grapple with in deciding the rules of overriding roles would be romantic relationships.
  + It would be useful to be able to describe a traditional marriage between a man and woman as a specialization of a more general romantic association between two spouses.
  + The general association would have only one Role: Spouse, which would be played twice.
  + The Traditional Marriage association would inherit from it and override the spouse Role twice, once with Husband and once with Wife.
  + This use case illustrates the need for overriding to be done explicitly, rather than relying on inference. It would be very difficult to automatically deal with cases where an inheriting Association Type defines more than one new Role that is a subtype of a Role defined in the supertype.
    - Actually maybe this could be done automatically.
  + Anyway, this is a useful Use Case to consider.
* The page that shows all the association fields of a particular type of the current item should be called that Item’s Association Field Set page
  + The URL for this page should be of the form: /07X56/Robert\_Jordan/Author/Is\_Author\_Of
* The page that used to be called a Role page should be called the Role Player List page
  + Its URL should be of the form: /07X56/Robert\_Jordan/Author/Is\_Author\_Of/Written\_Work
* Domain-specific views should be accessed by viewing an Item through the lens of its raw Item Type or one of its Traits
  + So the Robert Jordan item might be viewed as /07X56/Robert\_Jordan/Author which would show a specialised view dedicated to displaying information about Authors.
  + It is important to note here that Author in /07X56/Robert\_Jordan/Author is being treated as a Trait and Author in /07X56/Robert\_Jordan/Author/Is\_Author\_Of is being treated as a Role. If one were to try to access a page such as “/06XA/Australia/Container” where container is a Role not a Trait, this page would not exist, even though the page: “/06XA/Australia/Container/Contains” would exist.
  + This page should be called a Type page
  + Each Type page may potentially have multiple views available, each perhaps created by different people
  + Similarly, the details view may not be the only view for the global perspective on an Item
* Links to Items which treat the Item as a particular type should link to the appropriate Type subpage of the Item

# Sun 16-Oct-11

* Items should have a permission-level on deletion
  + Most new Items, like comments, should have a low permission-level on deletion
  + It should be relatively easy for someone who has a minimal amount of reputation to delete a comment they find offensive
  + Upon deleting an item, users should be expected to provide justification for the action, such as that it defames someone untruthfully or that it is spam or that it contains content of no legitimate interest to the public that undermines someone’s privacy.
  + When items are deleted, the initial creator of the item should be notified and the event should be placed in a list of recent deletion actions that anyone can watch if they wish.
  + If someone disputes a deletion, they need to provide a reason why they dispute the action, such as that the reason given for deleting it was not valid. The deleted item will then go into a limbo state where collective action (a decision market) will determine its eventual fate.
  + Actually, when an item is deleted, the user that deletes it should be able to specify the permission-level that prevents it being undeleted. The maximum permission-level they can assign is their current reputation in the relevant domain. The default permission-level blocking un-deletion should be the same as the permission-level on deletion.
  + A sufficiently high level user can un-delete an item unilaterally
  + As with any action that requires some permission-level, a user with insufficient permission to delete or un-delete an item alone may open a decision market to perform the action collectively.
    - Once a decision market is in progress, the only way to delete or un-delete the item will be to participate in the decision market.
    - A decision market must first grow larger than the current permission-level on deleting/un-deleting the item before it can have an effect
  + Upon completion of a decision market, a new permission level will be established on deleting/un-deleting the item according to the final size of the decision market. A user with sufficient reputation (more than the final size of the decision market) will then still be able to delete or un-deleted the item unilaterally.
  + It will be important to create a norm that punishes users for deleting or un-deleting an item without sufficient justification.
  + A set of explicit rules should be developed which govern when it is acceptable for an item to be deleted.
* I’m abandoning the idea of a Type subpage
  + The same functionality can be achieved by creating views with names such as Author.html or Person.html which are dedicated to displaying information relevant to Authors or People.
  + A URL such as “/07X56/Robert\_Jordan/Author” should be redirected to “/07X56/Robert\_Jordan/Author.html”. If the Author.html page doesn’t exist, a page no found error will be displayed.

# Sat 22-Oct-11

* Should role-player-list pages use transitive logic on applicable associations (like supertype-subtype)?
* I need a way to more formally define the equivalence between type-instance and item-has-trait associations and between supertype-subtype and item-type-implements-interface associations.
  + There needs to be a uniform way to refer to an item’s type or supertype that includes both ways of modelling each kind of association.
* It could be very useful to be able to create Calculated Association Types – Association Types that are never instantiated, but can be queried upon
  + A Calculated Association Type would contain a definition in some declarative language of how to find its virtual instances.
  + Example: Transitive Supertype-Subtype
    - Rule: Item A is a subtype of item B if:
      * (A,B) play the roles (subtype, supertype) in a supertype-subtype association
      * (A,B) play the roles (item-item, interface) in an item-type-implements-interface association.
      * There is an item, X such that A is a subtype of X and X is a subtype of B
    - This definition is somewhat confusing because of the reuse of the supertype- subtype terminology for two different things. That is one of the main issues I’m struggling with at the moment; how to distinguish linguistically between transitive (and other) calculated variations of associations and the similar concrete non-transitive associations they are based on.
* Is there really any need for an “item-type-implements-interface” association?
  + I’m starting to think it would be a lot cleaner to use supertype-subtype for inheriting from interfaces as well
  + One reason I think I initially liked the idea of a separate association for implementing interfaces was so that item-types could be required to have just one primary supertype, but now that I think about it, I cannot see any particular benefit of this structure. Why not allow multiple inheritance?
  + Multiple type-instance associations is a bigger problem and I think the item-has-trait association is still important.
    - Type-instance cannot be reified (because the reifier needs a type-instance association, creating an infinite loop) and so cannot be modifiable (because it cannot have an associated permission level). There is thus no elegant way to allow an item to have multiple direct types because any types added would not be removable.
    - It makes no semantic sense for an item to be an instance of two different types at once
  + The main purpose of an interface is to allow inheritance without allowing instantiation; I no longer see any reason why the inheritance needs to be modelled with a different association-type
* Without the item-type-implements-interface association it is easier to see a way to provide the transitive type and supertype relationships in an elegant way
  + Item-has-type association: auto-association between an item and each of its types (transitive)
    - Rule: T is a type of X if
      * (X,T) play the roles (instance, type) in a type-instance association
      * (X,T) play the roles (item, trait) in an item-has-trait association
      * There is an item Y such that Y is a subtype of T and Y is a type of X
  + type-has-supertype association: auto-association between a type and each of its supertypes (transitive)
    - Rule: U is a supertype of T if
      * (T,U) play the roles (subtype, supertype) in a supertype-subtype association
      * There is an item Y such that Y is a supertype of T and U is a supertype of Y
    - Name(supertype) = “Has Descendent”
    - Name(subtype) = “Has Ancestor”
  + The supertype-subtype association should be named “Has Direct Supertype” in the subtype scope and “Has Direct Subtype” in the supertype scope
* Concrete associations that will usually be used with transitive logic should be named to indicate that they are the “direct” association. They should have an accompanying auto-association that uses the more standard, less verbose name since this auto-association is probably going to be more typically used.
  + Example: contains association
    - Roles: contained, container
    - The concrete association should be called “directly-contains”
    - The auto-association should be called “contains” and be defined as the transitive version of “directly-contains”
* Every auto-association-type should have links to the association-types it uses.
  + This will allow for more efficient discovery of applicable auto-associations connected to an item.
  + When querying an item for its association-fields, association-fields defined by auto-association-types should be included
  + Example: the item-has-type association should have links to the type-instance and item-has-trait associations
    - If an item participates in a type-instance association (which all do), then it implicitly also participates in a item-has-type association
    - When an item X is being queried for its association-field-sets, its instance:type-instance association-field-set will be found. The type-instance association-type will be inspected for applicable auto-association-types. The item-has-type auto-association-type will be found leading to the discovery of an additional association-field-set of type item:item-has-type.
* It should be possible for a role declaration to override a role declaration in an ancestor association-type without the overriding role being a subtype of the overridden role
  + Consider inheriting from an association type: uses with roles user and used
  + Let’s say our subtype is for the connection between an auto-association-type and the association-types it uses and we want our roles auto-association-type and association-type to replace user and used respectively.
  + It would be handy to be able to specify that the auto-association-type role declaration in auto-association-type-uses-association-type overrides the user role declaration in uses without requiring that auto-association-type be a subtype of user
  + This is a pretty inelegant example. I think in this case it would be better to just use the uses association with roles user and used directly for connecting auto-association-types to the association-types used. A role-combination-constraint could be used to require that auto-association-types may only play the role user in a uses association where an association-type plays the role used.
* Is it important to stop item-types inheriting from more than one item-type?
  + This is something that the existence of the item-type-implements-interface association enables
  + Actually there is a need for multiple inheritance in my existing schema that I just discovered I implemented with a messy workaround.
    - A reified-property-type is a kind of reified-field-type as well as a kind of property-type
    - In order to allow reified-property-type to implement rather than subtype property-type, I had needed to make property-type an instance of interface while also being an instance of item-type so that it can be instantiated. I did this by creating field-type-interface, which is a kind of item-type while also implementing trait. All very messy.
    - With simple multiple-inheritance, I can just have reified-property-type subtype both property-type and reified-field-type.
* I don’t think I need Interface anymore
  + Type should have direct subtypes: Item Type, Role, Trait, Field Type
  + Only Item Type allows instantiation
* I think Field Type should be renamed back to Construct Type or maybe Scopable Type or something
  + The reason is that it is awkward to describe Associations as Fields as well as having a separate concept called Association Fields, which are actually more deserving of being thought of as Fields.
  + Ideally I want a name that merges the ideas of Construct and Scopable because Associations and Properties constitute the only constructs in the ZDM besides Item and they also constitute complete set of scopable constructs.

# Sun 23-Oct-11

* Does Item-Type need to refine Type’s subtype:supertype-subtype association field declaration to require at least one supertype?
  + Item will be the only item that has no supertypes
  + The item-type-has-root-item-constraint requires that all instances of Item-Type be a kind of Item.
  + So adding a requirement that Item-Types must have at least one supertype is redundant because it is already covered by the item-type-has-root-item-constraint. In fact, it will only add a necessity for the workaround of making Item a kind of Item.
* It really clogs up the interface of the details page to show empty association-field-sets mixed among the used ones
  + In view mode, I think empty association-field-sets should be omitted
  + Adding fields to empty field-sets should be done in the same way as Gmail
    - There should be an Add Field button which provides a dropdown menu to select a field-type that is currently unused by the item.
    - Once the field-set is added to the interface, the user will need to immediately create a field of that type, otherwise the field-set will disappear again.
* The only use case I can think of for applying a scope to an association is to specify the time period for which it is applicable
  + Example: a marriage association between two people that were married but have been divorced should be scoped by the period of time during which they were actually married
  + There are probably other kinds of scope that would be useful to put on an association, but I can’t think of any
* I’m pretty confident that there isn’t any use for requiring an association-type to be always instantiated with a particular kind of item in its scope
  + It is useful to be able to do this for properties though - for example, to require that a property always contain a scope item of type Language
  + Therefore there isn’t really any need for an association-has-scope-item-type-constraint
  + Similarly, I don’t think there is any need for an association-field-with-scope-item-type-declaration
  + Actually the marriage association is precisely such an example where it would be useful to require that marriage associations have a scope item of type Time Period.
* The scoped-construct-has-scope-item-constraint is almost redundant
  + If used with cardinality [1,1] it is providing no information because every single instance of the construct (property or association) would have the specified scope item and so it would provide no additional information
  + It cannot be used with cardinality [1,>1] because an item can only appear at most once in a scope
  + If used with cardinality [0,1] it says nothing because all items are already optionally allowed as scope items
  + If used with cardinality [0,0] it prevents a specific item from being allowed in the construct’s scope. This is the only case that has any use. If it every becomes necessary I think this functionality should be enabled by a dedicated association specifically for banning specific items from the scope of a particular kind of construct.
* The supertype of Property and Association should be Construct
  + The concept of Construct can de facto carry the implication of being scoped
* A better and less verbose name for scoped-construct-has-scope-item-type-constraint would be construct-scope-constraint
* Another use case for scope of associations so to scope an association to only be applicable to a fictional universe
  + Example: Hogwarts is a school in Britain, but only in the context of the Harry Potter fictional universe
  + It would be useful to be able to require that certain associations declared by a fictional entity be scoped by a Fictional Universe
* When specifying a scope constraint on declared associations, should the constraint be applied to associations or association-fields?
  + If the constraint is applied to association-fields it would give more fine-grained control over which associations the scope constraint applies to
  + If one association-field of an association must have a particular kind of item in its scope, this effects all the other association-fields of the association because only the association itself can have the scope. I don’t think this is a problem though.

# Wed 26-Oct-11

* Auto-types
  + Example: The Type “child” could be applied to Person Items with a date of birth less than 18 years ago
  + Auto Types should not be able to define fields
* Items with different direct types cannot be merged. Actually it should be possible to select which type to keep; if the merged item is valid after combing everything except one of the direct types, this should be accepted. This is to allow easier recovery from making a poor choice of direct type for an item (in fact the only other way to recover from this is to create the entire again from scratch.

# Thu 27-Oct-11

* What stops someone from creating a valid event with a timestamp long in the past?
  + You would need to find a request with a hash that leads to the selection of a set of hosts that you have control over and you would need to use the timeframe key of the timestamp you want to associate with the event.
  + This would be quite possible to achieve given enough time, so there will need to be a way to block Modification Events from being accepted by the rest of the network too long after their asserted timestamp.
  + I wonder whether I can use any ideas from BitCoin?
    - It won’t work to use Proof of Work in Zutha as it would be far too wasteful of computation power, however creating a Merkle Tree of events accepted by the whole network so far might be helpful.
  + Modification requests are given a timestamp by a random request handler group, but the request must also be confirmed by the controller groups of the items the request effects in the timeframe of the request's timestamp. This controller group will only accept the request if it is received within one timeframe after the timestamp of the request.
    - An attacker could potentially choose his request's request handlers with enough time and computing power, but with this rule his time is limited to one timeframe. The attacker has no control over the controller groups of the items he wants to modify.
    - Hosts that receive a request before it has been signed by the relevant controller groups can be pretty confident in accepting it if they receive it within the acceptable time window. They can also make a tentative judgement about whether they think it is semantically valid.
    - The controller groups will give final verification that the request is valid semantically.
    - What if some members of a controller group receive a request on time and some don't? They could use the same system as request handlers to select their own timeframe key. Then why not have the controller groups simply be the request handlers? That would reduce latency and have the added advantage if giving attackers no control over the selection of request handlers.
    - Each controller group involved in verifying a request (one for each effected item) should act as an independent entity in coordinating the timestamp selection between groups. Hmm, but messages still need to be sent between individual hosts. I need to avoid letting the request handler group become too large since the number of messages that need to be sent to agree on a median timestamp is O(n^2).
    - The request handler group could be selected from among the members of all relevant controller groups by using the message hash in the selection algorithm. They would agree on the timestamp and zid and then propagate this group-signed event to every relevant controller group using a propagation technique I invented a while back to avoid duplication of work (the message has multiple addresses and nodes along the way forward the message to the appropriate neighbours only the first time they see the message.) Each controller group would then group-sign the timestamped request if it is valid from the perspective of their item.
    - No. There is no need for the controller groups to verify the validity of the item. That can be done by the request handlers provided they have up to date information. In fact the validity of the request cannot be completely verified for a while after it is created because there may be other requests being generated at about the same time which conflict with it.
  + Each item's controller group should accumulate a hash of all the events they have accepted that effect their item.
    - Let X be a host in the controller group of an item Q that is effected by a new request R. When X first receives R from the request handler group that timestamped it, X will immediately fetch all information needed to verify the validity of R. X will also ask various controller groups to be notified of any new events that arrive with a timestamp earlier than R that conflict with R.
    - No, there is a problem with this whole approach. With sufficient network interruption and a complex enough dependency network I think it would always be possible (however unlikely) for conflicting events to be validated by different sets hosts. I need to rely on the timestamp as the definitive resolver of conflicts even long after the events occurred.
  + Once a request handler group H has timestamped a request R with timestamp T, making it into an event item E, E is propagated to all relevant controller groups to R (call this macro group W) as well as a new controller group G for the new event item. Every member of W will then send a signed message back to G containing the time they receive the request. These signed messages will be included as properties in E. Let N be the number of hosts in macro group W. Event E is considered definitively chronologically valid as soon as it contains at least N/2 signed timestamps from members of W that are less than one timeframe after T.
    - W members will only sign E if it is an allowed event according to the information they have when they sign it. Even after E has become chronologically validated, it may still be invalidated by an earlier, conflicting, chronologically validated Event that was unknown (or not yet validated) to more than half of W when they signed off on E.
    - If an attacker is able to cause a selection of H such that he has control of at least half of H, he has the ability to select the ZID which will determine G for the next timeframe. Even if he was able to select a ZID which picks G such that he owns more than half of G, he cannot push through a forged timestamp without the signatures of more than half of W. Thus control of more than half of W is required to forge timestamps and W is not selectable by an attacker.
  + It really will be necessary for events to eventually become guaranteed in some way. It will not be acceptable if there is even a slight possibility that an event will turn up hours or days or even years after its timestamp and invalidate lots of other events.
    - It might be implausible for an attacker to gain control of more than half of W at the actual time he wants to create a forged event. However, an attacker could work away at hacking into and obtaining the private key of each host in the W for a request he crafted months ago but didn’t submit. Once he has enough private keys, he can forge the requisite signatures to prove all stages of the timestamp verification process.
  + Once an event E is one timeframe old, at least half the members of at least one controller group G­i whose item Xi has a dependency relationship with E, must necessarily know about any events that occurred before E that affect Xi.
    - Actually to make this statement true, I need to tweak the rule for verifying the timestamp of an item: at least half the members of each controller group in W must sign a timestamp not exceeding one timeframe after the event’s official timestamp. Previously my rule had only been that at least half of all the members of W together were needed.
    - Another component of the above statement needs verifying: If an event E1 depends on the set of items S, can another event E2 conflict with E1 without having a dependency relationship with at least one member of S? If the answer to this question is no, then my assertion above is true. I’m pretty sure the answer is no. Fundamentally, a conflict implies that something that depends on both events cannot simultaneously take on the value implied by each events.
  + So if you are impatient and want a guarantee that an event E is valid as soon as possible, you can get it a little more than one timeframe after the event’s timestamp by querying all the hosts in W, the union of controller groups of items that have a dependency relationship with E. Once you have received replies from more than half of each controller group in W, if none have sent you proof of a conflicting event, you can be sure none exists – unless some of the members of W are lying, that is. Damn!
    - All the hosts that had received the conflicting event would have to be lying. There are only two ways I can think of that this would happen.
      * 1. An attacker has control of all of W, or is able to prevent messages reaching whichever members of W he doesn’t control
      * 2. For some reason, some hosts in W decided to lie about seeing an event that conflicts with E and they just happen to be the only hosts in W who have received that conflicting event.
    - Both of these possibilities are extremely unlikely, but not *impossible*.
  + Every item should have an accumulating hash chain H that is added to upon each event that affects that item
    - Once an event E has been chronologically verified with timestamp T, its controller group G will notify all members of its dependent controller groups W about its verified status.
    - At a time one timeframe after T (call it T1), if there are going to be any conflicts with E, one of the controller groups in W will know about it. They will immediately notify the other members of W of the conflict. If a member of W does not receive notification of a conflict, though, it cannot take absence of proof as proof of absence.
    - From T1 until the next timeframe boundary B, the status of E remains tentatively valid.
    - At time B all the members of B will send updates to all the others containing any conflicts they encountered during the timeframe preceding B. If they received no conflicts in that time, they say that.
    - Once a host in W has received notice from more than half of every controller group in W that there was no conflict with E, it will add E to its hash chain of valid events.
      * A block of data D is added to a hash chain H, making element Hi+1, by hashing the combination of D and hash Hi, the current last element in the hash chain.
    - Actually, after time B, each host will wait until it has received sufficient confirmation of validity for all the events it received in the timeframe preceding B. It will then append all of these events to its hash chain, locking them in as all valid.
    - Now, even if an attacker can gain control of the set of hosts that would have constituted the set of controller groups W responsible for managing the creation of his event at the time in the past when he wants it to have occurred, he cannot insert that event into the hash chains of the effected items because those hash chains have been extended by an evolving set of hosts. The attacker would need to steel the private keys of all hosts who have managed the effected items since the timestamp of his forged event, including the present set. Hacking the present set is of course the hardest because it must be done within one timeframe.
    - When controller group G1 adds a new event E1 to a hash chain H1, the last hash chain entries before E1 of all controller groups affected by E1 should be used by G1 to generate the next entry in H1 (the one for E1).
      * For this to work, I think controller groups will have to exchange confirmations about event conflicts on a per-event basis rather than a per timeframe basis. This is because, in order for a controller group other than G1 to be able to tell G1 the hash chain element right before E1 in their hash chain (call it H2), they need to know the last hash chain elements of the items affected by the event E2 associated with H2 and this will probably depend on other events that also occurred in the timeframe immediately before B.
  + If the controller group of an item at which a conflict occurs is entirely controlled by an attacker, that controller group could tell the other dependent controller groups of both events that all is well.
    - Responsibility is placed on the controller group of the item at which a conflict occurs to notify other hosts of the conflict, so if this controller group is malicious, inconsistent events could potentially be accepted.
    - The situation is nonetheless far better now than it was when an attacker could potentially forge an event with a timestamp one year ago. Now an attacker can potentially cause a conflict to last more than one timeframe (with extreme difficulty), but when the conflict is detected, it can be resolved so the situation can only get better over time – that is, one can become ever more confident in the validity of events as time progresses. This is as good a level of trust events as BitCoin achieves. I would like better though.
* Won’t practically every event depend on Item? Every event will depend on the properties and associations declared for the items it effects and on other schema rules and fundamentally, every schema rule is dependent on Item if you follow the dependency tree all the way up.
  + If this were the case it would really stuff things up because then every single request would have to go via Item’s controller group which effectively makes the rest of the network redundant, at least in terms of scalability.
  + I think there is a way out of this problem though by looking more carefully at the rules for identifying dependencies
  + The rules for determining the dependencies of an event must meet essentially just one criteria: if two events conflict, they must have a dependent item in common.
  + Let’s say, for argument’s sake, that Item defines a property P1 that as of time T1, no Item has yet instantiated. Let’s imagine that, at around time T1, someone tries to create an instance of property P1 on item X (event E1) and, at around the same time, someone tries to remove the declaration of P1 from Item (event E2).
    - Both actions would be permitted on their own. P1 can be removed from Item without making any other changes because the property has not been used, and a P1 property can be added to X because it is a property permitted by Item.
    - I think this is close to the most pathological situation that the dependency rules will have to cope with
    - So, there needs to be an item that has a dependency with both E1 and E2 and it would be preferable if that item were not Item itself.
    - Since Item Types will be modified far less frequently than ordinary items, they should take on as much of the burden of dependencies as possible.
    - If we use the rule that an event has a dependency with any Item whose semantics might be changed, however indirectly, by the event, then event E1 (removing the declaration of property P1 from Item) has a dependency with every single item in the knowledgebase because every single item now has different rules regarding what properties it is allowed to have.
    - So X is the item that depends on both E1 and E2 here.
  + What if a P1 property is instantiated with X in the same event that creates X itself?
    - If P1 was undeclared (event E2) soon before the request to create X with a P1 (event E1), then X wouldn’t be one of the items in E2’s dependency set because X didn’t exist at the time E2 occurred.
    - E2 will propagate to all items by being passed from type to subtype (starting from Item) and finally from types to their instances.
    - So X’s direct type Y will need to know about E2 and the creation of X (E1) depends on Y because a type-instance association must be created between X and Y. Thus Y is the common dependency of both E1 and E2
  + My intuition says that this simple rule for determining dependencies (find any items whose semantics will change as a result of the event) will satisfy the requirement that any two events that conflict will have a common dependency. However, I haven’t proven this yet.
  + Actually I think the above two statements are tautological. The definition of a conflict between two events is that the first one changes something that makes the other one invalid. So if I find every item in the knowledgebase whose rules change as a result of event 1, then if event 2 conflicts with event 1, it must try to change one of these items in a way that is no longer allowed. Whichever item it is, this item has a dependency with both events because both try to change it.
  + Let’s reinterpret the case of creating a new item X with property P1 (event E1) after P1 has been undeclared by Item (event E2)
    - E2 changes the semantics of Y (X’s type) because it changes the schema rules surrounding what conditions have to hold for an item to have a type-instance association with Y. Instances of Y are no longer able to have a property of type P1.
    - E1 effects Y by gaining a mutual association with it (the type-instance one)
  + All schema rules have some effect on what conditions have to hold for particular associations are properties to be instantiated. If a schema rule (or any of its arguments) is changed, the dependencies of this event E must include all the places (items) in the knowledgebase where any of the particular associations or properties involved in the schema rule might be created or modified. Whenever any of these associations or properties are modified therefore, the item(s) involved must necessarily be in the set of items dependent on E.
  + Using the phrase “a schema rule (or any of its arguments)” necessarily includes all pieces of data in the knowledgebase whose modification could cause conflicts because conflicts are defined as a violation of one or more schema rules.
  + I think I just thought of a counter-example: What if Item un-declares P1 (E1), then X is created (E2) and then soon after, a P1 property is created in X (E3)? What is the common dependency item of E1 and E3?
    - ???
* If someone wants a very high level of confidence that an event is valid (for the sake of a large financial transaction for example) they can trace back events as far as they desire until they are confident that there is no conflict.
  + Let’s consider an example where a vendor V receives a payment of $1 million ZUD (Zutha Dollars) at time T1 from consumer C to whom V will give something worth $1 million ZUD in exchange once he is satisfied that the transaction is valid. Let the event of C paying V $1 million ZUD be E1.
  + The transaction E1 can only be reversed if events that occurred before E1 are validated in the future that cause C to have less than $1 million ZUD at the time of E1
  + The common dependency item for such a conflict will be the user item C because the schema rule that will be broken is that a user can only make a payment of an amount less than or equal to the amount in his account and the field that will be changed to break this rule is C’s account balance, which resides in C.
  + Let’s consider that C is actually trying to pull off a deliberate attack on V and has control of all the hosts in the controller group C\* of his own user item, C.
  + Let’s say that at a time T0, just before T1, C made a payment of $10 million ZUD to another user K, emptying C’s account. Call this event E0.
  + E0 has a dependency with C and K. The controller groups of C and K (C\*,K\*) will thus need to sign off on the timestamp of E0 and send it to the controller group of E0 (E0\*). E0 will become chronologically valid once at least half of C\* and half of K\* have signed a timestamp for it that is less than one timeframe later than T0. Let’s call such a time T2 and assume that it is after T1.
  + At T2, C\* and K\* will update each other and E0\* of whether they saw any conflicts with E0. Along with this message, they will send the hash chain entry of the event that immediately precedes E0 in their hash chain. This hash chain entry will be the most recent one they created since E0 has not been confirmed yet.
  + Let Hk be the hash chain entry of K that immediately precedes E0 – the one K\* sends to C\* in the above step. And let Hc be the hash chain entry of C that immediately precedes E0.
  + Once C\* has received confirmation from K\* that no conflict with E0 occurred, it will be confident that E0 is a confirmed event. C\* will then create a new entry in its hash chain for E0 which combines the hash of E0 with Hk and Hc.
  + If K (that is the actual person the K User item represents) wants to verify the authenticity of E0, he will ask for the hash chain of item C. K can then examine the event items preceding E0 in the chain (which he will get from the controller groups of each event) to verify that C had enough funds to pay him at time T0. C will thus want to provide the correct hash chain to K if and when it is requested.
  + At time T3, one timeframe after event E1 occurred, it will be time for C\* and V\* to update each other and E1\* about whether they encountered any conflicts with E1. C\*, of course, did see a conflict with E1: E0, but C (who controls C\*) wants to trick V\* (and through it V) into thinking E1 is valid, so C\* reports no conflicts with E1. C\* must therefore omit E0 from the hash chain it exposes to V\* so it must fork its hash chain.
  + Now C\* has to figure out which hash chain to send whenever it is requested. If hash chain requests are made anonymously (which should be made the protocol) then C\* will easily be caught red-handed.
  + Both V and K are both likely to care a lot about verifying million-dollar transactions so in this case, they are both likely to (anonymously) request the hash chain from C\*. If one of them discovers a discrepancy with the hash chain entry that was sent by C\* with its verification of the authenticity of their respective transaction events, they will immediately report it (I need to address what “reporting” means).
  + Another hurdle to C’s exploit attempt is that in the next timeframe, a new random host is going to join C\*. C\* will have to choose one branch of its hash chain to stick to from that point onwards, because otherwise this new member will detect the deception. The newcomer will use whichever hash chain the rest of C\* gave it when it joined and the next time the rest of C\* uses the other chain to respond to a request for the hash chain or in the message that verifies a new event depending on C, the newcomer will see this conflicting message. The newcomer will then be able to prove the deception of the rest of C\* because it can produce conflicting signed messages from C\* members; some using one hash chain and some using a different one.
  + But what if, say, K imprudently fails to ask C\* for its hash chain after waiting a few timeframes (to allow new entrants into C\*) and C\* henceforth abandons the fork of its hash chain that it exposed to K\*? Which takes precedence when the conflict is finally discovered: the longer hash chain after E1 or the earlier timestamp of E0?
  + If K later tries to buy something with the $10 million ZUD he got from C, what should the recipient, Q, of his money do to verify that K has that much money?
    - Q can examine the last few transactions that occurred with K to a few layers deep.
    - To examine E0 to two layers of depth, Q would check that it is in the hash chain of item K. Q would examine the other dependencies on E0 – in this case just item C – and verify that, in C’s hash chain, Hc (the hash chain entry from C that was included in K’s hash chain entry for E0) is followed immediately by a hash chain entry for E0. If Q did this, he would discover that Hc is not in fact followed by E0 in C because, to try to avoid being caught, C\* has switched to using the hash chain that contains E1 instead of E0.
  + So when Q finds E1 in C’s hash chain, an event that occurs after E0, conflicts with E0, but has been accepted by C\*, how should the conflict be resolved?
    - Arguably, K should be punished because it was K’s carelessness that allowed the conflict. That would mean accepting the event that made it into C’s chain but rejecting the event that made it into K’s chain. Why would C get precedence over K in determining which event is accepted? Well C is the item at which the conflict occurred. So maybe the rule could be that the event that makes it into the hash chain of item at which a conflict occurs is the event that is accepted.
    - C would then be able to pay K (E0), deliberately create a conflicting event (E2) and then cause C\* to accept E2 into its hash chain soon after verifying E0 for K\*.
    - It would be much simpler if timestamps remained authoritative in resolving conflicts.
    - The hash chains make it implausible to inject an event long into the past because of all the hosts that would need to be sabotaged to spoof all the intermediate hash chain entries.
    - If timestamps are authoritative in this scenario, though, K benefits when K was responsible for allowing the conflict. This eventuality needs to be avoided then – that is, we cannot rely on K to detect C\*’s treachery in time.
  + One of the dilemmas in this problem is that V doesn’t know about E0 or K, so V cannot ask for information from or about them that will show E1 to be invalid.
* Oh! I just realised that the difficulty of injecting an event into the past is dependent on the precedence of the hash chains over the timestamp. So it looks like I have to figure out the rules for precisely when a hash chain can overrule a timestamp and which hash chains take precedence in determining the resolution of a conflict.
* What if a pair of events, Ea and Eb, conflict with each other at two different items, X and Y, and X\* accepts Ea into its chain and Y\* accepts Eb?
* What if being the earlier timestamp and being part of a valid hash chain are both prerequisites for an event being accepted?
  + E0 would then take precedence over E1 in the problem above because E0 is earlier and has also been accepted into a hash chain (K’s) which is valid as long as E1 is not considered valid.
  + With these conditions, an attacker still cannot inject an event into the past, because he needs to get it accepted into a current hash chain, which requires changing all the hash chain entries between his event’s timestamp and the present. Even if this was managed, the deception could be easily discovered because all the signed messages using the old, correct hash chain would still be floating around.
* Hmm… what can I do with the fact that it will be easily discoverable which hosts are involved in an exploit attempt.
  + It will be easy to discover, for example, that C\* (at least its members at time T1 to T2) has been attempting fraud because K\* and V\* will be able to produce messages from C\* that conflict with regard to C’s hash chain.
  + Does it make sense to accept C\*’s current hash chain as truth in order to punish those who failed to check what C\* was doing? I don’t think so. I sense that it would be possible for C\* to use this protocol to its advantage for some other exploit. Also, what if C\* tries to confuse the network by spouting out more than one different hash chain until its members are all evicted? Or it could spout out pure bullshit or cease functioning and take down the latest data about its item with it.
* There are actually quite a lot of destructive things that could be done by a malicious agent with control over an entire controller group. I think, though, that the random nature of controller group selection will make it virtually impossible for someone to gain control of an entire controller group except by chance, in which case they won’t be able to choose which item they gain control over.
  + The only thing that is essential, then, is to unsure that pathological controller groups cannot break anything, but if they are able to pull off some hack that benefits them – good for them; It had better be worth more than the combined reputations of all the hosts that participate in the exploit because they will all very likely be evicted from Zutha for their disobedience.
  + So, whatever a controller group does, however pathological, the response of the network must be well-defined.
* Conflicts should be resolved as follows:
  + Only events that are included in at least one current hash chain have any chance of being considered valid. If an event E is not in any item’s hash chain then it cannot have had any effect on anything (all items effected by E must necessarily include it in their hash chain) and for all intents and purposes, does not exist.
  + If a conflict is discovered between two otherwise valid events, the event with the earlier timestamp takes precedence.
* In the scenario described above where C tries to make a double payment to K and V, the payment to K (E0) would eventually be considered the valid one because it happened first and became included in K’s hash chain.
  + K\* and V\* should both be required to verify E0 and E1 respectively, a few timeframes after they occur. They will then catch C\*’s treachery.
  + Even if C controls both C\* and K\* at time T0, he cannot prevent the new members of K\* that arrive after T0 from sending the request for C\*’s hash chain. By this time, C\* will also have some new members which will respond to K\*’s request with a conflicting hash chain. The honest members of K\* will then notify C\* and V\* (which K\* knows about from C’s hash chain) of C\*’s treachery.
  + If C manages to obtain control of the new members of K\* before they send a request for C\*’s hash chain (note that this is far more difficult than merely stealing their private key) then C may sustain his double-payment until someone like Q comes along and verifies K’s account balance, at which time Q will discover the discrepancy between K’s hash chain and C’s hash chain.
  + If someone can pull off such a feat then so be it. I would venture to say that doing so would be far more difficult than taking control of BitCoin given a sufficiently large number of hosts. It would also be a far shorting-lasting boon than taking control of the BitCoin network because as soon as the treachery is discovered, the attacker will lose their hard won, high reputation hosts (of which there must be some in every controller group).
* A possible structure for a hash chain:

[

{

“Event” : “17EX09Q”,

“Dependencies” : [{ “Item”: “1B9E2”, “LastHTE” : “e8428f02…” }, { “Item”: “03XF78”, “LastHTE” : “8be9acd2…”} ],

“Hash” : “f3de103b…”

},

{

“Event” : “0P23XE”,

“Dependencies” : [{ “Item”: “07FED”, “LastHTE” : “2bec1ac2…” }, { “Item”: “04B0DD”, “LastHTE” : “02e7a02be…”} ],

“Hash” : “8ba1decc7…”

}

]

* Where
  + LastHTE e8428f02… is the hash of the hash table entry in item 1B9E2 just prior to the hash table entry for event 17EX09Q.
  + Hash 8ba1decc7 is calculated by hashing the combination of hashes: f3de103b, 2bec1ac2, 02e7a02be and the hash of event 0P23XE.
  + Obviously I have just made up the hashes and they don’t match the rule stated above.
* In the scenario above, when Q discovers that C’s hash chain does not have event E0 immediately following the Hc HTE (hash table entry) that was used to put E0 in K’s hash chain, Q will notify all the items that are dependencies in hash table entries following Hc in C’s hash chain. This set of items will include V, which needs to know about E0 because it conflicts with E1.
  + All the items following Hc need to be notified of the conflict, though, because they will all need to recreate their hash chains from that time forward. They can start doing so immediately because they can easily retrieve the requisite information to resolve the conflict and generate the correct hash chain for C themselves. However, C’s corrected hash chain will not become official until C\* has been purged and its new members agree on the corrected hash chain and start serving it to those who need to know it. Before then, hosts and clients will need to do a lot of processing themselves if they want to be confident of C’s correct hash chain.

# Fri 28-Oct-11

* I still need to resolve a problem I encountered with determining the common dependency of a conflict: What if Item un-declares a property P1 (event E1), then item X is created (E2) and then, soon after, a P1 property is created in X (E3)? What is the common dependency item of E1 and E3?
  + Of course! It’s the property type item P1
  + When Item un-declares P1, it changes the semantics of P1; P1 is no longer allowed to be instantiated by any item that is an instance of Item.
  + When a P1 property is instantiated in X, it effects P1 (the property type) because it creates a dependency on P1 which it is P1’s responsibility to police for schematic validity.
* If Property Types (and probably Association Types too) are going to be dependencies for every event involving them, then their controller groups are going to have an extremely high workload
* Consider the controller group for the Name property. Every time an item is created or its name is modified, the controller group for Name will have to timestamp the request and confirm whether any conflicts occurred with it after one timeframe. This would be an enormous amount of work and would make it impossible for machines with insufficient resources to keep up so Name’s controller group would have to consist of 100% extremely powerful servers.
  + Name seems to be an atypical case in this respect though. I can’t think of any other items that would be under such high workload.
  + I can think of quite a neat way to resolve the problem of name having extreme workload: The Name controller group can be divided up into smaller groups with responsibility for managing only a subset of Names, such as Names beginning with “K”
    - One simple way to achieve this would be to have many subtypes of Name, each with responsibility over a subset of names. So there would be a “D Name” and a “TH Name” and an “X Name” for example.
    - This approach can kill two birds with one stone because another responsibility of Name would be to detect duplicate Names and this can easily be a partitioned responsibility because, for example, a name starting with “TH” can only be the same as another name starting with “TH”.
    - I had already considered this idea of partitioning the Name-space for the purpose of indexing – well that kills another bird with the same stone.
      * Indexing would actually require events effecting Names to depend on additional Index items for each word in the name, but at least the index on exact matches of Names could be provided by the same items that resolve Name duplicates and divide up the responsibility of registering Names.
    - I don’t think it would work to have subtypes of Name like “A Name”, “B Name” and so forth, as that would require changing the type of a Name property when its first letters change.
    - Name could just be treated as a special case it is hard-coded into the Zutha standard that the Name controller group is partitioned into parts according to an agreed-upon set of Name prefixes. I don’t like this approach, though – it is inelegant and goes against Zutha’s whole paradigm of extensibility and generalizability.
    - If a Name is changed such that its prefix is changed, this could be treated as the deletion and creation of a Property item. Hmm… this idea seems ugly at first glance, but I’m starting to think it is actually rather elegant.
      * It is the type-instance association between a Property item and Name that encodes the dependency with Name. This association is what will be used by parsers of events to determine that an event that creates a Property has a dependency on the item at the other end of the type-instance association created by this event. It would be most ideal if Name could just use the same semantics as any other property.
      * Changing the prefix of a Name would actually be a big deal since the name would have to be moved to a new “registrar”, as it were: the “\_ Name” item responsible for the new prefix. It kind of makes sense that such a change would involve creating a new item and deleting the old one.
      * I think this is the way to go.
      * Both actions (deletion and creation) would have to occur in the same event, of course, because otherwise schema rules (such as the requirement that an item have at least one Name) might be broken between the deletion of the old name and the creation of the new one.
      * The permission level on the old name should be automatically copied to the new name
    - There is a certain degree of special semantics with this name structure because there is nothing currently in the standard that permits the rule that upon changing a name’s prefix it should be deleted and recreated with the appropriate Name type.
      * This rule doesn’t actually need to be encoded in the standard though
      * The “A Name” would be defined with a regular expression syntax “[Aa].\*” and the other “\_ Name” property types would have equivalent restrictions. The command to change a name such that its prefix is modified would actually be a compound command that requests the deletion of the old name and the creation of the new one.
      * Clients could encode the convenience logic that would hide these semantics from the user and allow the user to treat modification of a name the same way regardless of whether it modifies the prefix or not.
  + If a name is changed such that the first letter changes from “S” to “T”, say, then this event will have a dependency on both the “S Name” and “T Name” items.
* Let’s clarify what I have come up with for the implementation of Zutha’s distributed clock
  + A user U creates a request R to perform some modification to the Zutha knowledgebase
  + R has hash Hr
  + Let W be the set of controller groups Di\* whose item Di has a dependency relationship with R
  + Hr is plugged into the selection algorithm, along with the current timeframe key, to select a random set of hosts to be the request handlers Q, of R
    - I have abandoned the idea of selecting the request handlers from among the dependent controller groups because, if an attacker has control of the controller group(s) then he can choose the timestamp (within the boundaries of the current timeframe). If the request handlers are random however, it would be virtually impossible for an attacker to gain control of half of them within the one timeframe that he knows which hosts they will be.
    - The key distinction between the request handler group and the controller groups that is important here is that all of the request handlers are unknown for the next timeframe, whereas only the one new member that will join each controller group for the next timeframe is unknown.
    - It must be as difficult as possible for an attacker to gain control of at least half the request handler group because, with this, he gains some discretion over timestamps. Gaining control over the request handler group is made maximally difficult by making every host in the network a potential member.
    - The original reason for considering taking the request handlers from among the controller groups was because I was worried about an attacker gaining control of the request handlers of his desired request for a past timeframe, but the hash chain system prevents that kind of attack.
  + U bulk-sends R to Q
    - Bulk-sending means addressing a message to multiple recipients rather than sending multiple individual messages. Bulk sending takes advantage of the ability of intermediate nodes to discard redundant messages.
  + Each member qi of Q gives R a candidate ZID and timestamp upon receiving it and then bulk-sends the signed candidate event item Ei to the rest of Q and to W, the set of controller groups dependent on E.
    - The reason qi sends Ei to the rest of Q is because it is possible that U neglected to address R to the correct set of request handlers. Then again, maybe U’s message should simply be considered invalid if he does this.
    - Another reason for qi to send Ei to the rest of Q is to improve the likelihood that all of Q will receive R on time.
  + If E is a kind of event that is time-critical, such as in a MMOG context, then some controller groups in W might immediately forward each candidate event Ei to the other controller groups they know depend indirectly on E.
    - For example, in an MMORPG, U would be a user in the game located in a particular game cell C1 with controller group C1\*.
    - U\* and C1\* would constitute the set of dependent controller groups W of request R.
    - R might be a request by U to fire an arrow at an enemy user Ue (who might perhaps be invisible to U). At any rate, it would not be U’s responsibility to notify Ue of the approaching arrow. C1 will be the common dependency for any events issued by U and Ue that might conflict.
    - U would bulk send R to Q
    - Each member of Q would timestamp R and forward it to W as Ei
    - W would then forward each Ei to any other controller groups that need to know about E in a time-critical way. This would include Ue\*.
    - Ue\* would then forward the candidate Ei to Ue so that Ue’s client can respond as soon as possible.
    - Ue would thus receive the first timestamped candidate event item E0 almost as soon as is physically possible on average – which is about the time taken for light to travel ½ earth’s radius
    - In some cases, Ue might even receive E0 before E will actually occur because the time taken for E0 to get from U to Ue is less than half the spread of the timestamps given to E by the members of Q.
    - Ue would be able to estimate E’s eventual timestamp upon receiving the first candidate E0 because Ue would know some statistics about the usual spread of timestamps for a typical request.
    - Once Ue has received half of the candidate events it will be pretty confident of the median timestamp, and once it has received all the candidate events it will be certain of E’s timestamp, but not yet 100% certain of its validity. It can check E’s validity itself though and will, in the vast majority of cases I think, be able to judge it correctly.
  + When member qi of Q or a member wi of W receives a timestamped candidate event E1 from member q1 of Q, qi/wi should immediately reply indicating it has received q1’s message.
    - The norm for all kinds of communication between hosts should be that every communication ends with a confirmation of receipt of the last message.
    - Clients would not send confirmations to hosts that they received the response to a request though, because clients achieve reliability of response via redundant requests to multiple hosts.
  + If q1 has not received a confirmation reply from qi /wi after a fixed wait period, q1 should resend its candidate event E1 to that host
  + Once a member w1 of W has received all of the candidate events, w1 will identify the candidate event E with the median timestamp (call this timestamp Te). w1 will verify the semantic validity of E and check that it was received less than one timeframe after E’s timestamp. If E passes validation, w1 will send E along with a success message and the signed timestamp of when w1 received it, to the rest of W and to E\* (a new controller group formed from E’s ZID). If E fails semantic validation, w1 will send an “invalid event” message to the rest of W and to E\* and E will be abandoned (presuming the other hosts agree with w1’s assessment that E fails semantic validation). If E is too late, w1 will notify the rest of W and E\* of this, but this doesn’t necessarily mean E will fail.
    - To clarify, the timestamp w1 should attach to its message should be the time it received the candidate event that ended up being E. This means w1 has to keep track of when it receives each candidate event. The timestamp it attaches to its success message then, may be earlier then the time it sends the message, because it has had to wait until it has received all candidate events before it knows which one will be picked as E.
    - This rule allows for some members of Q having high latency. Every member of Q will still need to contribute its candidate event before E can be accepted, but even if this takes longer than one timeframe, it won’t cause the event to fail because the median event (which is unlikely to be from a request handler with very high latency) should still have reached all of W within one timeframe of its timestamp.
  + If w1 has received an incomplete set of candidate events Eci that conflict with E, w1 will wait until it has received all of Eci so that it knows whether Ec came before E. w1 will only sign whichever of E and Ec turn out to be the earlier event, but it will remember the timestamp of the second event in case the first event turns out to have failed.
    - If the first event fails (let’s assume Ec was first and it fails), w1 will know this if it receives an “invalid event” message that w1 can verify is true or if w1 receives “too late” messages from more than half of one of the controller groups in Ec’s dependency set. w1 will then reincarnate E, dispatching a success message for E that contains the timestamp of when w1 received it.
    - If Ec succeeds, which w1 will know if it receives success messages from more than half of each controller group in Ec’s dependency set, then w1 will send an “invalid event” message for E.
  + If no members of W ever receive all the candidate events, E will never be created. This should only occur if at least one member of Q has become completely disconnected from the network. This is a disastrous enough event that I think it acceptable to allow requests to fail because of it.
  + E\* and W will be able to accept E as valid as soon as they have received a success message from at least half of each group Di\* in W.
  + Let’s call the time one timeframe after Te, Te+1.
  + If a member of W or E\* receives a message from more than half the members of a group Di\* of W that has a timestamp later than Te+1 then that member will know that E has failed and will abort it. Eventually, all members of E and W should discover that one of the controller groups of W did not receive E in time and they will all abandon E.
  + Once a controller group D1\* knows that E is valid, it should put E into its hash chain.
    - To do this though, D1\* needs to wait until the event that occurred immediately before E (which also has a dependency on D) has been added to the hash chain. Let’s call the hash chain entry for this event HE-1(D1)
    - As soon as D1\* has entered HE-1(D1) into its hash chain, it should send HE-1(D1) to all the other dependents Di of E.
      * If HE-1(D1) already exists when a member d11 of D1\* receives the last candidate event Ei, and d11 receives this last candidate event after Te+1 (which is the only way d11 can be sure that the last entry in its hash chain will indeed be HE-1(D1) - the one immediately preceding E) then d11 should distribute HE-1(D1) at the same time that it sends the success message to W and E\* confirming E.
      * If d11 receives the last candidate event Ei before Te+1 (which should be common), then d11 will wait until Te+1 and, if it still has not received an event with a timestamp earlier than E’s, it will distribute its last hash chain entry, which it now knows to be HE-1(D1), to W and E\*
    - D1\* then needs to wait until it has received the last hash chain entries (the one just before E) of all the other items dependent on E, before it can create HE(D1), the hash chain entry for E, which will combine HE-1(D1), HE-1(Di) for each Di and the hash of E itself.
      * The hashes that go into creating HE(Di) should be ordered in the same way for all Di such that HE(Di) will be the same for all Di. Thus I can start calling HE(Di) just HE.
    - By the time a busy controller group gets around to putting E into its hash chain, it should already have received the last hash chain entries of all E’s dependents (unless one of them is even busier) in which case it will not have to wait on anyone to create E’s hash chain entry.
    - Each member of D1\* will sign HE with its private key and send this message to the rest of D1\*
    - If one of the dependent controller groups D1\* of E wants to verify later that another dependent D2\* of E has kept the promise it signed off on to include HE in its hash chain, D1\* simply needs to ask anonymously for D2’s hash chain and verify that the entry for E exists and is equal to the HE in D1’s hash chain.
  + Every host in W should send a signed copy of H­E to E\*. E\* will then include HE along with all the signatures from all the members of W, as properties in E
  + Once E contains all of these signatures it is considered officially valid.
* When a new host x1 joins a controller group X\* for item X, a random variable seeded by x1’s zid, X’s zid and the current timeframe key should determine how many timeframes back x1 must audit.
  + In order to audit a single hash chain entry Hf of X, x1 needs to check Hf for logical integrity, check that it is present in its companion hash chains (the hash chains of the other items Hf’s event depended on), and check that those other hash chains have been built correctly after Hf. The last step requires x1 to calculate each hash in the hash chain from the asserted inputs and verify that they were calculated correctly. It does not require x1 to check the semantic validity of the events in other items’ hash chains – that will be done by the new entrants of those other items.
  + If x1 is asked to do a level 3 audit on X, x1 must follow the above procedures on all hash chain entries of X created during the last 3 timeframes.
  + A new entrant will always be required to audit at least the last timeframe and sometimes many more
  + x1 must prove that it has completed its audit by creating a new Audit item with an association connecting it to X, The audit item should contain:
    - All the hash chain entries of X in the timeframes that x1 was asked to audit
    - For each of these hash chain entries Hi, all the hash chain entries in the required timeframes of each of Hi’s dependencies.
      * The owners of each of these sets of hash chain entries needs to have signed them, along with the current timeframe key to prove that they signed it this timeframe.
      * In order to make things more efficient, at the start of each timeframe, each controller group D\* of dependency item D should sign all the hash chain entries of D for the last timeframe, along with this timeframe’s key. This block of hash chain entries will be distributed to all the auditors who come asking for it. The auditors will ask for it anonymously and may get it from an intermediate node which has cached it without needing to get a message all the way to D\*.
    - x1’s signature on all of the above, along with this timeframe’s key
* Let’s revisit, now, the potential for double spending
  + Let A be an attacker who wishes to pull off an exploit whereby he purchases $1 million product without losing the $1 million
  + Let V be the vendor A will purchase the $1 million product from
  + Let K be a co-conspirator of A
  + Assume that for timeframe TF1, A has managed to gain control of every member of A\* and every member of K\*
  + At time TK, which is in TF1, A sends $1 million to K
    - This is event EK and creates the hash chain entry HK which all members of K\* and A\* must sign in order for EK to be valid
    - A\* creates and signs HK and gives the signatures to K\*, but does not include HK in its hash chain
  + At time TV which is after TK, but still in TF1, A sends a purchase request to V for the $1 million product, and sends V $1 million. This is event EV and creates the hash chain entry HV.
  + A\* creates and signs HV, but appends it to a hash chain that does not include HK.
  + At this point, V and V\* have no way of knowing that A\* is being fraudulent because A\* is the only common link between events EK and EV and it is solely A\*’s responsibly to identify and block this conflict.
  + So V\* accepts EV and adds $1 million to V’s account
  + If V immediately gave A the $1 million product, A would have successfully pulled off a double-spend attack – this is assuming no intervention by the Zutha community after they discover the exploit. I’ll get back to this possibility.
  + V would be unwise to accept A’s payment, especially since it is so large, before a few timeframes have passed. Actually, I expect that timeframes will be short enough (maybe in the order of 30s to 5mins), that it would be unlikely V would even be able to give up control of something worth $1 million in that time.
  + So now let’s consider what happens at the start of timeframe TF2.
    - K\* will gain a new random member k1.
    - A\* will gain a new random member a1.
    - V\* will gain a new random member v1.
    - k1 will perform an audit on the hash chain entries that were added to K in TF1. k1 will find HK in K’s hash chain and examine its dependencies to find A. k1 will request A’s hash chain entries for TF1 anonymously and will very likely be given them by an intermediate node which has cached them.
    - V\*’s new member v1 will also ask for A’s hash chain block for TF1, which it will also likely receive from an intermediate node which cached it.
    - A\* will not be able to send different hash chain blocks for TF1 to different hosts. It therefore has the following three options:
      * Omit HK from the hash chain in which case k1 will find it missing and report it.
      * Include both HK and HV in the hash chain in which case it must publish a hash chain which is cryptographically invalid (for example by inserting HK without using it as input to the hash chain entry that follows). In this case a1, k1 and v1 will all discover the error.
      * Include both Ek and EV in its hash chain and recalculate all the hashes to make the hash chain valid. In this case, v1 will find an entry in A’s hash chain claiming to be for EV, but its hash will not match the hash HV in V’s hash chain so v1 will discover the fraud.
      * Omit HV from the hash chain. This will require changing the hashes for any hash chain entry that was added after HV, which will tip off auditors that come to examine those. Even if HV would have been the last hash chain entry of TF1, omitting it will still tip off v1.
    - So k1 or v1 at a minimum will discover A’s fraud.
  + So the only way A can sustain the fraud into TF2 is if A continues to omit Hk from its hash chain and manages to take control of k1 before it commence its audit
  + What if K\* neglects to tell k that it is has been selected as a member of K\*? It is K\*’s responsibility to do this and k cannot find out by itself because the selection function is one way: from the item ZID to host id and not the other way around.
    - K\* can only sustain keeping k in the dark until someone makes a request of K
    - It looks like there is no way to guarantee that k will find out about its membership of K\* immediately if K\* is controlled by an attacker and acts concertedly keep k in the dark. The reason is that any deterministic rule that selects n additional hosts to take responsibility for notifying k of its role adds no more security than simply making K\* n members bigger. If an attacker can gain control of a randomly selected group of hosts K\*, there is nothing intrinsically different about the attacker gaining control of a broader group of hosts that includes the other hosts that might interfere with the attacker’s plans.
    - If any other events occurred in TF1 that effected K, then there will be auditors (other than a1) which will be asking K\* for K’s hash chain. These requests will reach k1 because using the selection algorithm on K’s ZID and the new timeframe key will select k1 by definition. k1 will then commence auditing K. k1 should also realise that K\* has misbehaved by keeping it in the dark because k1 should have been notified and prepared for its membership in K\* in TF1, when the timeframe key for TF2 was first known, but not yet active.
  + Another vector of attack for A would be to include both HK and HV in its hash chain right from the start and hope that no one notices that they are logically incompatible
    - a1 will discover this fraud as soon as it does its audit of the hash chain entries for TF1
    - A\* might not tell a1 about its membership of A\* though
    - As soon as an auditor like k1 or v1 asks for A’s hash chain, though, a1 will be notified of its membership and, just as with k1 as described above, a1 will know A\* has been deceptive and can report its members.
  + For a transaction of $1 million, V (the user) would be foolish if he didn’t perform a thorough audit of A himself before accepting the transaction as valid.
    - V’s client should provide a tool to perform an audit to a requested level of depth.
    - At a minimum, such an audit would involve examining A’s transaction history (a subset of A’s hash chain of events) to verify that the last transaction left A’s new account balance at greater than $1 million.
    - Note that all events should include the old and new value of any properties they modify, so just examining the transaction event in A before EV (that is, EK if A decides to keep it in its hash chain) will reveal that the “new value” field in Ek is less than $1 million.
  + If V wants to be really careful, he can wait a few more timeframes until it is practically guaranteed that, if A (and A\* and K\*) is indeed fraudulent, at least one of the many auditors to examine A over those timeframes will have discovered the fraud.
    - Waiting more timeframes also increases the chance that those auditors will discover their roles because it will allow other auditors to notify the uninformed auditors of their roles.
    - For example, even if A has control of A\* and K\* and refuses to tell a1 and k1 or their memberships of A\* and K\* respectively, in TF2, v1 will notify a1 of its role when v1 asks A for its hash chain. a1 will then immediately commence its audit of A, causing it to ask K for its hash chain, which will in turn notify k1 of its membership of K\*. k1 will then audit K, leading it to ask for A’s hash chain, at which point it will discover A’s fraud (if it has omitted HK from its hash chain).
    - As soon as a1 receives a request for A’s hash chain from v1, a1 will know that something is fishy and will notify v1. V will then become aware that potential fraud is at foot (via v1 and V\*) and will know to wait until it is resolved. That is, V will wait for the propagation of enlightenment described above where auditors notify each other of their responsibilities, until it reaches the auditor with access to the proof of fraud.
    - In rare circumstances, it might be that an auditor (a newcomer to a controller group) fails to be informed on time of its role and triggers a fraud alert that turns out to be a false alarm caused by extreme network latency or something.
    - If such a fraud alert occurred on A’s payment of $1 million to V, it would be understandable if V waited many timeframes and conducted extensive auditing before transferring the $1 million product to A.

# Sat 29-Oct-11

* When a host in a dependency group D\* on candidate event E adds HE-1(D) to its hash chain, it should send not just HE-1(D) to the rest of E’s dependencies, but all the properties and associations of their item D that E depends on and a sufficient block at the end of their hash chain to include the last events that effected each of these properties and associations.
  + The other dependency groups should then use this information to verify E’s semantic validity globally, rather than just from the perspective of their own item
  + In the case of A’s double-spending attempt described above, A\* would not be able to gain the signatures of the other dependents on HE if it included EK before if in its hash chain. This is because the other dependents on E would verify E with respect to A and its hash chain and discover the conflict with EK, which was the last event to effect A’s account balance and must therefore have been included in the block of hash chain entries A distributed to W (the dependents on E). The other dependents will therefor refuse to accept A’s HE-1(A) and will report A for fraud (because it is very unlikely that every member of a controller group would simultaneously make the same simple logic error accidentally.)
* Distributed querying
  + Consider a query for American companies that have a Chinese CEO (this is an example I use often to illustrate the limitations of Google)
  + The syntax for this query might look like: “type:Company based-in(country:America) has-CEO(CEO:(nationality:Chinese))”
  + There are two fundamental paradigms (that I can think of) that could be used to perform this query on Zutha’s distributed knowledgebase
    - Clients could retrieve all the necessary sets of items required to perform the query and implement the logic themselves
    - Clients could delegate the retrieval of the required items and the processing of the query to a request handler group and receive a simple answer from them containing the desired items.
  + I think both of these paradigms should be options – in fact, if the second option is available (delegating the whole query to a request handler group) then the first will be as well.
  + So let’s just consider how the delegation to a request handler group would work. A client could easily perform exactly the same task itself.
    - Let Q be the set of request handlers that are selected to respond to this request
    - Q receives a request R from U containing the above query
    - U also specifies a set of fields F that it wants included in the results
    - Each member of Q will parse the query to identify the most efficient way to retrieve and process the required items
    - Most hosts will very likely stay up-to-date with a cache of common items (especially types) and association types, so they will probably already know about “Company”, the “type” role, the “based-in” and “has-CEO” association types and the “nationality”, “CEO” and “country” roles. Most hosts will probably also have “America” and “Chinese” in their cache.
    - If a host is missing any of these from their cache (or need to bring their version up-to-date), they will have to retrieve those items by name from the distributed index before they can complete the query.
    - Actually, hosts that don’t know about, say, Chinese, could just use the string “Chinese” in their query to the “nationality” role when they ask it to send back all items that play the role nationality. Actually it might be unrealistic (at least in some cases) to expect controller groups of Role items to keep track of anything but the ZID of the items that play their Role. I’ll consider this problem more later, particularly with regard to very common Roles like Instance.
    - Let’s assume that member q of Q knows all the items named in the above query - that is: “type”, “Company”, “based-in”, “country”, “America”, “has-CEO”, “CEO”, “nationality” and “Chinese”.
    - The queries q might dispatch could be:
      * $chinese = “Nationality:Chinese” (substituting the ZIDs for both words) to the Chinese controller group
      * $companies = “type:Company” to the Company controller group. The following fields would be asked for in the results:
        + organisation/based-in
        + Company/has-CEO
        + The set of fields F requested by U
    - q would then create (if it has not already) indexes from Countries to Companies via the based-in association and People to Companies via the has-CEO association
      * Note that just building a topic map from the $companies data is equivalent to creating such indexes
    - q can now convert U’s initial query into one in terms of the data q already has
      * $results = organisiation:based-in(country:America) company:has-CEO(CEO:$chinese)
      * Note that q has inferred some additional components of this query, such as that the only role an item of type Company can play in a based-in association is organisation and the only role a Company can play in the has-CEO association is Company.
    - q will now process $results to include just the fields F requested by U and send this data back to U
* The Instance Role is another of those items (like I thought Name would have to be) that will be under extreme load
  + Every single item will play the role Instance – even Event items - so the controller group for Instance would need to maintain a database of every single item and event!
  + Maybe I can avoid requiring Role items (like Instance) to keep track of all the items that play them.
  + Actually, I was mistaken to think there would be a reason for Instance to keep track of all instances in the first place. I had been thinking that Instance would be part of associations which involve the Role Instance, but Instance is only part of the schema associations that declare which Association Types have an Instance Role and which Item Types can play the role Instance.
  + I had also been thinking that some queries would require asking a Role item’s controller group for the answer, but now that I consider this again, I realise I was wrong
    - I had thought that queries like Nationality:Chinese should be asked of the Nationality Role item, but such a query should be asked of the Chinese item since it will know which associations it plays the role Nationality in.
    - You cannot make a query such as Instance:\* (at least I’m deciding now that you can’t) which has only a Role item as an anchor to identify matches.
    - If you want to find all items that play a particular Role you can do so by finding all the Types that declare that their instances can play that Role, then finding all the instances of those types, and then filtering these results by the ones that play the Role in question.
    - So let’s say the syntax Instance:\* was permitted.
      * The query would be performed by finding the Types that can play the role Instance. In this case, that is only Item.
      * Then find all instances of Item. This would be done by finding all descendants of Item and asking each of them for their instances.
      * All instances of Item would then be examined one by one to find out which ones play the Role Instance. This last step at least could be skipped in this case because the schema specifies that all Items must play the Role Instance exactly once.
      * So finding Instance:\* would be a lot of work, but then, it’s a pretty pointless query to make.
  + There will surely be some items with substantially more workload than others, but I’m beginning to feel more confident that game-stoppers like Name (and I had thought, Instance) will be rare enough that the specific cases (like Name) can be worked around.
  + How convenient that Type-Instance will not be reified because that avoids the Type-Instance association type item becoming another choke-point
  + Supertype-Subtype and Item-Has-Trait will both be reified so both of those Association Type items will have high workload, though nothing like Type-Instance would, I expect.
    - Supertype-Subtype is only used by types and the type system is not likely to change extremely rapidly – in fact, Supertype-Subtype is likely to be far from among the most used associations.
    - Item-Has-Trait will probably be more common than Supertype-Subtype, but I think its workload will be reasonable because the vast majority of items should have just a single type
  + Possibly the highest workload items will be the Event Type items
    - The Create Item event, for example, will have a Type-Instance association added to it every time a new Item is created
    - One way to reduce workload on Event Type items is to make the Event Types more granular. So the Create Item Event could be broken into subtypes; perhaps one for each Item Type. Whenever an item of type X is created, then, only X and the “Create X” Event Type items would need to know about it. Of course, the Type items of the various reified properties and associations would also be involved.
  + Edit Property Events could also be divided up into specialised events for each Property Type
  + Every Event Type, in fact, can be granulized by type of the construct effected
* Compound Events
  + It probably only makes sense to have a single Compound Event type because there are an arbitrary number of ways that many events might be combined into a single transaction
  + Thus the Compound Event item will need to have Type-Instance associations added to it every time a Compound Event occurs, which I think will be very frequently
  + Every item creation, for example, will be a compound event because the creation of all reified properties and associations will be individual events.
  + Compound Events could potentially be granulized by User
    - The main problem with this approach is that it is not really semantically appropriate – but neither is having a different Name type for different prefixes I guess.
    - This might be the best solution, though
  + So whenever a User U performs a compound action (like creating an Item), the compound event created will be of type “U Compound Event” or maybe “U Action”.
    - This event will therefore have a type-instance association with “U Action”.
    - It should also have an association linking it to U itself (via, say, an “Actions Performed” association) and should be linked to all the granular events that make up the Action
  + Compound Event should be renamed Action
    - A User name “Chris Barnett” performs actions of type “Chris Barnett Action”, which is a subtype of “User Action”
    - An Action consists of one or more Events
* Events should all be granular, meaning there are no compound event types which consist of more granular events (this is a model I have considered before)
  + So a Create Item event consists of the minimum information required to describe the creation of an Item. That would include the type of the item and its ZID. That’s about it.
    - Type and ZID are chosen because they are “system properties” which cannot be modified and are generated automatically as part of the item creation process itself
    - There may end up being other kinds of system properties that are generated at item creation time
    - Date Created and Date Modified properties would also fall under this category actually
    - Auto-Description properties would, as well, because they are auto-properties which are actually stored because they can later be edited. Auto-Names are calculated when requested so it would be redundant information if included with the Create Item event.
    - Actually, Auto-Description can be calculated when it is first requested as well
    - So it looks like it is System Properties (ones that are calculated by the system but which are not inferable from other data) rather than Auto-Properties that need to be included in the Item Creation event
  + Other granular events would include:
    - <AT> Role Player Addition (specialized to Association Type)
    - <AT> Role Player Removal (specialized to Association Type)
    - <AT> Association Creation (specialized to Association Type)
    - <AT> Association Removal (specialized to Association Type)
    - <PT> Property Creation (specialized to Property Type)
    - <PT> Property Removal (specialized to Property Type)
    - <PT> Property Modification (specialized to Property Type)
    - <Item> Investment (specialized to Item)
    - <Item> Investment Sale (specialized to Item)
    - <user> Transaction (specialized to the paying user)
  + The names of these specialized event types should be auto-names generated by the current name of the item indicated between angular brackets (<AT>/<PT>/<Item>) and the current name of their supertype (like Role Player Addition)
* What might the Zutha community do if a successful double spending attack is discovered?
  + It should be pretty easy in most cases to figure out which users would have benefited if the attack succeeded even if the common controller of all the hosts that collaborated on the attack cannot be identified
  + The Zutha community might decide to punish the users that benefit from the attack, assuming that they were involved in the conspiracy somehow
  + This could be a dangerous approach, though, because it would enable a victim to be framed by an attacker so that the victim will be punished unfairly by the Zutha community once the fraud (actually done by the attacker) is discovered.
  + There should be many cases, however, where certain users involved in an exploit attempt can be very confidently implicated in the exploit attempt.
  + In the case of the double-spending attack described above for example, A is obviously involved in the attack because A makes the two conflicting payments and it is implausibly unlikely that A just happened to make these two conflicting payments just at the time that two controller groups (A\* and K\*) collaborated to conceal them.
  + K cannot necessarily be proven to be involved in this attack, however, because K does nothing out of the ordinary – in fact K takes no action whatsoever; he just receives a payment of $1 million from A
  + As soon as A\*’s fraud is discovered in this case, all transactions that depend on either of the conflicting transactions should be frozen.
    - This means K, A and V will not be able to make any transactions until the fraud is resolved
  + Actually, all events that depend on any dependencies of the conflicting events should be disallowed.
    - This means that no event will be permitted that involves K, A or V
    - Hmm… this would prevent any of these users from even creating comments. This would be unacceptable as they need to be able to communicate with the rest of the Zutha community to resolve the fraud.
    - I’m thinking it might be better to decouple different events involving the same item that don’t depend on each other. I’ll return to this idea later.
  + All hosts that signed conflicting hash chain entries will immediately be evicted from the network so that in the next timeframe, A\* will be replenished with entirely new members.
    - a1 (the new member of A\*), which signed the hash chain containing EV and not EK (which turns out to be the correct event to have included), will not be punished for having signed EV, even if a1 later has to sign EK. This logic will be implemented by discounting (in the future), all the hash chain entries that occurred between the conflict and the resolution of the conflict.
  + K\* members have not produced any evidence of crime unambiguous enough for other hosts to automatically evict them (as is the case with A\*), but there is enough circumstantial evidence that human judges will probably vote to evict them all, or at least drastically reduce their reputation.
  + The replenished controller groups A\* (and probably K\*) will then recreate their respective hash chains with the correct semantics, which means they will include EK (which came first), but not EV. V\* will need to recreate its hash chain to omit EV which has now been found to have been invalid.
    - In order to do this, the new members of A\*/K\*/V\* will have to sign hash chain entries for events that occurred before they were members of A\*/K\*/V\*. This will need to be justified in the historical record with evidence of the fraud of the old A\*
  + Unless V acted very unintelligently and sent the $1 million product to A already, it shouldn’t be harmed by having received $1 million for a short time.
  + There is enough evidence that A was involved in the attack though, that A should be punished severely and probably evicted from the network altogether, forfeiting all his money and reputation.
  + K is very suspicious because A chose to send his $1 million to K to keep it from V. It is possible that K is another authentic vendor of $1 million goods, however, in which case it does not deserve to be punished.
    - K should have to prove that it is such a vendor (and that the $1 million was for the purchase of a product) or give some other very good reason for having received the $1 million for it to remain out of suspicion.
    - If K can only claim that it doesn’t know why A sent it $1 million then it will remain under suspicion, but still cannot be punished immediately.
    - This is where there would be need for vigilante hackers (and eventually formal private law enforcement agencies working for Zutha) to investigate A and K to look for evidence of a conspiracy.
  + All of the above procedures need to exist to account for the possibility that someone takes control of all of A\* and K\* and pulls off a double spending attack for a timeframe or two, but achieving this deception for one timeframe would require resources of incredible proportions (to take control of two whole randomly selected and evolving controller groups) and maintaining the deception for one more timeframe would require the ability to intercept a randomly selected new member of V (v1) before it performs its audit and notifies a1, or a1 before it notifies k1 or k1 itself. It would also require a lot of luck, since as soon as someone makes a transaction with K, the absence of EK from A will be discovered.
* There should be a policy of random inspections of items that are selected in the other direction (from Host id to item id) so that there is no way for an attacker to block them or even know of the inspection. These inspections would have to be unpredictable though, because there is no way to design a selection algorithm based on hashes that will guarantee that every item is selected.
  + Some kinds of items and some event types could potentially be made to receive more random exposure to audits
  + For example, to implement a User audit, each host could be required each timeframe to enter their id and the current timeframe key into the selection algorithm to select among all users. Whichever user is selected, the host would inspect all the actions performed by that user in the last timeframe
  + Random inspections like these should have a powerful dissuasive effect against attempts at fraud because the attempts would become fundamentally unpredictable and the cost of failing would be very large (such as A losing its $1 million)
* What if any event is invalidated if any of the dependent controller groups unanimously contradict it?
  + So if A\* signs EK and then omits it from its hash chain in order to include EV then EK will be rendered invalid.
  + Actually, all the actions signed by that controller group from the invalidated event onwards should be considered invalid
  + This way, if A\* unanimously signs both EK and EV then it loses the authority to approve any other events involving A
  + If A tried to send his only $1 million to both K and V then, both transactions would fail (if A\* allowed both) and the $1 million would remain in A’s account. A would not be able to use it though, because A\* will have lost its authority to approve A’s transactions.
  + Until A\* is actually caught, though, another potential receiver of A’s money, J, will not know that A\* has no authority to transfer them money. However, In order for A\* to convince J that A is able to pay it money, it must omit both EV and EK from its hash chain. This leaves J in essentially the same position that V was in, except that the attacker now has to have control of both V and K now in order to prevent them sounding the alert when they discover that A\* has removed EV/EK from its hash chain.
  + This rule also eliminates the requirement that the new members of A\* sign events that occurred before their arrival in A\* because all the events approved by the old A\* will simply be rendered invalid and do not need to be re-added to the fixed hash chain.
  + Now, when A\*’s treachery is discovered A will be found implicated and probably punished with loss of his account which means he will lose the $1 million still frozen in it.
* Couldn’t an attacker take control of a past controller group and invalidate an event they signed back then?
  + Potentially, a time limit could be placed on how soon a controller group contradiction has to be seen and resolved for events to be invalidated, but this seems pretty messy to me.
  + I think this means that invalidating events because a controller group later contradicts them will not work
* Perhaps hash chains should be event-centric rather than item-centric
  + Instead of a hash chain, every event would have a hash which combines the hashes of all the events this event depended on
  + The difficulty with this idea is that it would be difficult to define all the events that an event depends on. Does every Create Property event, for example, depend on the event that created its Property Type and the event that created the schema association type allowing the item it will be associated with to have Properties of that type? What about the event that last changed the cardinality of that property type?
  + Also, in this system, who is responsible for identifying which recent events might conflict with a proposed event?
* Intermediate nodes should see A\*’s conflicting messages
  + For EK to be valid, A\* (and K\*) must send a signed copy of HK to EK\*
  + When A\* sends EK\* its signed copy of HK, lots of intermediate nodes will receive this, including most of the nodes nearby each member of A\*
  + Before V\* will accept A’s transaction, it will need to see A\*’s signatures on HV, which will be collected by EV\*.
  + When A\* sends EV\* its signed copy of HV, many of the same intermediate nodes that saw its signed copy of HK will now see HV.
  + I was thinking along the lines that these intermediate nodes would be able to identify the incompatibility of HK and HV, but they would only be able to do this if they knew all the hash chain entries of A from HK-1 to HV.
* Some additional degree of event-centrality for hash chains might be possible and desirable
  + What I really mean is that hash chains should be more specific about the dependency lines they create
  + The current approach of keeping all events that effect an item in one hash chain even when adjacent entries may not be related, is not ideal
  + It would be much better if an event can be validated simply by examining the set of events it depends on.
  + If all the events that effect an item are bundled into the same hash chain, then, in order to validate an event E, one must search back through the hash chain looking for past events that E depends on. This is quite impractical because the hash chain consists only of hashes and ZIDs, so each event in the chain before E has to be individually looked up from its controller group.
  + Now that I am thinking in these terms, I realise that, even if A\* included both EK and EV in its hash chain, it would be very difficult for V\* to find that EV has a conflict earlier in A’s hash chain because there may be many events between EK and EV in A’s hash chain.
  + Each item could have separate hash chains for each logical element that is applicable to that item
  + So a Property Type would have a hash chain for its cardinality for each Item Type that declares it as well as separate hash chains for its name property and supertypes association field set etc.
  + There would be two different kinds of relationship between an event E1 and an event E2 that depends on it
    - E2 could supersede E1 as in the case of E1 and E2 being transactions involving the same item X. So E2 becomes the new holder of X’s current account balance. Once E2 supersedes E1, E1 should not be used as a dependency in any subsequent events.
    - E2 could require E1 but not supersede it. For example the creation of a Birth Date property (event E2) depends on the event that last changed the cardinality of the Birth Date property (event E1) but E1 may still be required as a dependency in later creations of Birth Date properties.
  + The distinction between a superseding event and dependent event will have to be Item-specific
    - Consider a transaction E1 which transfers money from user A to user B and a transaction E2 which transfers money from user B to user C
    - From B’s perspective, E2 supersedes E1 because E2 becomes the last event to update B’s account balance, but from A’s perspective, E1 is not superseded yet because E1 is still the last event to have updated A’s account balance
  + So I think hash chains need to be fundamentally item-specific.
* A new proposal for hash chains and the distributed clock
  + Let E1 be a new event that was created at T1
  + Let D be the set of dependent items on E and D\* be the set of controller groups of the set of items D (yes I’m changing terminology)
  + Requirements
    - A dependent item’s controller group D1\* must send a public message showing that it has appended E1 to one of its hash chains, superseding the previous head of that hash chain
    - It should be possible to validate E1 by downloading it from E1\* (or from someone’s cache) and then downloading and examining each of its dependent events
    - It should be possible to easily verify that an event is still present in an item’s hash chain (whichever one it was appended to)
    - All the hash chain heads created by E1 must be cryptographically linked such that if any of them are changed later, they will not agree with the others
      * A hash chain head is the current active entry in the chain. All the entries below the head in a chain have been superseded.
  + Once more than half of each Di\* in D\* have received E1 within one timeframe of T1 and sent a signed success message to E1\*, E1 is chronologically valid
  + At time T2 (one timeframe after T1), at least half of each Di\* in D\* will know (correctly) whether there is going to a conflict at their item with E1
    - There is a very slim chance that some of Di\* (but less than half) will get to time T2 without having received an event which will succeed that conflicts with E1.
  + If, at time T2, D1.1 does not know of a conflict with E1, D1.1 will send the hash chain heads of D1 that E1 depends on to E1\* and D\*.
    - For each hash chain head it lists, it will specify whether E1 supersedes it or merely depends on it
    - D1.1 should also send the full event associated with each hash chain head it lists. The event will contain information (such as signatures, hashes and semantic information) that other hosts need in order to verify the correctness of D1.1’s set of dependent hash chain heads.
    - By signing this message, D1.1 makes a promise to append E1 to D1.H1 after D1.H1E1-1 if E1 succeeds
  + Once D1.1 has received the dependent hash chain heads of all the other dependencies D (along with their associated events), D1.1 should check them all for validity
    - This involves checking that the state of the world implied by all the events D1.1 has collected makes event E1 valid.
    - For each dependent event E1d D1.1 should also, at a minimum, verify that the most recent member to join E1d\* has signed off on it indicating it is valid
    - If E1 is a very sensitive event (such as a large cash transaction) then D1.1 should also do a more thorough check of each E1d, examining its history and original signatories to verify they were the right signatories, and possibly examining all the newcomers that have signed off on E1d since it was created.
  + Once D1.1 is confident that E1 is semantically valid and that its dependent events are also valid, it needs to keep its promise to append E1 to the hash chain it declared E1 will supersede.
    - So D1.1 will put all the dependent hash chain heads into E in a standardized way and hash the result, yielding H(E1)
    - D1.1 will append H(E1) to the D1.H1 hash chain, yielding entry D1.H1E1
    - D1.1 will sign H(E1) and send the signature to E1\*.
    - D1.1 officially appends H(E1) to its hash chain just by signing it and sending it to E1\*. It should keep its own internal hash chain only for the sake of convenience. Every internal hash chain is redundantly implied by the events it is composed of.
  + Every member of D1 (and all the other controller groups in D) need to sign H(E1) and send the signature to E1\*, even if that member didn’t receive E1 by time T2
  + The signatures of all the members of D\* will then be stored permanently with E1
  + Once E1 includes at least half the signatures of each group in D, E1 is a confirmed event
* Every Hash Chain should have a unique identifier
  + A hash chain won’t be an item so its identifier should not be a ZID
  + A hash chain identifier should be a sub-address of the item’s ZID such as 14FE0.05TR, where 05TR is the ZID of the item that characterizes the dependency the hash chain encodes
  + For example if 14FE0 is a property type that is declared by item type 05TR, then 14FE0.05TR would be hash chain that is extended whenever an event occurs which effects the semantics of creating 14FE0 properties for 05TR items. Such an event would be a change in the cardinality of 05TR’s declaration of property type 14FE0 because that would effect when 14FE0 properties can be created for 05TR items.
* Hash Chain entries should be numerically indexed with a simple counter starting from 0 for the very first entry
* Every node N in the network should keep a record (an Event Auditing Map) of the events that it has seen in the last few timeframes
  + N should create a map from Hash Chain Entries (addressed as <itemZID>.<hashChainID>.<index>) to the corresponding event
  + When N encounters a new event E, it should examine the event to find the hash chain entries that are marked as superseded by E. This will tell N the hash chains that E is a part of, as well as the index E should have in each of these hash chains.
  + For example, if E states that it supersedes hash chain entry 0FD6.04FE.7 then E needs to be added to N’s Event Auditing Map at key 0FD6.04FE.8.
  + Before adding E to its map, N should look for conflicts between E and previous events it has added to its map. A conflict would occur if
    - A key that E should occupy (like 0FD6.04FE.8) has already been taken by another event
    - The hash that E uses to refer to 0FD6.04FE.7 (which it says it supersedes) differs from the hash N has associated with 0FD6.04FE.7 in its map.
  + N should also associate every event it sees with the signatures it sees for that event
    - An event’s signatures from each involved host will arrive in different messages because hosts send their signatures for event E to E\* as soon as they have received all E’s dependent events.
* Events should have a link to the events that supersede them (there could be multiple because events supersede events with respect to a specific item)
  + When hosts send their signed copy of a new event E to E\*, they should also include in the address of this message, the controller groups of events E supersedes.
  + When the controller group Es\* of a superseded event Es receives notification of a new event E that will supersede it with respect to item X, Es\* will create a non-reified association indicating this which has Es, X and E as roles. X and E will have independently created this same association.
  + If Es has already been superseded with respect to X, though, Es\* will send proof of this to E\*
    - When a conflict like this is discovered, the events involved will be frozen such that any events that would depend on them will not be permitted. To the extent possible, the conflict will be resolved automatically, probably by cancelling E and evicting the hosts (at least those in X\*) that signed contradictory messages.
    - The exploit attempt will also be reported prominently so that the Zutha community can be involved in implementing additional discretionary punishments and clearing up anything that might have been missed by the automated conflict resolution system.
  + If Es finds no conflict with E it will send back a confirmation message to E\* containing Es’s signature on E
* There are now quite a few stages to complete before an event reaches official acceptance, however I don’t think this is really a problem. For the vast majority of events, clients and hosts will accept them as soon as they are first seen and no harm will come of it.
  + A new event should have an associated confidence level which increases as it progresses through the various stages of validation. This confidence level should be something implemented by clients to help users, though; there should be no official formula for calculating the confidence in an event’s validity.
  + Clients should permit users to view the current stage of validation of events if they chose to
* A proposed structure for a hash chain

[

{“HCE”:”14FE0.05TR.0”,

“event”:”2E7TRV”,

},

{“HCE”:”14FE0.05TR.1”,

“event”:”2F5TEE0”,

­},

…

]

* Proposed structure for an Event
  + Note that this structure has some redundant information. This is to reduce the number of different items that would need to be looked up and the number of hashes that would need to be calculated to do thorough auditing.

{“body”:

{“ZID”:2F5TEE0”,

“type”:”056E”,

“timestamp”:2011-10-29T19:16:25.2387313,

“request”:”…”,

“dependencies”:[

{“event”:”29FEFP0”,

“HCE”:“0XE03.15FEU.2”, //HCE = Hash Chain Entry

“hash”:”7c52d…”

}

],

“supersedes”:[

{“event”:” 2E7TRV”,

“HCE”:“ 14FE0.05TR.0”,

“hash”:”a1bed6…”

},

{“event”:”18FEQW7”,

“HCE”:“ 19FEN.13DE8.56”,

“hash”:”8ade35…”

}

]

},

“hash”:”e8ae…”,

“signatures”:[{“publicKey”:\_,”signature”:\_},…]

}

* So what I have ended up with is exactly the event-centric approach dependency tracking that I thought would be impractical.
  + The biggest problem I had seen with the event-centric approach was how to find all the other events an event depends on
  + Well, that’s the problem I’ve solved by routing dependency-finding responsibility via dependent items
* Let’s re-examine the scenario of an attacker A trying to double spend $1 million with K and V
  + Let Ea be the last transaction event involving A
  + Let A.Ht be A’s hash chain for transactions
  + Let A.Ht.12 be the hash chain entry in A for event Ea
  + Let Ek be the last transaction event involving K
  + Let Eak be A’s payment of $1 million to K (which is the last $1 million in A’s account)
  + When A first receives Eak from its request handler group, it will wait one timeframe to make sure no transactions occur between Ea and Eak and then send the dependency message {“type”:”supersedes”,”HCE”:”A.Ht.12”, “event”:”Ea”} to Eak\* and K\*
  + Once Eak is complete, it will contain both Ea and Ek in its supersedes field
  + Note that Eak supersedes Ea only with respect to A. Ea is not superseded from the perspective of A’s transaction partner for transaction Ea. This is encoded by the fact Eak’s “supersedes.HCE” field mentions a hash chain in A.
  + In Eak’s supersedes field, Ea’s ZID appears alongside its associated hash chain entry key for A: A.Ht.12
  + At least half of K\* and half of A\* will have to sign Eak for it to be valid
  + Eak will be sent (signed) to Eak\*, Ea\* and Ek\*
  + Ea\* and Ek\* will both verify (by sending their signatures on Eak to Eak\*) that Eak is the first event to supersede them with respect to A/K
  + Nodes that see Eak will create an entry in their event auditing map with key A.Ht.13 and value Eak
    - The nodes know that Eak should have index 13 in hash chain A.Ht because Eak supersedes A.Ht.12
  + Eak will be considered officially valid once it contains the signatures of at least half of each of K\*, A\*, Ek\* and Ea\*
  + Now let’s consider what happens if A tries to pay V $1 million which he doesn’t have for a $1 million product
    - A’s request R to send $1 million to V is handled by a randomly selected request handler group R\*
    - It is not R\*’s responsibility to check the semantic validity of R; R\* just gives R a timestamp and ZID (creating event Eav) and forwards Eav to A\* and V\*, the controller groups of the dependent items of Eav.
    - V\* will accept Eav because they know nothing about Eak, so V\* will send the event of its last transaction Ev to A\*
    - If any member of A\* was honest it would detect that Eav conflicts with Eak because the final account balance of A asserted by Eak is 0$. However, we will assume that A has complete control over all members of A\* and causes them to pretend that Eav is valid.
    - So which transaction event should A\* mark Eav as superseding?
    - Let’s say A\* pretends that Eak never happened and send V\* event Ea as the event that Eav supersedes.
      * Ea is A’s hash chain entry A.Ht.12 so Eav will have to be A.Ht.13
      * When A\* sends the dependency message {“type”:”supersedes”,”HCE”:”A.Ht.12”, “event”:”Ea”} to V\* and Eav\*, intermediate nodes that see it could detect the fraud because they would try to put Eav (the event the dependency message is about) into slot A.Ht.13 in their Event Auditing Map only to discover that that slot is already used by Eak.
      * I may settle for requiring intermediate nodes to only examine completed events for fraud rather than the earlier exchanges of dependency messages.
      * Once V\* has received the dependency message(s) from A\* it will create the completed Eav, sign it, and send the signatures to Eav\*, Ea\* and Ev\*
      * A\* will do the same once it receives V’s dependency Ev
      * There is a very large chance that some intermediate nodes that carry V\*’s and A\*’s signatures on Eav will have seen Eak, which A\* also signed
      * Even if no intermediate nodes report the conflict between Eav and Eak, Ea\* will because Ea\* was required to sign off on Eak for Eak to be valid.
      * Ea\* will, upon seeing Eav, discover that Ea has already been superseded by Eak with respect to A and cannot be superseded with respect to A again. Thus, Ea\* will not sign off on Eav, but will report its invalidity to Eav\*
      * So in order for the attacker to even get Eav to official validation status, it would need to have control not only of A\* and Ea\*, but of all the nodes that might see both Eak and Eav. Since the attacker has no control over routing, it simply cannot guarantee this unless it controls literally every node in the network, at which point worrying about double spending attacks is superfluous.
    - What if the attacker tries to create a fake transaction event between Eak and Eav?
      * To see whether an attacker could do this, let’s examine what measures are taken to verify the event Eav claims to supersede
      * Let Ef be a forged event that A\* places between Eak and Eav in its hash chain
      * The hosts in A\* will send the dependency message {“type”:”supersedes”,”HCE”:”A.Ht.13”, “event”:”Ef”} to V\* and Eav\*
      * Each member of V\* and Eav\* will validate Ef by, at a minimum, checking that the most recent newcomer Ef1 to Ef\* has signed off on it.
        + Ef1 is found by using the selection algorithm with Ef’s zid and the last timeframe key as inputs
        + In order to have control of Ef1’s public key to sign Ef, the attacker would have to engineer a ZID for Ef within a single timeframe, that selects a host it controls. This would be a lot easier than obtaining control of an entire controller group that the attacker had not control over selecting, so we will assume that the attacker can select a ZID for Ef that picks a host it owns for the current timeframe.
      * It seems that V\* and Eav\* need to do a more thorough audit of Ef then simply trusting the single most recent newcomer to its controller group
        + Well first of all, I just realised that more than one newcomer will need to join Ef\* in the timeframe of the attack because Ef is ostensibly a new event created this timeframe, just after Eak.
        + At least some members of V\* and Eav\* should be randomly selected to do more thorough audits of Ef.
        + The more thorough auditors should check the signatures of several of the current and past members of Ef\* (in this particular case there will be no past members of Ef\* though). They should also check the signatures of the dependent controller groups of Ef that were required to sign off on Ef for Ef to be created.
        + To fend off these audits, the attacker would have to have control of the entire controller groups of two different users (A and another one B) so that it can create a phony transaction event Ef between B and A (of $1 million) that A\* and B\* can sign off on. For Ef to look valid to auditors at first glance, though, the attacker will have to own B so that it can sign the request with a valid user. If it does this though, it needs to make sure Ef will fail; otherwise no fraud has been committed – A will have paid K $1 million and B will have paid V $1 million indirectly via A.
        + The attacker could ensure Ef will fail by pretending B has $1 million in his account when he really doesn’t. Auditors would have to traverse a second layer of events to discover that the event Ef supersedes does not leave B with the account balance Ef claims B has at the time Ef occurs.
        + If auditors will check the signatures of all of Ef\*, though, then the attacker will have to either control all of Ef\* (meaning he has to pick a ZID for Ef that will pick an entire controller group for the current timeframe that he owns), or use another layer of fake events because Ef\* will do its own checking on Ef’s dependencies if it is to accept Ef as valid.
        + To convince Ef\* to sign Ef, though, the attacker would need to present another fake event Eff before Ef which Ef\* would audit, meaning the attacker would need to either own Eff\* or use yet another fake event before that and so on.
        + Yet another complication for the attacker – it cannot simply pick any ZID since the ZID of an item is decided by the request handler that assigns the median timestamp to it. The request handlers are selected by the hash of the request, so, in order to pick an event’s ZID, the attacker has to find a message whose hash will lead to the selection of a request handler group which the attacker owns.
        + If an attacker could arrange to control A\*, Eav\*, B\* and Ef\* then it could possibly create a fake event Ef that tricks V into thinking it has received payment of $1 million… that is, provided no auditors search more than one layer of events deep.
      * It is clear that we can go on and on like this, making more expansive assumptions about how many hosts the attacker can have control over. This is no surprise. The greater the proportion of the network an attacker controls, the more damage he can do to the network in the form of fraudulent changes that require more thorough auditing to detect. In the limit of the attacker owning the entire network, he can do anything he wants.
  + So an attacker can do some damage if he can obtain control of a large number of entire controller groups, but the level (depth) of auditing can be increased to require the number of controlled controller groups for an attack to be arbitrarily large.
  + It seems one of the most important security concerns will be ensuring that the selection algorithm is cryptographically irreversible and difficult enough to perform that trial and error does not permit significant discretion in choosing which hosts it will select.
    - It would also make it harder for attackers if the range of possible nonce values the attacker has to work with, given the action he wants to perform, is as small as possible.
    - For attackers whose strength is processing power more than ownership of a large proportion of all hosts, it would help to randomly limit the set of hosts available to select from before the hash of the attacker’s request is used to narrow this restricted set down to a request handler group.
  + Actually the hardest task for an attacker will probably be to obtain control over controller groups it had absolutely no influence in selecting, especially since, in order to have much power, those controller groups must be specifically for users the attacker also controls.
  + I feel pretty confident that (with refinement) the concepts I have developed here can make Zutha virtually impregnable to attacks that try to push through invalid actions. There are still other things to worry about of course, like money laundering and the susceptibility of individual users to manipulation by attackers. But I think I’ve done a pretty good 3 days’ work.

# Sun 30-Oct-11

* I think there will inevitably be situations where it will be desirable to change the type of an item
  + Items should have an permission-level property associated with the item itself.
  + The permission level property P of item X specifies the reputation required to edit P itself, delete X or change X’s type
* Type-Instance and Item-Has-Trait associations should both be able to be scoped by time period
  + An item can thus have multiple direct types if the types are scoped by disjoint time periods
  + An item can have a type T in the global scope and be typed with multiple subtypes of T in disjoint scopes
  + All fields declared by a scoped type must have the same scope as the type that declares them
* Simple Binary Associations (associations with two roles of cardinality 1 that an item may play only once) should be displayed in the same way as properties
  + My inspiration for this design is Freebase, which rather neatly treats binary associations the same as literal properties
  + So binary associations should be displayed like
    - <associationType>: <otherPlayer>
  + Unlike other associations, the association type name in simple binary associations should not be a link to the association-table page. In fact, just as with properties, the type name should not be a link at all.
  + Users should be able to go to the association-table or role-player-list pages of a simple binary association, but only by typing the url in directly; there will be no link to these pages anywhere because they are not useful for simple binary associations.
* I quite like Freebase’s representation of properties in edit mode and I think I will emulate it for details view in Zutha
  + Properties and associations should all be combined into a single block
  + Properties and simple binary associations should look the same except for the fact that the value part of simple binary associations will be a link
  + Binary associations that have cardinality other than exactly 1 should be displayed as a table with just one column
  + It needs to be a clear when cells in a table contain multiple values. This means making the boundary between rows clear.
  + The header row of association tables should have the association type in the first column (in the same place that the property type goes) and the various roles that can be played in that association type should be listed to the left in the same row as the association type
* Actually I quite like Freebase’s representation of properties in summary mode as well and I’m strongly considering copying every aspect of both the Summary and Edit modes including the visual styles

# Mon 31-Oct-11

* Binary Associations should be displayed as a simple vertical list (like Freebase) without the other role being displayed.
  + The “view entire collection” link should link to link to the role-player-list page.
    - This link should be actually be called “View all <otherRole>s”, for example: “View all Types”
    - Using this naming scheme for the link has the added advantage that it provides the name of the other role, which otherwise would not have been shown anywhere.
  + The “view entire collection” link for multi-role association fields should still link to the association-table page. The role links should be used to get to the role-player-list page for association tables
  + The Association-table page should of binary associations should still be accessible by directly typing in its URL, but it will be a boring one column table which won’t be very useful, which is why there won’t be any links to it.
* The Role Player List page should have an Edit button
  + Edit mode will allow members of the Role Player List to be deleted
  + For associations which have properties or more than two roles, the Role Player List page will allow editing of those other roles and properties in edit mode

# Wed 2-Nov-11

* User Identification
  + Users may have multiple Digital Identities
  + A Digital Identity will usually be associated with a single form of authentication
    - A Digital Identity will only be associated with more than one form of authentication of those means are fundamentally connected. For example, multiple biometric identifiers might be combined because, together, they more reliably identify a specific human being.
    - A public key, in contrast, is not fundamentally related to another public key (even if owned at a specific time by a single individual).
  + Every action performed by a user will be associated with one or more Digital Identities
  + Some methods of authentication will be verifiable by anyone who examines the event. Let’s call such methods Strong Authentication.
  + Examples of Strong Authentication
    - Public key cryptography
  + Other methods of authentication can only be verified at the time of a request by the host that executes the request. Let’s call such methods Weak Authentication.
  + Examples of Weak Authentication
    - OpenID
    - Biometrics
    - Password
    - Ownership of an email address
  + Strong Authentication can be used by clients to directly interact with the Zutha network
  + Weak Authentication can only be used to authenticate oneself with an individual host
  + In order for someone to perform an action on the Zuthanet using a Weak Identity, they must delegate Strong Authentication of their request to a trusted 3rd party.
  + For example, for a while, Zutha.net will be the sole means of interacting with the Zuthanet. Users will identify themselves with Zutha.net using OpenID (a weak form of authentication) and Zutha.net will perform actions on the Zuthanet on their behalf. By signing a user’s action, Zutha.net will be vouching for the fact that the actual owner of an OpenID was the one that made the request.
  + What if an attacker who owns lots of hosts made them all vouch for a malicious request under the name of a victim?
  + There needs to be a very reliable way of ensuring the trustworthiness of hosts that can perform actions on behalf of weakly authenticated users
  + The primary reason I want actions to be performable using weak authentication is so that the actions performed by users on Zutha.net before the Zuthanet goes fully decentralized can remain trusted events.
    - Another use case is allowing users whose identities have been stolen to re-assert their ownership of their account using backup authentication means like Biometrics. They would need to have a 3rd party vouch for them if using Weak Authentication as a backup .
  + When a Digital Identity is created using Weak Authentication, it must be created through a 3rd party that will vouch for them (such as Zutha.net). This 3rd party should be associated with the Digital Identity such that it remains the only entity that is allowed the vouch for that Digital Identity.
    - In this way, Weak Authentication on the Zuthanet will operate like OpenID does. A Weak Digital Identity is associated with an Identity Provider.
    - So Zutha.net will be an Identity Provider
    - Other 3rd party websites will be able to become Identity Providers as well so that they can provide their own standard web user interface to the Zuthanet
    - There could eventually be Identity Providers who specialise in Biometric identification, allowing users to go to one of their facilities to authenticate themselves and ask for a request to be made to the Zuthanet on their behalf using their Biometric Digital Identity.
  + A User can have multiple Digital Identities that use the same means of authentication but are associated with different Identity Providers.
    - For example, a user U might use their Google OpenID to log into multiple different 3rd party Zutha websites. They would have a different Digital Identity with each website, even though they are using the same OpenID with each.
  + A user adds new Digital Identities to their account by vouching for another Digital Identity using one of the Digital Identities already associated with their account.
  + If one of a user’s digital identities are compromised, he can use one or more of his other Digital Identities to deactivate that Digital Identity.
    - The compromised Digital Identity will then no longer be allowed to perform new actions.
    - An attacker will not be able to use a deactivated Digital Identity to plant events into the past either, because the attacker would need to gain control of the request handlers for the request to determine its timestamp, as well as all the hosts between the historic timestamp and the present who were supposed to sign it.
  + The Zutha.net host will sign all events that occur before Zutha goes distributed
    - If the Zutha.net private key is ever compromised, the controller of Zutha.net will be able to use other private keys to deactivate the compromised one.
    - Once Zutha has gone distributed, it will no longer be feasible to inject new events into the period before decentralization because all the events created during this period will have been assigned to controller groups at the time decentralization commenced. Thus it will be just as hard to inject events into this period as for any period since.
    - Actually, no, it will be slightly easier because one would only need Zutha.net’s private key from the pre-decentralized era to specify the timestamp. The harder part of the injection attack would still remain though, which is obtaining control of the most recent hosts which would have been assigned to your injected event. This is much harder since an attacker doesn’t have as much time to compromise hosts they only recently found out that they need to compromise.
* As I was designing the system to prevent historic injection of events, I was concerned about the fact that attackers would have more time to find and take control of a group of hosts they have known about for a long time. Perhaps the greater vulnerability, though, is that private keys used to sign requests and give them timestamps long in the past may eventually be crackable because of greater computing power and new computing paradigms like quantum computing.
  + This vulnerability will be avoidable because hosts will update the strength of the cryptography they use to keep up with current technology so it will remain impractical to cryptographically compromise the current private keys of hosts.
  + Old private keys will expire such that they are only authoritative as signatures on events that occurred before they expired.
  + In order to use an old private key, then, an attacker must create an event with a timestamp before the private key expired, but this, as already discussed, will require the control of all hosts that would need to have signed the event between its timestamp and the present and this is impractical because the private keys that would need to be used for the most recent of those signatures will be cryptographically secure to current technology.
* What happens if a host refuses to sign valid events
  + It should be the responsibility of the rest of the controller group to ensure that a new member signs off on the event data they send it
  + If a newcomer N to a controller group of item X fails to sign off on X, X\* should report this to the controller group managing the active host list HL. HL\* will ask N for the missing signature or for proof of the invalidity of the item it was asked to sign.
  + Actually, responsibility for checking up on a missing signature should be randomly assigned to a new controller group Z. If Z fails to get a signature from N (or valid proof of an error in the item) then Z will evict N from the network.
  + If N was simply unresponsive, the eviction from the network will not be permanent and N will receive the standard devaluation of its reputation for unexpected downtime.
  + The two hosts that join X\* in the next timestamp should both sign X in that new timestamp.
  + When hosts in the future discover the missing signature from N, they will be able to see proof that N was evicted so the missing signature will be acceptable
* Many items should not need to be signed every single timestamp
  + The frequency with which items need to be signed could vary from item to item depending on how critical the item is
  + Apart from reducing load, this would also reduce the reliance on individual hosts signing every item whose controller group they join.
  + A host should be expected to sign an item if it is due for a new signature. If they are too busy to do so, though, they can pass, but at the possible penalty of a drop in reputation.
  + No this is all silly. Signing an item is practically the lowest overhead task a host will ever be asked to do. It’s absurd to try to save resources signing a hash when the whole item has to be downloaded and processed.
  + Saving of overhead on unimportant items should be done by reducing the frequency with which new hosts are added to its controller group
  + Perhaps the requirement that every item be signed by every newcomer should be relaxed anyway, though, so that every single host does not need to be relied upon in order to retain data integrity.
* There is no need to expose the details of how a Zutha Identity Provider authenticates the identities it vouches for
  + Thus there is no need for users’ OpenID identifiers to be stored with their accounts
  + A Zutha Identity Provider such as Zutha.net will perform actions on behalf of a Dependent Digital Identity item. The Dependent Digital Identity item need only be associated with its Identity Provider.
  + It will be up to users to decide which, if any, Zutha Identity Providers they trust to be custodians of their identity on Zutha
* It would useful if users could authenticate themselves as the owners of various kinds of digital identity even if the providers of those digital identities do not interface with the Zuthanet.
  + For example, someone might want to prove their ownership of an email address, Facebook account, Blog, Usenet handle, Person or Business.
  + All of these kind so identity should be verified with the same system of redundant verification by independent high reputation Zutha users
  + An email address won’t be used to authenticate a user for every request made to the Zuthanet (though it might be used indirectly to authenticate oneself with a Zutha Identity Provider which would then vouch for your requests through it), but it could be desirable to have it associated with your Zutha account. Thus a relatively time-consuming process should not be a big deal for this kind of authentication.
* It could be useful for a user to have an OpenID associated with their user account even if they cannot use the OpenID directly to authenticate themselves with Zutha
  + Once a user has been authenticated as the owner of an OpenID (via redundant checking by multiple high reputation users), 3rd party sites will be able to use the public information associated with that users account after verifying that they are the owner of an OpenID associated with their Zutha account.
  + A use case could be for a book selling site to lookup the books a user has invested in on Zutha in order to make recommendations. They might even use a recommendation service provided by yet another 3rd party which implements data-mining and collaborative filtering techniques on the public Zutha database.

# Thu 3-Nov-11

* It would be elegant if Scope could be represented using properties of a reified construct (property or association)
  + Scope is best conceptualized as a simple list of Items (though an unordered one)
  + An unordered list of values can be represented by multiple instantiations of the same property type. At least, it would need to represented this way in Topic Map notation, but there is no reason why it couldn’t be represented as a simple list of values in the Zutha Serialization Language (ZSL)
  + A list of values in ZSL would need to have a well-defined order for hash uniformity, but the order would not be relevant conceptually. Ordered lists will have to be represented with additional information. For example by using a linked list or by placing index properties on each element.
  + The Scope property would be non-reified, just like the ZID property
  + A property or association’s Scope property would come under the same permission-level as the whole property or association
* Would it make sense to have Reified Association items conceptually store the role players of the association it represents (rather than reifying an association object)?
  + One benefit of this approach would be that all the information necessary to construct an association could be passed around inside a single Association item. If reification was used then an association could only be fully described by the combination of two different kinds of construct: an item and an association construct
  + A Role-Player construct could be represented conceptually by a 3-way association, perhaps called “item plays role in association”, with 3 roles: Item, Role, Association
  + Infinite Recursion can be avoided by making the “item-plays-role-in-association” association non-reified in the same way as the type-instance association
  + The item-plays-role-in-association association should conceptually exist such that it can be used in queries to find, for example, the reified Association items which item X plays the role R in.
  + Hosts would be free to represent associations however they wish under the covers though, including as topic maps the way I am currently doing
  + Non-reified constructs would not be able to participate in non-reified associations, so a type-instance association would not be findable with a search that uses the item-plays-role-in-association association because a type-instance association does not actually exist and cannot be referenced by an ZID.
* Property items could also conceptually have a non-reified Value property
* It would be handy to have a syntax in the Zutha Serialization Language for representing non-reified associations directly inside an item
  + Perhaps only non-reified properties should be represented inside an item block as well
  + Even if the Zutha Canonicalization Format uses the approach of only including non-reified properties directly inside an item, it would be convenient for all properties, and even all associations to be represented inside the focal item
  + But the Zutha Canonicalization Format is really the only serialization format the Zutha specification should define because all messages between Zutha nodes that it talks about must be comparable via their hash value.
  + Perhaps all properties and associations should always be represented within their focal item – meaning properties and associations are always treated as kind of “inner items” of the items they are associated with.
  + But this would mean that Association items would be duplicated in a result set that contains more than one of its players
    - This isn’t necessarily a bad thing – in fact it better captures the nature of the way association items will be managed in the distributed database. Associations will have to be mutually coordinated by the controller groups of each of their players as well as (in the case of reified associations) its own controller group.
    - Actually, for reified associations, any changes made to it will be managed by the Association’s controller group and the controller groups of each of the players will be notified because they are dependencies on the Association.
    - For a non-reified Association, nothing about the Association is modifiable so everything about it is specified in the event that creates the item. Thus the controller groups of each player in it will just store that same initial value indefinitely.
    - The deletion of an item would be an event with dependencies on other players in non-reified associations in participated in
    - Some special events might be allowed (such as changing an Item’s type) which would have dependencies on other players in a non-reified association – in this case a type-instance association.
  + Duplicated Associations would not really cause any problems with regards message size because compression during transport would practically eliminate any size increase.
  + Then again, why am I so concerned about human readability. Only machines are actually going to read these messages in practice.
  + Representing non-reified associations inline would be easy since the only thing that needs representing are the other role players in the association.
  + Non-reified associations should be duplicated if more than one of their players appears in a result-set
* The Zutha Serialization Language (ZSL) should use JSON syntax
* Should ZSL use human-readable item-identifiers when they are present? For example, the identifier “item” would be used instead of its ZID “01”.
  + The problem with this is that if a host didn’t know an identifier, it wouldn’t be able to look it up because controller groups of items are selected by ZID not by text identifiers
  + Also, this would only be valuable for human readability, which is not important
  + I will use text identifiers in my examples, though of course
* A possible serialization syntax
  + I have put self-references in bold

{ //The Robert Jordan Author item

“zid” : ”08FED”,

“permission-level” : 1000, //affects only deletion of an item or changing its type

“type-instance” : {”instance” : ”**08FED**”, “type” : ”author”},

“property-ownership” : {“property” : “26E4DE“, “owner” : ”**08FED**”},

“property-ownership” : {“property” : “2Z8J8 “, “owner” : ”**08FED**”}

“item-plays-role-in-association” : {“item” : “**08FED**“, “role” : ”author”, “association” : “1D42B”},

}

{ //a name property of Robert Jordan

“zid” : ”26E4DE”,

“scope” : [],

“value” : ”Robert Jordan (Author)”,

“type-instance” : {”instance” : “**26E4DE**”, “type” : ”modifiable-name”},

“property-ownership” : {“property” : “**26E4DE**”, “owner” : ”08FED”}

}

{ //a name property of Robert Jordan

“zid” : ”2Z8J8”,

“permission-level” : 100,

“scope” : [“books”],

“value” : ”Robert Jordan”,

“type-instance” : {”instance” : “**2Z8J8**”, “type” : ”modifiable-name”},

“property-ownership” : {“property” : “**2Z8J8**”, “owner” : ”08FED”}

}

{ // The Eye of The World book item

“zid” : ”07FE”,

“type-instance” : {”instance” : “**07FE**”, “type” : ”book”},

// property ownership…

“item-plays-role-in-association” : {“item” : “**07FE**“, “role” : ”written-work”, “association” : “1D42B”},

}

//Eye of the World Name properties…

{ //The Authorship association between Robert Jordan and The Eye of the World

“zid” : ”1D42B”,

“type-instance” : {”instance” : ”**1D42B**”, “type” : ”authorship”},

“item-plays-role-in-association” : {“item” : “08FED“, “role” : ”author”, “association” : “**1D42B**”},

“item-plays-role-in-association” : {“item” : “07FE“, “role” : ”written-work”, “association” : “**1D42B**”},

}

* Well the serialization format is extremely verbose, but I guess there is a lot of information to represent
  + I don’t like the way property-ownership, item-plays-role-in-association and type-instance associations result in lots of redundant information and unnecessary verbosity though
  + The question is, do I want to use a special syntax for commonly used non-reified associations?
  + On the one hand, it would make ZSL look nicer if the type-instance association was compacted to just “type: …”
  + There are also other reasons to try to avoid duplicate information including the fact that it makes validation of syntax more difficult and generally makes the authoring of documents more error prone.
  + On the other hand, using special syntax for the non-reified associations I define now will reduce the extensibility of the language – or will it?
  + Even if the core pseudo-associations like type-instance get a special representation syntax, a general syntax for newly-defined pseudo-associations can still be available
  + Also, new pseudo-associations are essentially only going to be defined in the process of modifying the Zutha standard, since they require new rules to determine how they operate. Thus it may make sense for ZSL to acquire new syntax upon the definition of new pseudo-associations
  + A distinction needs to be made here between pseudo-associations (a term I just dubbed meaning non-reified associations that represent constructs that are otherwise not part of the topic map) and auto-associations, which are defined in terms of other associations. I think these terms need more clarification.
* Pseudo-Association
  + A conceptual non-reified association that cannot be defined in terms of real items, properties or associations of the knowledgebase
  + Hosts must calculate the state of Pseudo-Associations based on events that affect them, but no events will directly reference a Pseudo-Association
  + A Pseudo-Association is like a manifestation of a deeper construct that would otherwise not be exposed via the Zutha API
  + Examples
    - Type-instance
      * Exposes the type property of an item
    - Item-plays-role-in-association
      * Exposes the role-player constructs of an association
    - Property-ownership
      * Exposes the connection between a property item and the item it is a property of
* There is still some lingering terminology in the schema that treats Associations as Fields (Association Fields are Fields but not Associations)
* I need to re-examine zsi:association-field-with-scope-item-declaration and related concepts
* I’m thinking of abandoning the idea of non-reified auto-names
  + Non-reified constructs should only be used when something is specified at the creation time of an item which is not modified afterwards (as in the case of ZIDs), or when the permission-level for modifying the property is specified somewhere else than a reified property item (as in the case of the permission-level property)
  + But it would be useful to have non-reified auto-properties that can be referenced and indexed, but which are calculated from other fields rather than specified manually.
* How do you find an item by one of its old ZIDs?
  + The problem is that an item’s controller group is specified by its primary ZID
  + The controller group that gets selected by a non-primary ZID should simply be made aware of the new ZID of the item so they can redirect anyone that comes looking for an item using an old ZID
  + This should be a very minor responsibility and require no involvement in managing requests involving the item
  + A Redirector Group (as I will dub this kind of host group) should still be punishable if they cannot correctly report the new ZID of an item identified with the old ZID they are responsible for.
* Zutha Serialization Language again
  + It doesn’t make sense for associations to be embedded in items in ZSL, but properties should be embedded in their items
  + Properties that must have exactly one instance in an item should be represented with a single value in JSON syntax
  + All other properties should have their values represented as a JSON list, even if there actually is one instance in a particular item
* Another attempt at the Zutha Serialization Language
  + Keywords are in bold
  + Identifiers that are not bold would actually be replaced by their corresponding ZIDs
  + While I’m still unsure about this, I’m leaning towards using ZID references even for system properties like ZID, value and permission-level

{ //The Robert Jordan Author item

“zid” : [”08FED”],

“**type**” : ”author”,

“permission-level” : 1000, //affects only deletion of an item or changing its type

“modifiable-name” : [

{

“zid” : [”26E4DE”],

“scope” : [],

“permission-level” : 200

“value” : ”Robert Jordan (Author)”,

},

{

“zid” : [”2Z8J8”],

“scope” : [“books”],

“value” : ”Robert Jordan”,

}],

}

{ // The Eye of The World book item

“zid” : [”07FE”],

“**type**” : ”book”,

“modifiable-name” : [

{

“zid” : [”26E4DE”],

“scope” : [],

“permission-level” : 200

“value” : ”The Eye of the World”,

}],

}

{ //The Authorship association between Robert Jordan and The Eye of the World

“zid” : [”1D42B”],

“**type**” : ”authorship”,

“scope” : [],

“**role-players**” : {

“author” : [“08FED”],

“written-work” : “07FE”

}

}

* Ahhh, much more elegant!

# Fri 4-Nov-11

* There should be both a None and Unknown special value
  + Unknown will be used for semantically required fields (like Author for book) which are allowed to be Unknown
  + None will be used to explicitly encode the knowledge of absence of a value. For example, that a Person has no children.
  + None will not be a valid value for semantically required fields
  + The absence of an optional property entirely should represent a subtly different concept than Unknown: that of absence of information.
  + Unknown is a declaration that the item creator did not know the value at the time of creation. This should act as a kind of flag that the value should be discovered if possible.
  + The Absence of a property (or association) is a much less strong statement. It just means that no one has entered a value one way or the other for that field.
  + An optional property should be able to have the Unknown value. This would be a way of someone saying that they think it has a value, but don’t know what it is.
  + Ah, so there is a clearer distinction that can be made between these three values:
    - None = the item is known to possess None of this field
    - Unknown = the item is known to possess this field, but the value is unknown
    - Absent field = it is currently unknown whether the item possesses an instance of this field or not
  + If I’m going to adhere to these semantics, then properties should probably not encode an alternate semantic meaning in their own absence
  + For example, the permission-level property should not be allowed to specify that it should be inherited if missing as this encodes meaningful semantics into the absence of the property (other than the absence of information).
    - The permission-level property should define a special value called “inherit” which is used by default upon item creation. Thus the permission-level property will never be semantically absent.
    - This isn’t to say implementations of the Zutha standard can’t encode the “inherit” value of the permission-level property as the absence of the permission-level property internally, but they must report a value of “inherit” for absent permission-level properties when responding to requests that ask for it.
* There should be a special permission level (say “schema”) for constructs that are relied upon by the current version of the Zutha Standard
  + Constructs with permission level “schema” can only be modified during an update to the Zutha standard
  + Updates to the Zutha standard will have various procedural requirements that ensure stability. For example, an update will have to be scheduled for a specific date and will require a minimum percentage of hosts to give the go-ahead before they can even be scheduled.
* Application design ideas
  + Schema Items should be represented by Scala Objects
    - The Objects I want to use in most cases already exist.
    - For example, the ZItem Scala Object should actually be the instance of ZItem with identifier zsi:item.
  + Item objects should be immutable
    - The schema items need to be exceptions so that they can be implemented as Scala Objects.
    - For example, if ZItem is a Scala object and it implements its Properties field as a val, then it won’t be possible for it to be modified without restarting the application
    - Actually, instead of making all Items be immutable, there should be a subclass of ZItem called ZItemSnapshot, which is immutable and can be created from any existing ZItem object.
    - The reason there is a need for an immutable version of items is so that they can be used to calculate a diff from when modifying items.
    - If user U1 is modifying item X and item X has changed by the time U1 commits, I don’t want U1’s changes to be calculated based on the new value of X because U1 didn’t know about the new value. Also, it is possible that U1’s changes conflict with the changes that occurred in the meantime.
    - Why don’t I just record a modified timestamp and reject U1’s changes if X changes before he commits?
    - In fact, while Zutha is not yet distributed, I can just mark items that are being edited as locked and not permit them to be changed by others until unlocked. There should be a time limit on locking an item, though so that Users don’t accidentally or deliberately leave Items locked indefinitely.
    - Once Zutha is distributed, the issue of collisions will be dealt with in a whole new way, but I don’t have to worry about that yet.
    - Mind you, Zutha.net will still operate at least for a while once the Zuthanet is distributed, and it will have to change to allow multiple users to attempt to edit at once when that time arrives.
    - Another problem with the locking approach is that it is too sweeping a lock given the degree of interconnectedness of Zutha data. For example, should the Comment item be locked whenever a user is in the process of creating a new instance of Comment?
    - A lock’s scope could be limited to only properties and existing associations of the locked item
      * Still, what happens if User 1 maxes out item X’s allowed association fields of type Y and User 2, who started editing before User 1 tries to add another Y field? Without immutability, the state of User 1’s edit form would change while he edits. In fact, his edit form could show that all is well, and then fail when the form is actually submitted. This isn’t really so bad though.
    - What I really want is for Edit requests to be calculated as a diff from the moment the user pressed the Edit button. This requires an immutable version of the item as it was at that moment to be associated with the Editing form until it is submitted.
      * If conflicts are detected at submission time then the whole request will be aborted
      * If someone else edited a different part of the item in the meantime though, this should not cause a conflict and both events will be permitted
    - Wouldn’t it be kind of convenient if you were updated on changes made by others to the item you are editing in real time?
      * Imagine you are editing an Item and you have changed its description to a new value V1 when all of a sudden its value changes to V2 (along with a notice saying what happened) because another user has modified it before you had a chance to submit your changes.
      * To facilitate this functionality, collision detection should occur on each ajax request
      * An ItemBuilder should store the timestamp T0 at which it was begun
      * If, on a new ajax request that modifies the ItemBuilder, the timestamp of the underlying Item X has changed, the ItemBuilder should examine the event(s) that have occurred on X since T0 and find any collisions with properties or association fields that have been modified in the ItemBuilder.
* Properties of Associations should not be allowed to be reified properties
  + There should be a subtype of Property called Association Property which is not reified
  + An Association’s properties inherit permission-level and scope from the association
  + Does this mean an association is not allowed to have a modifiable-name?
  + Also, what happens if someone wants to upgrade an association type to represent a more concrete entity with additional properties and even its own association fields?
    - For example, a Marriage might start out as an Association, but later be upgraded to represent an actual Marriage event involving associations with guests etc.
    - For such a use case the additional fields that are added to a Marriage association should be reified because some may need their own permission-level and scope
    - I think that, at least in this particular case, it would be better to avoid complicating an Association Type to describe an event as well. There should be a Marriage association type and a Wedding event type for example.
    - Still, it is conceivable that it would be appropriate to upgrade an Association Type in this way for some use cases
  + There should be an Association Property Type which is used for properties that are actually part of an association. This kind of property will not be reified and will inherit its permission-level and scope from the association
  + An Association should still be allowed to have normal properties though; typical use cases being Modifiable Name and Description. These kind of properties will be reified as specified by their type, and they will not appear as members of association fields alongside role-players and association properties.
* I’m abandoning the idea of reified-auto-properties
  + Calculated Properties will be non-reified
  + Use Default Value instead of Reified Auto Properties
  + Datatypes, Property Types and Property Declarations should all be able to specify a Default Value. The Default Value used will be the most specific one define where specificity (from least specific to most specific) goes in the order: Datatype, Property Type, Property Declaration

# Sun 6-Nov-11

* What happens when two different types of an item have declared a Field Set of the same Field Type?
  + For example consider a Gamebook like *Journey to the Year 3000* which has Book and Game among its types. If both Book and Game independently define a member:member-of-category association field set, then which of these declared association field sets determines the meta-data (like cardinality) of association fields of type member:member-of-category?
  + If the only metadata defined in a field declaration was the cardinality and the cardinalities of the two field sets matched, then there would be no real issue. And if the cardinality of one field set was stricter, but still compatible with the other field set then the stricter field set could implicitly override the other. However, these rules won’t generalize to cases where the cardinalities of the two field sets are incompatible (such as if one is [1:1] and the other [2:2]), let alone where there may be unforeseeable additions to the kind of metadata that a field-set declaration may define.
  + This issue is very much related to the classic issues of multiple inheritance experienced in object oriented programming such as the Diamond Problem
  + There may be a solution open to Zutha for this problem that is not available for analogous multiple inheritance problems in a programming context. The entire existing ontology is available at the time new ontology rules are defined, so the new rules can be subject to constraints that potentially incorporate all possible scenarios the new rule may allow. In a programming context this cannot be done because different pieces of code are isolated from one another and may be written and executed on entirely separate machines yet make use of common libraries.
  + It could be required that any two types A and B that declare a field set of the same field type must be disjoint – that is, there must be no way for an item to be both of type A and of type B.
  + The Gamebook scenario can be restructured to satisfy this constraint by refactoring the member:member-of-category association field into a Categorizable trait
    - Another approach would be to specialise the member-of-category association into subtypes book-has-category and game-has-category, with appropriately subtype roles as well. This would allow a separate set of categories to be available for games versus books, while still allowing all categories to be viewed via the member-of-category supertype association.
    - Actually both of these approaches should probably be combined. Game and Book should be required to mix in the Catagorizable trait in order to be allowed to participate in a subtype of the member-of-category association.
    - This approach will lead to three separate association field sets appearing in *Journey to the Year 3000* (an instance of Gamebook); member:member-of-category, book:book-has-category, and game:game-has-category. Note that these are three different field types, each declared by different types, so there is no problem of ambiguity about which field declaration specifies the metadata for a given field.
  + Here’s a more difficult scenario
    - The Region type declares the contained:contained-by association field set
    - The Place type also declares the contained:contained-by association field set
    - A place is contained by exactly one region and a region may be contained by a larger region
    - For example, The UCSC ILC (treated as a place) is contained by UCSC and UCSC is contained by Santa Cruz
    - So far it is ok that contained:contained-by has been declared by two different types because there are no items that have both types
    - But what happens if we want to assert that Room 5403 is contained by the ILC? The most sensible thing to do is to give ILC the trait Region. But now ILC has two conflicting definitions of contained:contained-by – one with cardinality [1:1] declared by Place and one with cardinality [0,1] declared by Region.
    - What we actually want to happen in this case is for The ILC to have contained:contained-by as declared by Place and just add container:contains as declared by Region.
    - Using the rule that the more restrictive cardinality prevails would result in the outcome we desire for cardinality in this case, but this rule will not generalize so it cannot be relied upon even in the cases (like this one) where it would work as desired.
  + What if field declarations have the additional metadata of default-permission-level and display-order-precedence?
    - If Place’s more restrictive cardinality for contained:contained-by will override Region’s cardinality for contained:contained-by in the ILC item, should that determine which default-permission-level value should be used for the contained:contained-by field set in ILC?
  + The contained-by conflict could be avoided in a similar way that the Gamebook one was.
    - The contained:contained-by association field should be declared by the trait Containable
    - The container:contains association field should be declared by the trait Container
    - Region should mix in Container and Containable and declare its own subregion:contained-by and super-region:contains specializing association fields
    - Place should mix in Containable and declare its own place:contained-by specializing association field
    - This solution would work in this case, but it would be a massive pain to convert to this design if containment semantics were initially implemented differently and problems were only found with it after many items already directly used the contained:contained-by association field type.
  + Generalizing the design principle I’ve just devised for containment and categorization seems to imply that every field type should be declared by at most one type.
    - This is essentially the rule I had (for association fields only) when I was experimenting with the idea of Roles and Traits being the same thing; an item would specify that it can play a Role and the Role would be the declarer of associations it participates in.
    - I abandoned this idea because of these two problems:
      * The role “victim” could be played by a mortal in the association “murderer-murders-victim”, but a mortal wouldn’t play the role “victim” in an association like “victim of hoax”
      * Roles also might have different cardinalities for different players. For example, a phoenix could play the role “died” or “killed” more than once, but a normal mortal could only play that role at most once.
    - The first of these problems is not applicable if the whole association field (victim:murdered-by) rather than just the role is the unit of declaration
    - The second problem can be solved by allowing field declarations to be overridden, but only in a linear chain
      * For example, a Trait called “Can Die” (I couldn’t find a better name than this) could define the Date of Death property with cardinality [0,\*)
      * The Mortal trait could then subtype Can Die and override its declaration of Date of Death with cardinality [0,1]
      * Another trait, say Historic Figure, would be allowed to inherit from Mortal and override its declaration of Date of Death, but would not be allowed to inherit from Can Die and directly override its declaration of Date of Death.
      * If field declarations can only be overridden in a linear chain, then it will always be well defined which declaration applies to a given field set
      * If, on the other hand, a field declaration is directly overridden more than once by different types, then the diamond problem can still occur.
      * For example, imagine we have a Dies Twice trait that subtypes Can Die and overrides its Date of Death declaration with cardinality [0,2]
      * If some item was to end up with both the Dies Twice and Mortal types, then it would not be clear whether it is allowed to have 2 death dates or not.
      * It is conceivable, though, that a situation with a similar structure to this could occur where it would be desirable to have more than one specialization of a field declaration.
      * The conflict here only occurs if an item can be both a Dies Twice and a Mortal, but these two types should be disjoint
      * Conflicts like this will be avoided if types that override the same field are required to be disjoint
    - So the design principle is that a field type should only be declared by types that are guaranteed to be disjoint.
      * Actually, the act of declaring a field type that has already been declared or overriding a field type that has already been overridden should require an explicit disjoint declaration with the other types that have already declared/overridden the field type.
      * The design principle is really stating that types should not be forced to be disjoint in this way unless it is guaranteed that there won’t be a use case requiring something to have more than one of the disjoint types.
  + A convention should be established that fields be declared by types that capture the logical essence of what kind of entity has that field
    - So it would be inappropriate for Book to declare the member:member-of-category association-field because being a Book does not capture the essence of what kind of entity is able to be a member of a category.
    - It would be appropriate for Book to declare the field book:book-has-category though, since this field is clearly specifically applying to entities of type Book. A use case for doing this would be to narrow the options for what categories a book is allowed to be a member of.
    - Similarly, it would not be appropriate for Place to declare contained:contained-by because there are other kinds of things than places that can be contained
  + I think this convention would be helped along substantially if fields were required to have only one top-level declarer. Additional declarers would have to be ancestors of the root declarer and would have to override the root declaration (possibly indirectly through multiple intermediary overriding declarations).
    - This would force ontology designers to think about whether they really need to re-declare a field or whether they should just mix in a type that has already declared it.
    - For example, imagine that user X is creating a type for a Virtual Puppy and wants it to have a Birth Date field.
      * It might be that the Virtual Being type from which this new Virtual Puppy type is going to inherit does not declare a Birth Date field
      * X’s easiest course of action would be to have Virtual Puppy directly declare Date of Birth, and this is probably what most people would do given the choice
      * Consider for a moment that this is what happens: Virtual Puppy gets its very own Date of Birth field
      * Now imagine that someone else, user Y, defines a Virtual Pet type and does the same thing, giving it its own Date of Birth field
      * Now when someone creates an instance of Virtual Puppy and gives it the trait Virtual Pet, there will be a collision between the two declarations of Date of Birth. In this case it is likely that both declarations would be identical, but it is possible that one has a different default-permission-level or a different display-order-precedence value.
    - Now let’s re-examine the Virtual Puppy scenario in light of the requirement that all declarations of a field type override (possibly indirectly) a root declaration
      * User X will no longer have the option of directly declaring a Date of Birth for Virtual Puppy because the Date of Birth field will already have been declared by another type
      * Let’s say, to make things more challenging, that Date of Birth has been declared by Person only so far
      * In order for User X to give Virtual Puppy a Date of Birth now, he must first do some house cleaning
      * X is forced to think about what type (most appropriately a Trait in this case) should be the root declarer of Date of Birth
      * An appropriate choice might be a trait called something like “Is Born” which declares Date of Birth with cardinality [1,\*)
      * Subclassing Is Born might be “Born Once” which overrides Date of Birth with cardinality [1,1].
      * In order to make these changes valid, User X will have to modify Person to mix in the Born Once trait and remove its own declaration of Date of Birth before committing the transaction.
      * X can now mix in Born Once to Virtual Puppy and Y can mix in Born Once to Virtual Pet and there will be no collisions

# Sat 12-Nov-11

* Zutha’s reputation system could be imported into 3rd party sites
  + Let’s consider a use case of the Khan Academy basing its reputation and rewards system on top of the Zutha platform
  + In order for Khan Academy to be able to allocate reputation to students and tutors that use the site, it will need a seed allocation of reputation on Zutha itself
  + So let’s imagine that lots of already highly reputable users on Zutha have invested Zuth into the Khan Academy and so the Khan Academy has a high reputation and a large income of Zuth to spend.
  + Khan Academy could automatically reward achievements on its site with investments of Zuth (they would be responsible for preventing their system from being gamed as it would be their Zuth on the line)
  + Students would gain a reputation on Zutha for their efforts at Khan Academy and would begin receiving an income of Zuth.
  + Students could then invest in helpful answers to questions and videos they like etc.
  + There would then be an incentive for people to help students on Khan Academy in order to earn a reputation of Zutha.
  + A reputation on Zutha would have a more transferable value than reputation on Khan Academy alone and so this system provides a more potent incentive for people to contribute their time to teaching students at Khan Academy.
  + Khan Academy is just a specific example of course – this concept could be transferred to any number of use cases.
  + As more services start using Zutha has their reputation platform, reputation on Zutha will become ever more valuable, creating stronger incentives for more services to plug into it.

# Wed 23-Nov-11

* The investment interface should automatically populate the amount field with something sensible based on the amount of Zuth available to the user
  + This will create a psychological anchor on that value. This may have good and bad consequences.
  + More importantly, it will reduce the need to make a decision about how much to invest, which will probably result in more users bothering to invest.

# Fri 2-Dec-11

* I think names should be implemented in a similar way
  + Name should be a simple, top level property type
  + Any features like auto-generated names and such should be added to the system later in a way that doesn’t make the straightforward use of names any more complex

# Wed 7-Dec-11

* I should start Zutha without a focus on long term ambitions. Instead, I should focus on creating something that is simple and useful and encourage people to contribute in a low key way. It will be important to cultivate trust that contributors will retain control over their creations though.
* Reading Here Comes Everybody has convinced me of the value of this more low key approach. Wikipedia and Linux both started this way and I think it is an important pattern to note. I don't think a grand vision in a new and unproven concept is attractive to prospective contributors, perhaps because people have an innate wariness of giant new visions due to their overwhelming tendency to fail. Perhaps people are more willing to contribute to low key new ideas (provided they have some incentive like interest, usefulness or generosity) because they are not thinking in terms of its likelihood of failure. A grand vision however will draw people's attention to its prospects for success or failure.
* Zutha will allow groups to have bank accounts. currently you must incorporate to get a bank account owned by more than one person.
* Zutha is trying to facilitate the highest level of cooperation: collective action. I think this will be the next stage in our society's evolution in cooperative capability and it will have very powerful effects.

# ? Late Dec-11

* Write a book and film script in parallel about the birth of Zutha. The events of the novel and film script should line up exactly. The book will contain character inner monologue and other book-specific features which cannot be represented in a movie and vice versa. These medium specific features should add richness to the story without being fundamental to the plot.

# Sun 15-Jan-12

* Users should be able to (and encouraged to) declare projects they are working on that they hope to get reputation for. Example projects:
  + Filling in missing metadata about albums and tracks by a particular artist
  + Locating (and adding to Zutha) as much of the currently available software in a specific category as can be found
  + Finding and rewarding (by zuth investment) users who are making great contributions to Zutha’s knowledgebase in a particular field such as The Wheel of Time
    - This is a kind of meta-project of which I expect there will need to be multiple levels for the reputation framework to be truly effective in shaping user behaviour towards benefiting Zutha as a whole.
    - A user who undertakes this project of rewarding the labour of users who add data about The Wheel of Time would expect to be rewarded by other users playing an even higher level meta-role.
    - Users taking on a level 1 meta-role should be judged on how good they are at locating well-performing users in the tier below them.
    - Users taking on a level 2 meta-role would judge level 1 meta-behaviour by randomly sampling their work, assessing the appropriateness of their praise to other users. The level 1 meta-users who are the most prolific in finding users worthy of praise and are the most fair in assessing them should be rewarded the most.
    - If a user is found to be rewarding users who clearly don’t deserve it, this should be criticised.
    - Blatant favouritism should be investigated and reported to interested members of the Zutha community so that it can be considered as evidence for merging ownership of the two users involved. (By merging ownership I mean marking the two users as having a shared real-world controller so that they are treated as the same user for many purposes by the network.)

# Thu 19-Jan-12

* Lisp is the perfect model to use for thinking about how a programming language could be implemented in the Zutha cloud
  + A Lisp program is just a tree of symbols and atoms
  + A Cons could trivially be represented as a Zutha item with two association fields to the “car” and “cdr” fields (in lisp terminology)
  + Actually this would be a doubly-linked list if implemented with standard Associations.
  + It might be worth considering making one-way links an available construct in the Zuthabase (a term I just coined)
    - Only the behaviour of one-way links needs to be standardised. They can be implemented however is most convenient to the implementer.
    - In the ZSL (Zutha Serialization Language), one-way links could be serialized as properties whose value is a ZID
    - One-way links can just be a property-type whose value type is another item
    - The One-Way Link property type should be subtyped to restrict the allowed type of the item linked to
  + A function definition would be an Item of type Function
    - It would have a field of type Linked List (beginning with a one-way link to a Cons item) which specifies the argument names (and possibly types as well)
      * An argument name would be specified by a link to a Symbol item which would act as the name for that argument for the scope of the function
      * A Symbol item would have a human-readable name (potentially multiple names with different names in different scopes) but the human-readable name would not be significant for the definition of programs, it would only be used when the program’s code is to be expressed for humans to read.
    - It would have another Linked List field for the body of the function
      * A function body would be much like a Lisp function body being composed of function calls and macro calls
      * Variables would be specified by links to Symbol items. This could include Symbol items that were specified as arguments to the function in which case the value that will be associated with that Symbol for the scope of this function call will be whatever value was passed to that argument for the current call to this function.
      * A Symbol may also have a value in a scope outside that of this function call (i.e. a global variable – of which there may be multiple levels)
      * A Symbol could represent a function as easily as any literal value so a link to a Symbol that represents a function in the current scope could be used instead of a direct link to a Function item for calling functions.
  + A function call would be a linked list whose first element is the name of the function to call and whose subsequent elements are the arguments
    - The function to call would be identified by a link to a Function item
    - Any of the arguments could also be linked lists or other data structures and may be executed before being passed to the named function
* Some Definitions
  + Zuthanet
    - The network of nodes that constitute the Zutha network. Comparable in meaning to the Internet
  + Zuthabase
    - Refers to the actual data stored by Zutha nodes and made accessible via the abstract Zutha API.
  + Zutha
    - Encapsulates everything that Zutha is.
    - This word should also be used when there is no need to distinguish between the database and the network of nodes.
* Once the Zuthabase contains huge number of tiny items (such as Conses), it may be desirable to package groups of items together for passing around among hosts
  + There is always going to be a need to locate an item just from its ZID though, so the controller group selected for each tiny item will need to have at least the responsibility of knowing where to find the primary controller group for the item
  + For example, a Function item might be the grouping item and all Conses it links to would all be under the responsibly of the Function item’s controller group, (say GroupF)
  + If someone tried to locate one of the Conses (say X) from its ZID alone, the controller group specified by this ZID would point the user to GroupF
  + It would be much better if the client could know to go directly to GroupF so that all burden could be lifted from the hosts that would otherwise have to know about X. It may be that the resources required to just notify hosts of their ownership of new items every timeframe, when multiplied by enough items, will be almost as great a bottleneck as giving the hosts the extra responsibility of managing the item in its entirety. This seems even more likely to be true when I consider that for most items in most timeframes, the hosts responsible for them will not be required to do anything except store a tiny amount of data, thus hardly anything will be saved by splitting that responsibility in two.
  + Perhaps some additional information could be added to ZIDs to allow different types of items to be treated differently by the selection algorithm
* Why not specify the type of an item in its ZID?
  + An item’s type is fixed so the ZID will never become unaligned with the actual type
    - Actually I do remember considering that it might be useful to be able to change an item’s type to a subtype of its old type so that a broad type can be made more granular.
    - If I’m going to specify item type in ZIDs then I think it will be best to make item types 100% fixed so I’ll need to think of another way to deal with this scenario
  + An item’s type is a not-reified field, so the ZID of a type item encoded in an item’s ZID would be all the information that is needed about an item’s type
  + Including the full ZID of the Item Type in every item’s ZID would not be ideal because it would lead to a great deal of redundant characters in every ZID. The ZID of the Type item, for example, would be included in every single ZID and the ZID of Item Type (which itself would include the ZID of Type) would be included in the ZIDs of most items.
  + Type items could be excluded from including their Type in their ZIDs
  + All Type items would be treated the same by the selection algorithm, thus the absence of a type field in an item’s ZID would be all the information the Selection function needs to determine which selection parameters to use (those specific to Type items)
  + A ZID could look like: [Host ID][Type ID][Unique ID}
  + In order for Type IDs to be separable from the rest of the ZID, they will need to follow the same scheme as Host IDs, where the first character specifies the length of the ID.
    - One way to achieve this would be to put Host IDs and Type IDs in separate namespaces
    - For example, a Host ID of say “host25G” would be referenced in a ZID as 25G
      * And a Type ID of “type2CC” would be referenced in a ZID as 2CC
  + Type IDs could be kept shorter by putting them in a global Type namespace (as opposed to putting them in a host-specific namespace as with normal items)
    - This might require the creation of a new type to be a much more privileged endeavour than it would be otherwise because it would require a more complicated coordination process to select a unique Type ID than it does to select a normal ZID where uniqueness is guaranteed by the allocating host.
    - A process needs to be found to select unique Host IDs, so the same one can be used for Type IDs
    - Come to think of it, there may be no reason after all for the creation of Types to be super-privileged. Assuming that creations of types will be relatively rare, the extra network resources required to perform the more complex process of selecting a unique Type ID should be no problem.
  + Example ZID: 25F34GT5G7Z:
    - 25F is the Host ID
    - 34GT is the Type ID
    - 5G7Z is the unique ID

# Fri 20-Jan-12

* A field-set should be divided up into subsets for each available scope
  + It should be possible to associate different permission levels with a given field-set for each scope it is allowed to have
  + A Field-set’s subsets should be conceptually arranged in a hierarchical manner
  + A permission level can be associated with the whole field-set, defining the default permission level required to modify any field in the set in any scope
  + A permission level associated with a broad scope acts as a default value for all sub-scopes
  + The permission level for a given field will be determined by the most specific permission level declaration
* Every item could have a long-term controller group
  + This could eliminate a need to encode categorization information in item ZIDs
  + If you are trying to find the controller group of an item you already know something about (such as its type) then you will probably be able to determine which algorithm to use to find its main controller group (the frequently changing one). This should be the most common case.
  + In the rare case when an item needs to be located from only its ZID, its long-term controller group can be found using a universal algorithm that depends only on the ZID.
  + Long-term controller groups would be updated much more infrequently than an item’s main controller group and would therefore be much more vulnerable to accidental loss of members or deliberate sabotage.
  + For most items most of the time, the long-term controller group should not need to be used and its use should be discouraged, particularly for critical items, so that there is no incentive for attackers to target an item’s long-term controller group.
  + For items which are accessed very infrequently, though, a lot of network resources could be saved by reducing the need for their ownership to be passed around frequently.
  + Some items may even have only a long-term controller group
    - Archived items, events and Conses might fall into this category
    - It would be important that such items be stored by more hosts than just their long-term controller group and that their authenticity can be verified once their long-term controller group has been contacted.
    - I think the event hash chain system will robustly protect against compromised controller groups being able to manipulate the items under their control because a user will always be able to follow the hash chain to verify an item’s validity.
    - The main vulnerability of stagnant controller groups will be to being nuked entirely, perhaps for the purpose of eliminating access to data they were responsible for.
* Many items could be grouped together and passed around as a bundle between hosts
  + This would probably best be done by using the hash of the ZID and allocating different parts of the hash number space to a kind of index item, whose controller group would be expected to keep up to date with all the items whose ZID hash lies in the range of that index item.
  + This kind of bulk allocation would be appropriate for the long-term controller groups, to further conserve resources. (It is probably going to be much more efficient to pass around large collections of items in one data stream rather than as many asynchronous messages)
  + Actually the large-scale grouping of items is probably only going to make sense for items that will change very infrequently such as event items, archived items and immutable items. It would defeat the purpose of grouping items together if individual hosts will still have to manage the asynchronous updates of all of the items in the bunch they are responsible for. Also, once a large set of items is grouped together they will have to remain together regardless of the capacity of the hosts they are allocated to. Thus a small host may be overwhelmed if it is given responsibility for a huge collection of items containing lots of high demand and rapidly updated items.
  + It could work to have items be allocated to either bulk archive or standard individually selected controller groups (or possibly both) based on real time discretion of the user community.
  + Items that are detected to be accessed and updated infrequently could be automatically or manually archived at which point they would transition from having their own controller group to being part of a bulk archive based on the hash of their ZID.
  + This would have advantages for auditing because it would be easier to check that a host is faithfully maintaining an entire archive of items than it would be to verify all the items they are responsible for individually. More significantly, the items that end up in an archive will be ones that users care less about and so there would be less incentive to audit them individually. If they are packed together however, there would be an incentive to audit the entire archive (because there would be – or at least might be - at least some important items in there).
  + Items that are accessed and updated regularly are less vulnerable to neglect by hosts because the demand for them would create an incentive to maintain them so that hosts can earn reputation from successfully responding to requests involving them.
  + When a client tries to find the controller group of an item it will probably ask its individual controller group first. If they don’t know about it, the client will then try contacting the controller group of its allocated archive. If neither know about it, then it has probably either been deleted or never existed.
  + For some kinds of items, like event items or immutable items, a client (that knows the item’s type) will know to only check the archive controller group because such items will be allocated to archive status immediately upon creation.
* Immutable Items
  + Especially once data starts being used to construct programs (a la lisp), it may be very useful to have immutable data structures – especially lists
  + Immutable Conses would have the advantage that they could be reused. If they were mutable, then modification of a Cons for one purpose might break something that reused it in another context.
  + A guarantee of immutability would also allow Conses to be more efficiently managed in a distributed system because there would be no need for them to have a dedicated controller group. Instead, they could be copied around to whatever hosts need them just like an atomic value like a number or string.
  + A host that is asked for a linked list data structure associated with an item they are responsible for would not need to ask or listen for updates about all the Conses in that linked list since they would know that there will never be any such updates. Thus they can simply store the Conses in their own database and return them when required without any need for network calls.
  + If Conses will not need a controller group and they cannot be modified is there any need for them to be a discrete item at all?
    - In order for it to be possible for a Cons to have multiple incoming links, it must have an identity (a ZID)
    - But if a Cons can never be modified then there would be no difference between a shared list tail and a duplication of that tail in every place that would have linked to the shared tail.
    - In the distributed context where data structures would be duplicated on multiple hosts anyway, there would be no resource conservation benefit to the conceptual sharing of a list tail
  + So perhaps there isn’t really any use for immutable data structures. Any functionality they could provide would be just as easily be provided by more complex value types
* Mutable Conses still make sense, though, because they enable more granular revision control and the ability to link to a list (such as a program) that might change, from more than one place

# Wed 25-Jan-12

* Larger investments should carry a greater expectation of a satisfactory justification
* Items can be marked as “controversial” or “sensitive” (or some more appropriate name), in which case a higher burden of justification will be expected for investors. This basically means it should be seen as more appropriate for vigilante users to rebuff users who post unsatisfactory justifications with their investments in these kinds of items.
  + An example scenario: An amateur film that uses some content created by Disney needs to have ownership divided up among the content contributors appropriately.
  + This process would be done via a decision market open to the whole Zutha community (perhaps weighted towards reputation in the film domain)
  + If a really high reputation user decides to invest an enormous amount to shift ownership of this amateur film towards Disney and away from the amateur producers, then they would need to provide an extremely good justification for this – one that most users would accept. Otherwise the code of the community would dictate that it is appropriate to criticize this action by the powerful user, and many users may do so. This norm would create a strong disincentive for a powerful user to accept money from Disney in exchange for attempting to grant them greater ownership of a popular amateur production that uses Disney content.
  + If lots if separate users independently vote to increase Disney’s share of the ownership pie, then a user who disagrees with this trend may mark the item as “sensitive”, which would simply be a social cue that a more rigorous level of justification should be expected for every investment in this item. If there are a lot of users who disagree with the users pushing for greater Disney ownership of the film, and believe that the Disney endorsers are not justifying their position sufficiently, then those Disney endorsers may find that their reputation suffers as a result. This should reduce the influence of the group of Disney endorsers (because their reputation would fall) and would also create a disincentive for anyone to take bribes to endorse a position since taking a position you have been bribed to take probably means you will be opposing the popular opinion and will be at risk of retribution.
  + The label of “sensitive” on an item will be open to contest as well through the normal means of conflict resolution: escalating permission levels and decision markets.

# Sun 29-Jan-12

* Before an item has been claimed, real money investments should be presumed to be donations. Once the item has been claimed, however, the owner should have the choice to retrospectively grant investment assets to the donors.
  + A norm should be encouraged that expects owners to do this as a way of showing gratitude towards the donors that both promoted their item and gave money to them gratis.
  + It would also be in an owner’s interest to do this if they expect to create any new items as it would improve their reputation as someone who rewards generosity towards them
  + It would be beneficial for all creators generally if a norm arose around this protocol (retrospectively grating investment assets) because it would create a greater incentive for speculators to donate to items that have not yet been claimed; they would expect to become the earliest investors in that item once claimed and thus be in the best position to benefit from the future popularity of the item that they helped promote by their public donation.

# Mon 30-Jan-12

* The Power Belt that Daniel Suarez describes in his novels for powering HUD glasses could be realised using wireless power transmission by Resonant Inductive Coupling.

# Wed 1-Feb-12

* I want to write a “recursive narrative” that blurs distinctions between fiction, prophecy and documentary
  + For example, imagine a story about the future (or present) of Zutha that depicts the Zutha community working on a collaborative project to produce an evolving story (perhaps in a mix of formats) about Zutha to inspire the public to participate in their community and their causes. This story could itself, recursively describe this dynamic and the story (and the story within the story) could evolve to reflect events that occur in reality.
  + This mind-bending dynamic could lead to a lot of creativity in blending the future with the present.
  + As an example, imagine the community inventing stories to depict possible disturbing consequences of crumbling civil liberties such as activists being spirited away from their homes by government agents for being threatening to the establishment, or crowds of Zuthans being rounded up by police to have unauthorized (and not government controlled) computing devices confiscated from them. Accompanying such stories might be stories of triumphs by the Zutha community where the solidarity and resourcefulness of the community lead to the prevention of such crimes being perpetrated. These kinds of stories could be presented in a dynamic list which could come to combine not just made up stories predicting the future, but eventually real stories of the same nature with links to their presentation by real-world news providers. This blurring dynamic could be implemented in virtual worlds where the imagined scenarios are acted out in practice in the virtual world or portrayed in its in-world history, where it could appear side-by-side with similar stories from the real world.
* I should seek support from the Free Software Foundation to develop Zutha
  + In Free Software, Free Society, I found an appeal by Richard Stallman for volunteers to work on free software, peer to peer replacements for web applications that would solve some of the issues with SaaS in removing user freedoms.
  + Zutha is precisely such a peer to peer application
* A way for collaboratively producing audiobooks
  + Provide a tool on the Zutha platform for annotating a book to assign blocks of text to characters
  + Provide a tool that uses this annotated text and provides an easy tool for users to record themselves as the voice for a character of their choice. The software would automatically highlight what they need to read next, along with context. It would associate the recordings with the corresponding text (which would also be useful for producing combined text/audiobook files)
  + Once many recordings of each character have accumulated, users could vote on the best voices of each character and compositions could be made of the best combination of voice actors.

# Thu 2-Feb-12

* I need to think about how to design a decentralized legal system that would create incentives to prevent crime (and in particular repeat offences) instead of focusing on punishment for the purposes of revenge
  + If protection agencies are paid to protect their clients, the incentive their clients may impose on them could be to focus on punishing successful criminals to appease their clients’ desire for revenge (perhaps justifying their behaviour with the claim that the punishment will deter other criminals).
  + A protection agency would not want to face the cost of imprisoning criminals who harm their clients because it would cost them money to no particular benefit. Their incentive would be towards suffering or death as punishment for crimes against their clients.
  + If the community as a whole is sufficiently opposed to such approaches to punishment, then the empowerment that a Zutha-based society would give to the community as a whole to organise to achieve their collective values may be sufficient to prevent the protection agencies from defying their values. Maybe. However it would obviously be far better if the policing system was designed to have its interests aligned with those of society.
  + What is needed, I think, is a way to measure the impact of actions by protection agencies on society’s long-term interests. A means could then be devised for rewarding the protection agencies for acting in the interests of society as far as can be measured.
  + Imagine if a protection agency was commanded by a court to take custody of a criminal. They would be required to provide the inmate with a minimum standard of living at their own expense (subject to open auditing). They would be permitted to release the offender at their own choosing, but would remain liable to some degree, to subsequent offences by that person. This would create an incentive for the protection agency to find cost-efficient ways of rehabilitating (for want of a less political and vacuous term) and retraining the offender.
  + It would be better again if the whole concept of imprisonment could be avoided as much as possible. Even if the actual balance of incentives could be made to favour pursuing real rehabilitation, the concept of locking people up to keep them away from society – even without all the other counterproductive practices associated with most prison systems – probably has fundamental flaws. And at any rate it is worth searching for alternatives in case better ones exist.
  + A huge proportion of imprisonment seems to result from poverty and the various dysfunctions and lose-lose choices that imposes on people. For such cases, it would be surely be more beneficial to society to provide the support required to allow such people to become productive citizens than to lock them up at public (or in the decentralized case, commercial) expense.
  + Who could have (or be induced to have) an incentive to support a person with a poverty and crime stricken background in turning their life around?
  + When people chose to buy protection services from a particular protection agency, they are endorsing that agency generally.
    - It would seem to make sense to combine the identification of good protectors and good rehabilitators with the same monetary vote.
    - That could be done be requiring a person’s protection agency to be responsible for supporting them in avoiding committing crime as much as they protect them against crime.
    - Two things immediately strike me as concerning about this idea.
      * The poor people who are most likely to commit crime will not have enough money to pay for a protection agency
      * The current legal system does not lead to lawyers trying to convince their clients to abandon a life of crime.
    - The first problem may not be so bad. The system already requires a way for public funds (gathered by Zutha and allocated by Zuth investment or directly donated) to be directed towards paying for protection agencies to serve people without the means to pay for one themselves. It seems likely that the non-profit institutions who manage the allocation of this money would want to direct it at protection agencies who benefit society by being good at supporting their clients to avoid crime.
    - It’s obvious why lawyers don’t try to prevent their clients from being criminals in the current legal system: lawyers are paid when their clients commit crimes and need to be represented in court. If lawyers were liable for damages when their clients committed crimes they might behave differently. The latter scenario is closer to what I’m imagining imposing on protection agencies in a “Zuthocracy”. Protection agencies would be paid to represent the interests of their clients by defending them against crime (like a private security guard) and defending their rights when prosecuted by others (like a lawyer), but they would also be liable for damages if their client was found guilty of crimes by a court (like… I dunno… an employer?).
  + It might be beneficial to separate the role of protector and rehabilitator/educator, even if just for the benefit of specialization.
    - It does seem as though a protection agency with an incentive to avoid their clients getting in trouble with the law might invest more resources in concealing their clients’ illegal activities than preventing them, especially if their main area of expertise is law rather than teaching.
  + But why would someone pay to have someone prevent them (even if just by psychological influence) from disobeying the law?
    - And if someone is poor (and thus more likely to resort to law-breaking) they will not be able to afford to pay for a “rehabilitator” even if they wanted to.
    - As in the case of paying a protection agency though, this service could be funded for them by a charity.
  + I began to consider the concepts of rehabilitation and education together before – perhaps that could be taken much further.
    - It seems clear that an educational institution whose income from a student is determined by that student’s lifetime income, would have an incentive to teach the student to avoid crime.
    - But the existence of such an incentive covering some students does not mean it will be useful to cater for all people. If someone is already a criminal, why would an educational institution invest resources in them over someone with more chance of being successful in life?
    - It is clear that some deliberate modification of the incentive environment would be required to cause any profit-oriented entity to invest resources in training someone with less expected potential for earning than alternatives. The question is, what to modify?
  + This is an issue I want to come back to (if only there were more time!)

# Tue 7-Feb-12

* Names need to be unique in their own scope and every overlapping scope
  + Scopes overlap with their superscopes and subscopes
  + The global scope overlaps all other scopes

# Tue 14-Feb-12

* Write a book about the Birth of Zutha from a historic perspective
  + I think it would be cool to write a book that reads like a history book written in, say 2025, about the birth of Zutha in, say 2015
  + Actually, instead of a book format, it should take the form of an online wiki that is purportedly written by the Zutha community. It should pretend to be looking back from the future, having had years of work in the intervening years between 2015 and the pretend present. However it could actually be a real Zutha hosted wiki written now (or as soon as I have a functioning Zutha.net up and running), imagining the events that might unfold in the future.
  + The prelude should say something like “for those countries that still have old-fashioned copyright laws, this document is published under a copy-left license”
  + There should be an afterward that discusses the actual present in relation to Zutha's evolution and invites readers to participate in making the vision of Zutha become a reality.
  + I think this medium could be a powerful way to illustrate what Zutha could be.
    - It would allow the narrator to describe a historic context for events and even reflect on the imagined repercussions of those events in the time between the events of the book and the fictional present.
    - The personal history of characters could be described in a condensed form without any need to pretend that anything other than the part of their history that matters for current events is being included (as is somewhat necessary in fictional stories).
    - Technologies could be described for the benefit of the lay reader in the way that popular non-fiction books describe things like the Internet MRI scanners
  + One disadvantage of this medium could be that it has less potential for immersion in the world as experienced by the characters themselves.
  + I suppose there is no reason why multiple mediums for imagining the future of Zutha couldn't be used simultaneously by different collaborative projects.
  + I could start writing in both a novel and history book format as separate projects and eventually open both projects up to collaboration on Zutha.
* Concentration of power is going to become more and more dangerous with the evolution of autonomous robots
  + A large centrally controlled network of automated machines will be more vulnerable both to hijacking and abuse of power
  + robots built to obey a central master will be much easier to turn to unethical pursuits than human soldiers are
  + The larger a system, the greater the potential impact of software bugs
  + Even if the current controller of a highly centralised system is trusted with that power, the fact that the reigns are all held in one place makes the entire system vulnerable to being co-opted by a far less trustworthy entity.
  + The system need not remain under the control of humans either. I think the fears of AI the abound in popular culture are not entirely unrealistic in this respect. If we design large scale systems to be centrally controlled by AI systems, than those entire systems become vulnerable to undesirable behaviour by those AI systems.
* A decentralised system could avoid this problems
  + Many independently controlled and designed systems that are instructed by their controllers to temporarily work together in a collaborative pursuit will robust to:
    - malfunctions anywhere in the system
    - attempts by any one entity (including AI systems, terrorists, governments etc.) to command the system to behave against the will of the majority of owners

# Sat 18-Feb-12

* Exploitation/Gaming for Reputation and Zuth accumulation
  + Fundamentally a high reputation user can act as the equivalent of the first Zutha user in seeding the reputation for an entire community of users he controls.
  + An algorithm will not (yet) be able to tell the difference between investment between a network of investments in dummy items and a valid network of investments based on the actual merit of items.
  + People will, however, be able to identify large-scale investment in dummy items and will be able to trace the trails of investment back to the seed investor. It should be easy to get community consensus in such a blatant case of gaming, to annihilate the entire fake network of users.
  + It would be necessary, however, for enough users to participate in this process to overpower whatever reputation the exploiter has accumulated before being detected.
  + It will be important, therefore, for such exploitation schemes to be found and neutralised early. Though if one does succeed in becoming very powerful, it is likely going to be easier to rally the support to eliminate it.
  + The fact that this kind of exploitation scheme will resemble, from the perspective of an algorithm examining the graph structure of investments, the initial seeding of reputation in the Zutha network, is not necessarily much of a problem. After the initial seeding of reputation has taken place, the rules can be changed to label as suspicious, any similar-looking isolated graph structures that arise later on.
  + An exploiter could seed his network of junk users and items from a large number of low-reputation users.
  + A fundamental paradigm of Zutha is that a group of low reputation users should be able to have as much influence as smaller number of high reputation users with equivalent net reputation, so an exploiter need not pool his accumulated reputation into a single user in order to use it. The exploiter could instead accumulate large numbers of separate small networks, each seeded by an independent user that at one point created an item of minor value.
  + The fundamental nature of the exploit is to keep all Zuth earned through their salary internal to a network of users they own. This way they gain the maximum possible multiplier effect of reinvesting their salary in their own reputation.
  + If we assume that all Zuth is invested immediately upon being earned, then any self-contained sub-network of users that owns x% of the Zuth (and reputation) in the network at time t0 can at best retain x% of the Zuth (and reputation) in the network by time t1, by investing only in items it owns. This is because all the other Zuth in the network experiences the same multiplier effect over the same period.
  + In practice, not all Zuth will be invested as soon as it is earned, which means its multiplying capacity will be left on the table. Thus an exploiter that re-invests optimally within its sub-network could increase its proportion of the Zuth in the network over time.
  + This seems to me to be a fundamental vulnerability because it is becoming increasingly clear to me that arbitrarily sophisticated sub-networks of colluding users are possible and they need not behave in obviously suspicious ways, such as by investing in blatant junk items or investing exclusively in a self-contained network. A sophisticated attacker could allow Zuth to leave the network of users it owns, but still ensure that less Zuth leaves its network than comes in from outside. The fact that more Zuth flows into a sub-network than flows out cannot alone be grounds fore suspicion because this must be the case for about half of all sub-networks. The only way to identify the sub-network as a parasite would be to determine that the value being created by it is obviously less than that of other sub-networks that are similarly profitable. This could be very difficult when there is so much scope for “dark” areas of the knowledgebase that could conceal the self-contained flow of Zuth through items that would not merit their investment under greater scrutiny. An attacker could conceivably pretend to be a community from an obscure language group and create junk items in a language no one else understands.
  + Pathological behaviour need not be executed by an attacker. A perfectly legitimate community could end up behaving like a parasite by focussing on investing Zuth as quickly as possible and keeping all investment self-contained within their community in order to benefit as much as possible from the multiplier effect.
  + The fundamental problem is that there is an incentive to keep Zuth “local” and that a competitive (though pathological) advantage can be attained simply by being faster than the rest of the community at reinvesting your Zuth.
  + There would be a tendency for Zuth and reputation to become concentrated among networks of users that invest their Zuth the fastest. This is a text-book example of an unintended consequence of a system.
  + It seems clear to me now that rapid re-investment of Zuth must carry no systemic advantage.
  + This could be achieved by making the multiplier effect apply regardless of whether Zuth is invested or not. So a user's salary would be determined by the sum of their total reputation and their holdings of Zuth.
  + This would prevent a sub-network from gaining control of an increasing proportion of the Zuth in the network simply by re-investing it faster. Instead, the best they could achieve would be to break even in terms of percentage of Zuth controlled, which would be exactly as impressive as a user who does absolutely nothing... except for the fact that a sophisticated self-contained network could convert all of their Zuth into reputation, but a hoarder could not. Provided that the percentage of Zuth hoarded does not increase over time, this would not allow the sub-network to accumulate disproportionate influence, but it could be enough to create an incentive for communities to hoard, especially for a loss-averse species like human beings.
  + If the items being invested in in a hoarding sub-network are sufficiently undeserving that they attract attention in the form of negative investment, then the hoarders may begin to lose ground as their hoarded reputation is negated by the negative investment and is thus unavailable to contribute to the multiplier effect.
  + The ability of the community to combat blatant hoarding should be encouraged and facilitated by tools such as group-criticising items that allow the negative sentiment of the community towards the hoarders (or attackers depending on the severity) to be concentrated in a single item. Such an item will be more attractive to invest in because it will be more likely to accrue a profitable amount of future investment than all the obscure junk items that have been used by the hoarders to pass reputation amongst themselves. The users that have been involved in the hoarding conspiracy can be added as owners to the group-criticising item as they are identified.
  + Even if obvious cases of hoarding can be identified and punished, however, the incentive will remain for communities to encourage each other to invest “locally” so as to keep the influence within the community (yes I'm using the quoted “locally” because of the ironic analogy with the commercial “keep the money local” fallacy). Unfortunately, unlike in the commercial scenario, which involves trade of money for something of (theoretically) equal value, there really is a rational reason to hoard Zuth within a community because investing in items owned by strangers implies a direct loss of potential reputation to a stranger. Actually a community can gain reputation by investing in outsiders' items, first by making a profit in Zuth from a good investment and then investing that profit back in the community to use as reputation. But if most people can be deceived by the local trade fallacy, there is hardly much hope that this more convoluted link between outside investment and self-interest will become the common wisdom.
  + I think the separation between reputation and Zuth is largely responsible for these pathological incentives and perceived incentives
  + Merging the concept of Zuth with reputation seems like an elegant idea, but the greatest barrier I see is how to reconcile domain-specific reputation (especially where reputation domains can be overlapping) with a one-dimensional measure of Zuth.
* Merging Zuth and Reputation
  + reputation domains form a tree structure.
  + Zuth is divided into a discrete set of domain-specific buckets
  + converting between adjacent reputation domains is costly. For example, it might cost 120 Physics Zuth to buy 100 Science Zuth and 120 Science Zuth to buy 100 Chemistry Zuth
  + Items are associated with one or more domains in specified proportions
  + Purchasing an investment requires making whatever conversions of Zuth are required to buy investments in the target item's domain(s).
  + When you sell an investment you gain Zuth in the domain(s) of the item (in the proportions that apply to that item)
  + When someone buys an investment in an item you own, you gain Zuth in the same proportions and domains as what was paid by the investor
  + When someone sells an investment in an item you own, you lose Zuth in the same proportions and domains as what was gained by the seller
  + If your Zuth account becomes negative in a domain, you can not make investments that require Zuth in that domain, or transfer Zuth through that domain until you have brought its balance back into the positive.
    - This would prevent someone who is flamed excessively for incompetence in a specialised domain from having their reputation dissolve, even in unrelated fields
    - It would still allow serious crimes to be severely punished however because the punishment could occur in a reputation domain like “trustworthiness” that is a required domain for investing a large proportion of items. A user that continually abuses trust would have to “spend” their reputation in other domains to replenish their trust reputation so that they can invest in items that require the trust reputation domain. Thus most of a user's influence could be wiped out if suffer too much criticism on the trust front.
  + It will be important that new and unsophisticated users do not have to concern themselves with decisions about which conversions to make in order to be able to make an investment. This would ideally be implemented as configurable preferences in the client software for making automatic conversion decisions at investment purchase time.
  + Actually, it would be much better if no one had to worry about micro-managing conversions of Zuth between reputation domains, but I'm not sure if that will be possible.
    - If conversion is costly in both directions then it matters from which side (above or below in the hierarchy) you convert Zuth from for each investment, because your distribution of Zuth across domains will change the relative cost of different investments.
    - Even if it only costs to go down the hierarchy (from general to specific), which probably makes sense actually, then it still matters which pools of reputation you draw from for each investment because it will affect which specialised domains remain cheap to invest in in the future. For example, if you want to buy an investment in the Universal domain, it will cost the same in terms of units of Zuth regardless of which pool of Zuth you draw on (because the Universal domain is at the top of the hierarchy), but if you use up all your SciFi Zuth for this investment, and then find yourself wanting to invest in a SciFi novel the next day, you will have to make an expensive conversion from another reputation domain, whereas if you had chosen a different source domain for the investment in the Universal item, the SciFi investment would not have required any conversions.
  + I think it will be practical to design algorithms that do a better job than most humans could at predicting which conversions are most optimal, based on a user's past investment behaviour.
    - Users should be encouraged to leave this algorithm to itself because meddling in such decisions will likely only lead to frustration and regret for most people.
    - Client writers can compete with each other to provide the best Zuth conversion scheduling algorithms and users can evaluate them based on experience or expert reviews in the same way that people evaluate sporting equipment or collaborative filtering services.
    - The need to make decisions about Zuth conversion will just become part of the competitive environment and expert service providers will compete to provide the best tools for that environment
  + The user interface for making investments (for most users) should be something like a slider that updates two different numbers as it is moved; one representing the size of the investment purchased and the other showing how much it will cost the user in total Zuth (measured in the Universal Domain, which costs zero to convert to from any other domain).
  + Users are paid a salary in proportion to their Reputation/Zuth (as measured in the Universal Domain). Their reputation in all domains increases by the same proportion.
    - This means the salary mechanics are identical to simple inflation of the “Zuth” supply.
    - Zuth will have no concept of debts or exchange rates with other goods, so the inflation rate can be completely arbitrary – though it should probably be proportional to the rate of expansion of the Zutha economy so that the perceived inflation rate stays roughly constant.
    - If a user has a negative balance in a domain, the size of this debt will grow over time, just as with standard inflation.
  + Conversion of Zuth between domains should be required infrequently.
    - The purpose of the conversion cost is to encourage users to express their opinions mostly in the domains they have earned their reputations for.
    - It will nonetheless be important for the integrity of the system, that a large group of non-specialists are able to step in and overrule a group of experts in extreme circumstances, especially if the domain has been “captured” by a non-representative group of “experts” who have begun to abuse their authority and discriminate against competing experts in their field.
    - Conversion will also be necessary to pay back debts in important general domains like “trustworthiness” or “impartiality”
  + It doesn't seem right that a user can “buy” reputation in domains they never earned it in by conversion from other domains.
    - For example, should a user be able to purchase a high reputation in the Trustworthiness domain using a reputation earned in the Diplomacy domain?
    - Furthermore, should it cost a user reputation in the Trustworthiness domain to increase another user's reputation in the Trustworthiness domain?
    - Both seem both unfair and inappropriate
    - Paying in trustworthiness score to increase someone else's trustworthiness score seems unfair, but it might reasonably be reconcilable as necessary to avoid the system vulnerabilities and pathological incentives we discussed earlier.
    - It does seem pretty nonsensical, however that someone should be able to purchase reputation in a field in which they have not earned it using reputation from a completely unrelated field
    - It would seem desirable to try to separate the capacity to give reputation to others in a particular domain from the capacity to give yourself reputation in that domain. But this faces exactly the same problems as the attempt to separate wealth in Zuth from actual reputation – in fact it is just the domain-specific analogue of exactly the same thing.
    - Unless I discover an entirely new paradigm, I think the only way to deal with this problem is to promote a more blurry definition of reputation. Instead of a reputation score having a direct link to a person's personality or skill-set, it will simply be an asset which is more expensive the less honest it is. This will allow the macro-level intentions of the reputation system to continue to work, but it will nonetheless be very confusing to most people. It's a dilemma.
* Implications of Zuth/Reputation integration for Zuth's interoperability (read: “capacity for being tainted”) with real world money.
  + Zuth now seems more like a conventional currency because it is both the medium of exchange and the holder of value
  + … yet it is still not exchanged, only invested. When you purchase an investment, it is a zero sum transaction in that the owner of the item gains the same amount of Zuth that you (temporarily) lose. Yet the transaction is reversible. You can sell the investment and immediately take that much Zuth back off the item owner.
  + Real money transactions are not so reversible. They can sometimes be reversed by intermediaries or legal force, but they are certainly far stickier transactions than investments of Zuth
  + This distinction makes for a difference in kind between the two currencies, and I think it will be quite effective in keeping them separated.
  + Ironically, this is an aspect of the dynamics of Zuth that I am only noticing now as I ponder a change (the merging of Zuth and reputation) that I thought would make Zuth more vulnerable to interoperability with the dollar.
  + The friction, of course, between the two currencies, arises because of an imbalance in confidence between the two parties of a trade. The buyer of real world money can be pretty confident of retaining his share of the deal as soon as the money has been deposited into his account. The buyer of Zuth, on the other hand, can at no point be sure that Zuth invested in an item of theirs will remain invested.
  + Actually perhaps there is a way to irreversibly exchange Zuth: invest a large amount of Zuth on top of a small investment by the intended recipient. The recipient then sells his investment and takes some value permanently from the payer's investment. This is of course no different from what was possible before; it's just that now this kind of transaction gains the buyer something which is both reputation and the ability to influence reputation in the same currency, whereas, before, the buyer would have had to invest the purchased Zuth into another user's item in order to make use of the kind of influence enabled by reputation. The whole point of abandoning the separation between Zuth and reputation was that this conversion was not really all that difficult.
  + The kind of transactions via mutual investment in an item that will permanently transfer ownership of Zuth from one user to another should be pretty difficult to pull off both safely and undetectably. If the transaction occurs in an empty investment market (an item with no investments in it yet), then the pattern that indicates a direct transfer from one user to another will be obvious. If, in the interests of obscurity, the transaction takes place in a busy market, then there will be a significant risk that the seller of Zuth will lose a large part of their temporary investment before they can fully sell it again. The size of investment required to transfer a significant amount of Zuth to a specific user will also be very large because investments in an already high-reputation item need much more Zuth invested on top of them to grow substantially in value. I suspect it will also become a popular practice to start selling whenever you notice the tell-tale pattern in investments above yours of small investment immediately followed by a huge investment, followed by immediate sale of the small investment. You'll be able to extract substantial value from the huge investment while its owner is still desperately trying to sell it off (which takes longer for larger investments)
  + The fact that the safer transactions (in empty markets) will be so easy to track, will make easy targets of users who try to buy reputation in this way. I'm sure there will be plenty of users who find this behaviour distasteful enough to track down the abusers (through however many layers of obfuscated transactions they try to conceal their tracks with) and punish them severely. This risk of paying for reputation, only to lose it again, will contribute to the unattractiveness of the transaction.
  + There is another way that Zuth/Reputation will be transferable. In order for control of the stock of Zuth to be distributed, the owners of items must be the ones who gain from the inflation of Zuth. So a seller of Zuth could invest in an item owned by the buyer and wait until the buyer has accrued the desired transaction amount through accumulation of “interest” (previously called salary or inflation), before asking for the payment of real money from the Zuth buyer. This would be a very slow process, would also be either obvious (especially in hindsight, as confirmation of suspected suspicious activity) or dangerous (if the investment was made in an active market), and more value would be lost by the seller than gained by the buyer, provided the seller is capable of earning greater return on investment than the rate of inflation/interest.
  + Fundamentally though, someone with a lot of money will be able to buy reputation. Surprise, Surprise! This is a fundamental, if unfortunate, implication of the nature of money – it buys things, including influence. The only variable that can be controlled is cost, but the cost can be made very high when things like reputation and law are part of the equation.
  + I think Zuth will be costly enough to purchase and to sell that the integrity of the reputation system as a whole will remain pretty robust.
  + Even if Zuth can be purchased, the proportion of Zuth in the Zutha economy that is potentially available for sale should remain small, provided the community is successful in maintaining its integrity and keeping a large proportion of influence among people who would not sell their reputation for money even if the selfish cost could be kept low. This sounds like a circular statement: that Zutha will maintain its integrity provided that Zutha maintains its integrity. But it just means that the equilibrium point (of honest versus corruptible users) can be kept stable at a high percentage of honest users.

# Sun 19-Feb-12

* An item owner must only gain Zuth/Reputation from the interest on the Zuth invested in items he owns. Otherwise he could infinitely re-invest in his own items (regardless of how much indirection is required).
* What happens to the permissions system now that Reputation is a spendable currency?
  + Performing an action could require the possession of a minimum amount or Zuth
  + If you think an action might be challenged, you can commit an arbitrary amount of Zuth to put more force behind your decision. Another user (or users) that want to challenge your decision would have to do so by committing more Zuth than you have.
  + This can be visualised like a kind of tug-of-war, with Zuth acting like force.
  + If no Zuth is committed to maintaining a particular state (like the state of a document) then no force (no Zuth) is required to change it
  + This system can completely replace the current permission-levels system. Instead of requiring a minimum reputation to change the permission-level or the permission-level lock, you just have to commit some amount of Zuth to keep an item locked in a particular state.
  + If a wealthy user puts a strong lock on an item that another wealthy user thinks is not justified, he can oppose the lock by committing Zuth to its opposite (just like negative investment). A poorer user can then be the decider by committing enough Zuth to make up the difference between the two large commits.
  + If there are multiple states to decide between (which there always will be), then there will be multiple separate buckets available to commit Zuth to both positively or negatively. Whichever bucket has the most Zuth in it will be the active state.
  + The permission system and decision markets have just been merged. How elegant!
  + I can't see any reason why decision markets shouldn't behave just like investment markets in allowing profit to be made by predicting the decisions of others, or loss to be made by being too slow to recognise the changing tide of opinion.
  + The only difference between decision markets and investment markets should be that decision markets have no owner, and thus interest continues to accrue to the owners of the investments.
* I could go back to defining reputation as amount invested in items you own, now
  + Reputation is no longer tied to influence, it would only be a measure of your Zuth-accruing “assets”
  + Only, Zuth itself would be an asset of equal value in accruing Zuth.
  + Reputation could be 'stock of Zuth held' + 'Zuth invested in items you own' + 'Zuth invested in decision markets'
  + This doesn't help with the confusion problem about what Reputation is, though. It will still be possible to purchase Reputation in any domain (at various prices depending on your current distribution of reputation).
  + Reputation will just be the measure for determining how much interest each user gets each time-period
* It may no longer be important to try to limit the ability of a user to influence another user's reputation.
  + It seems pretty clear that any attempt is essentially futile anyway, because it will always be possible to create arbitrarily complex networks of users that collude to increase each others' reputation
  + This is no longer meaningful, however, because giving another user reputation implies losing the same amount yourself. So you are most welcome to share the reputation you have earned among multiple users, but you gain nothing from it.
  + This will make calculation of Reputation much easier, because it will no longer be necessary to do the complex calculations required to limit, through arbitrary degrees of indirection, the influence of user A on user B's reputation to 10% of A's reputation
* Every state that can be changed should be a kind of document with versions
  + Every version of every document can have an associated decision market
  + An empty decision market is equivalent to the decision market item not having been created yet. In both cases, every new version of the document immediately becomes the active one
  + Changing some states will require the consensus of two different documents. For example, changing an Authorship association will require modifying both the Book's Author association field and the author's Books association field.
  + Actually, I think the unit of modification for changing roles in an association should be the association field set. There is really no sense in blocking users from adding associations to an item if they could create the same desired association by modifying an existing association's roles.
  + An association's roles should be immutable. Changing an association's roles should require deleting the existing association and creating a new one with the different roles
  + The decision market on an Association should apply to deletion of the association and modification of its properties
  + There should be separate decision markets for each property inside an association (or at least the option to create such fine-grained decision markets, though the need is likely to be rare for association properties, since most associations will have at most one property)
  + There should also be a separate decision market just for the delete action
    - No, deletion should simply be a special version (version 0), which can have a negative weight placed on it to make it difficult to revert to it
  + Every change to an item's properties (but not its associations) should change its version
    - A weight can be placed on a particular item version. For any of its properties to be changed, then, the new item version created by the change proposal must be given the most weight of any version for it to become active.
  + If two versions have equal weight, the latter one wins
  + An item version will be linked with the parallel versions of its properties
    - For any version to become active, it and all its linked versions must be the highest (or equally highest) weighted version in their version chain
  + Similar to the link between an item version and its property versions, an association field set version is linked to the versions of the association field sets of all the other roles in the association.
* There is no longer a way to have default permission-levels
  + This may be a good thing. Default permission level suffers from being a very arbitrary number. It would also be quite possible for it to be made too high such that instances that inherit it would be effectively immutable.
  + The decision market system, on the contrary, does not involve arbitrary numbers, but directly measures the strength of will (weighted by reputation) behind each possible state.
  + It will now be important for the community to be aware of the importance of putting weight behind important states, though, such as property declaration association field sets of type items
* Linked versions should actually be represented by a single “revision” item with a single decision weight associated with it
  + For a revision to be active, its weight must exceed the weight of any revision it supersedes
  + This system looks like it could elegantly integrate with the hash chain network for ensuring network integrity
  + But I think it removes the ability to specify a higher weight for modification of a specific property than for modification of the item as a whole
  + So I think weights will have to remain associated with each state (item/property) that is changed
* Changing a state requires committing enough Zuth to counter all locks that affect that state
  + So changing an item property requires enough Zuth to counter the sum of the weight on the item and the weight on the property
  + Putting a weight on an item essentially adds that weight requirement to all changes to that item's properties
* Users will be notified when a revision they have invested in is superseded. Presumably, if they agree with the change, they will sell their investment in their revision.
  + In practice this should usually mean that the most recent user to make a change takes responsibility for maintaining the lock, and will be able to reduce the strength of the lock (once all prior revisions have been emptied) by reducing the amount of Zuth they commit
  + If a user had invested in the negative side of a previous revision in order to reduce the strength of the lock, they may decide to sell this investment if they think the new revision has been given a sufficiently weak lock, or, if not, they may transfer their investment into the negative side of the new revision.
  + A script could be written to automate the behaviour of maintaining a field below a desired maximum lock-level. Users could allocate a maximum amount of Zuth they are willing to spend to keep the lock level below this amount.
  + Such scripts could be integrated into client software and made accessible as options in the user interface. The scripts would be authorised by the user, but would be executed by hosts and would thus not require the user to be connected to the network for them to work.
* An item revision will contain no data, only links to one or more property revisions and a new decision market
* Just as with the mercurial revision system, revisions should be labelled internally with their ZID rather than a serial number, but a serial number can be calculated by client software for the convenience of humans.

# Mon 20-Feb-12

* The concepts of Item Attributes and Reputation Domains could be merged
  + Attributes like Objectivity and Humour could both be Reputation Domains for which an Item gets credit
  + Items would have a separate pair of Investment Pools for each Reputation Domain associated with them
  + Users could add additional Reputation Domains to Items, creating a pair of Investment Pools in the process
  + This approach also has the benefit of eliminating the need to determine an Item's Reputation Domains (and the proportions of each) by a separate process.
* How should the Reputation Domain(s) of Decision Markets be determined?
  + I think it still makes sense for a single Decision Market to require Reputation in multiple domains in varying proportions
  + The Reputation Domains required for decisions should be determined for whole categories of decisions at once
  + The process of determining the relative proportions of different domains of Reputation for a particular kind of decision should itself be determined by a Decision Market. Each candidate domain would have its own pair of Investment Pools and the proportions of each domain would be determined by the relative sizes of each non-negative pair of Investment Pools.
  + Each Investment in a Decision Market will conceptually consist of some combination of one or more different kinds of Zuth. Investing in it may thus require the conversion of Zuth from other domains to make up the requisite proportions of different domains.

# Fri 24-Feb-12

* The community should vote to decide community norms - that is, protocols that only social expectations rather than programmed rules will enforce.
  + e.g. should book publishers or other media distributors receive any credit (ownership) for digital goods?
* The normal way for clients to perform queries (view requests) should be to download all the necessary data to perform the query into their private database cache.
  + Most clients are likely to have plenty of computing resources and enough space to perform most queries.
  + This will distribute the load of query computation significantly and also put query processing time in the hands of the user (and the innovation of client-makers).
  + It will make requests much faster.
    - Only one request-response cycle will be needed for each host group that needs to be contacted and all hosts will be contacted in parallel.
    - Caching can be used far more productively when requests are decomposed into many common requests.
      * The first node along the message path that has the requested data can return it immediately (the copy signed by the controller group that owns the data that is).
      * If a node returns an old copy of the data (older than the requester wants), it should pass on the query further towards the target controller group with an an added pieced of data: the age of the data it returned to the requester.
      * If a node has the desired data with an age between the latest data so far returned and the requested age, then it should return its data and also pass on the request, updating the latest-returned data in the message it sends onward.
      * If an intermediate node's copy is at least as recent as the age threshold specified in the request, then they need not send the message onward.
    - Clients can construct the result of the query in real time for the user as data comes in, updating the age of the data in real time as more and more recent data arrives.
  + Queries should be broken up into atomic requests that can be answered by a stock response from a single controller group.
    - For example the query to find “US companies with a Chinese CEO” might make the following atomic requests:
      * All items that “are Chinese” that is, play the role National in the has-nationality association with the Chinese Nationality. The controller group of the “Chinese (Nationality)” item would know this list.
      * All items that are CEOs (known by the CEO item controllers).
      * All items located in the United States (the United States item controllers would know this list).
      * All instances of Company (known by the Company item controllers).
      * The sets of items: “US Companies” and “Chinese CEOs” would then be calculated. Depending on which set is smaller, either the CEO of each US Company would be looked up, or the Company of each Chinese CEO would be looked up.
      * Whatever specific information about each US Company with a Chinese CEO would then be looked up
    - This example requires 3 rounds of queries, but is a particularly complex query so this is not unreasonable.
    - It may seem like overkill to collect in the client, such huge sets as all items located in the USA and all items that are Chinese, yet these sets will simply be a list of short ZIDs so even millions of items could be stored in a few MB. The number of items located in the USA could eventually number in the billions or trillions, though. For sets that big, query planning algorithms would have to devise alternate ways to perform the logic. For example, it could ask for all the Companies and their located-in Regions and then find each Company whose location is known to be within the USA (which may involve looking up some previously unknown Region items).
    - There is a major benefit for caching of passing around huge standard sets of items. The larger the atomic units that are passed around, the fewer of them there will be and thus the more likely each one is to be cached recently by a given node.
  + I'm going to start using the term 'fieldset' instead of 'data chunk' (or whatever else I've used) to refer to an atomic piece of data that is known by a single item controller group.
  + The amount of data that needs to be passed around can be enormously reduced by passing sets around with their version information
    - When a client requests a fieldset that he previously possessed no prior versions of, he will receive the entire version history of the set - which, given that the vast majority of changes to the semantic graph will involve addition rather than subtraction of information, will mostly consist of a set of ZIDs grouped by the date they were added to the set.
    - When the client subsequently requests an update to the set, it will include the timestamp of the latest revision it possesses. It will then be returned only the revisions after the latest one it possesses.
    - The authenticity of a revision will be verified by a signature on its hash. Thus a client can compile the fieldset that results from factoring in all the newest revisions, and then verify the complete set by checking whether its hash matches the signature returned in the response to its request.
* Every Item should be assigned to an archive item based on the hash of its ZID
  + For items that are not archived, the archive manager need only record the item's type alongside its ZID
  + An Item's archive manager can be consulted to find out an Item's type so that the right selection algorithm can be used to locate its controller group
* A subset of a fieldset can be verified as valid without the availability of the entire fieldset.
  + Hosts could sign a revision of a fieldset by signing a Merkle Tree of all its members.
    - First the hash of all member ZIDs would be calculated
    - Each pair of level-1 hashes would be hashed, forming level-2
    - Each pair of level-2 hashes would be hashed, forming level-3 and so on until only one hash is left
    - To verify a subset of a fieldset, each element of the subset and its pair would need to be known, as well as all the intermediate hashes that cannot be derived from the known elements' ZIDs.
    - The amount of data required to verify that a subset of size m of a fieldset of size n is valid would be of order m\*log(n). It would approach n as m approaches n though.
  + This would mean that a client could ask a 3rd party host (without redundancy) to run a query for it and return results (along with some extra hashes), that could be verified as valid despite the fact that the entire fieldsets that were actually signed by controller groups are not known by the client.
  + There is a problem with this, though: the Merkle Tree only makes it possible to prove that a given item is currently a member of a particular fieldset; it doesn't allow any proof that the returned set is complete though. For example, if a client asked for all books written in Israel and it received a reply from a 3rd party host, it could verify that all the returned items are Book items (by verifying the partially, but sufficiently complete Merkle Tree and checking that the signed document associated with the top-level hash labels the fieldset as instances of Book) and it could do the same to verify that all the Books were written in Israel, but it would not be able to verify that the set of items returned constitute all items currently in the Zutha knowledgebase that satisfy the query conditions.
  + I think the only way to ensure completeness of 3rd party responses to queries will be through the standard method of redundancy (multiple parallel responses from independent hosts).
  + It would also be more efficient for fieldset hashes to be constructed as a chain of hashes - one hash for each revision – instead of necessitating the construction of a whole new Merkle Tree with each revision.
    - A fieldset's hash could be based on the whole revision history since we are going to pass around whole revision histories (or the missing parts of them) instead of flat fieldsets.

# Fri 2-Mar-12

* Distributed insurance
  + People form layered (multi-level) communities that share risk.
  + A community's liability to the larger community it is a member of would depend on its relative drain on the larger community - determined by the frequency and size of claims made by members.
  + Every individual would pay a constant percentage of all the claims made to the entire global network (thus, by the law of large numbers, the amount they pay would effectively depend only on their total percentage of liability). The percentage they pay would be the product of the liability percentages of each level of community they are a member of, to the level above it.
  + When a member's risk (claim size/frequency) increases, their liability should increase by a smaller factor. The slack is absorbed by higher layers of communities.
  + The smallest scale communities would be able to affect the amount they pay most effectively by minimising the risk of claims among its own members (people in this case). For health, this would entail encouraging healthier lifestyles among members, for example.
  + Higher level communities would have an incentive to encourage their member communities to minimise their risk, which might entail sharing knowledge and influence strategies.
  + The arrangement of communities should be determined by rules rather than choice. Otherwise, communities would have an incentive to simply evict high risk members.
    - An example of a good selection rule would be for doctors and their patients to be grouped together.
    - Grouping by geography would also make sense as there are risks that are correlated by geography that could be reduced by the collective efforts of the local community.
  + It should be possible to be a member of more than one community. You would be assigned a percentage of membership to each community and your risk and liability to each community would be determined by your relative proportion of membership in that community.
    - So, for example, a doctor might be 60% a member of his practice's community and 40% a member of his suburb's community. If his total risk metric would be 200, then his liability to his practice community would be determined based on a risk metric of 120, and his liability to his suburb community would be determined using a risk metric of 80.
  + The risk metric should be determined by claim rates and automated statistical analysis alone. It should not require the input of subjective analysis.
    - Every person should start with some globally constant initial risk, determined by the global average claim rate (in dollars per year, say)
    - If a person, X, makes a claim of $100,000 for an expensive open-heart surgery operation, their risk factor should increase to reflect the expected future claim amount per year of someone who has just had such an operation, based only on data known by the automated system.
    - If X makes no claims for the next 5 years, X's risk will decrease to the expected claim rate per year in the future of someone who has had one $100,000 heart operation 5 years ago and none since.
    - If X makes another large claim associated with heart problems the next year, X's risk will increase to reflect the expected claim amount per year of someone who has had two expensive heart operations in the last 2 years.
  + If X's risk increases from $1000/year to $10,000/year in a community, Y, that had a total risk of $11,000/year initially (and now has total risk of $20,000/year), X's liability should increase by some logarithmic (or maybe a power law) function of X's risk.
    - If we use the log function to determine relative liability, and all other members of Y have a risk of $100/year, then X's liability would initially be:
      * log(1000)/(log(1000)+log(100)\*(10,000/100)) = 1.48%
      * everyone would each have liability: log(100)/(log(1000)+log(100)\*(10,000/100)) = 0.985%
      * and 1 \* 1.48 + 100\*0.9852% = 100%
    - After X's risk increases to $10,000/year, the relative liabilities would become:
      * liability(X) = log(10,000)/(log(10,000)+log(100)\*(10,000/100) = 1.961%
      * liability(others) = log(100)/(log(10,000)+log(100)\*(10,000/100)) = 0.98039%
    - The liability of the whole Y community would also increase according to the log of its increase in risk from 11,000 to 20,000
    - If Y is a member of larger community Z which initially had a total risk of $11 million/year, made up of 1000 communities with 11,000 risk each, then Y's initial liability to Z would have been:
      * liability(Y) = log(11,000)/(1000\*log(11,000)) = 0.10%
    - After X's risk increases from 1000 to 10,000 and Y's risk increases from 11,000 to 20,000, Y's liability increases to:
      * liability(Y) = log(20,000)/(log(20,000)+999\*log(11,000)) = 0.1064%
      * which is an increase of 6.4%
    - X's liability to Z would therefore increase from
      * liability\_Z1(X) = 0.0148\*0.001 = 0.00148% to
      * liability\_Z2(X) = 0.01961\*0.001064 = 0.00209%
      * which is an increase in 41.2%
    - the other members of Y would have their liabilities change from
      * liability\_Z1(Y-others) = 0.00985\*0.001 = 0.000985% to
      * liability\_Z2(Y-others) = 0.0098039\*0.001064 = 0.001043%
      * which is an increase of 5.9%
    - These numbers seem to be pretty appropriate.
      * X suffers a hefty increase in liability (equivalent of insurance premium) of 41%, but for an increased risk of 1000%
      * Each other member of Y suffers an increase in liability of 5.9% because of the increased risk now attributed to their community because of X. They are not absorbing anywhere near the whole increase in risk associated with X, but they each wear more of it than people in more distant communities. This gives them an incentive to focus on reducing the risk of people more closely related to them in the hierarchy of communities.
    - Note that the total liability of Y would increase by slightly more than the 5.9% calculated above, because the liability of the Z community would increase very slightly as a result of X's increased liability as well. This effect would be exceedingly small though. Z's risk would only increase by 11,009,000/11,000,000-1 = 0.082%. Z's liability would increase with respect to its parent community P, by:
      * delta liability\_P(Z) ~= log(11,009,000)/log(11,000,000) -1 = 0.005044%
  + Exploitation
    - The cost of this system for an individual of risk R is less than the collective cost of N individuals whose collective risk adds up to R. Thus a group of people who entered themselves in this insurance scheme under a single name would be able to get the same benefits for a much smaller cost.
      * This is why it is important that allocation to communities is automatic, based on pre-defined and global rules, rather than being manipulable by individuals or conspiring parties.
      * To enter the insurance scheme, a person would have to verify their identity to the Zutha community and they would be allocated to an insurance community based on details about them (such as where they live, which hospitals attend and who their current GP is etc.)
    - Individuals could make fraudulent claims and acquire a large payout and only pay a fraction of the cost of the increased risk that would be associated with them as a result.
      * This issue faces current insurance schemes as well. The only way minimise this problem, as far as I can see, is to have effective mechanisms for determining eligibility for payouts and detecting fraud.
      * It would surely be possible to design a claim-arbitrating system with better aligned incentives than those faced by current insurance corporations, though.
      * This is where a distributed justice system would need to come into play.
      * Courts that compete for endorsement by as many protection agencies (and indirectly the customers of those protection agencies) as possible, would rule on the eligibility of a claimant to receive an insurance payout.
      * There would be only one party in an insurance claim: the claimant, because the payer is the entire distributed community of communities that participates in the insurance scheme.
      * Thus the selection of a court to rule on the claim cannot be based on the usual method of consensus between two parties. Instead, it should be a random selection among available courts with probability proportional to the reputation of the court for ruling on this particular kind of insurance claim.
      * This system would provide an opportunity for start-up courts to get a chance to prove themselves
      * The incentive of the court is to balance cost with making a good ruling.
        + They are not directly paid by anyone involved in a case.
        + They are paid by a pool of money dedicated to funding courts that comes from Zutha earnings and donations. The proportion of this fund they receive is determined by their reputation in the legal reputation domains.
        + A court earns its income by earning its reputation, and it earns its reputation by ruling on cases in a way that meets broad approval.
        + If a court favours a claimant unreasonably, conscientious members of the community interested in the insurance issue will likely punish them for unjustly gifting money to favoured individuals from the Zutha community's purse.
        + If a court is obviously cruel to a claimant by denying a claim using dubious reasoning, then members of the community who care about the issue are likely to side with the claimant because this time their feeling of empathy with someone who could easily be them will be triggered.
        + In both cases, the users who vote on the court's reputation will be acting rationally from their own perspective; if too much money is given away to claimants for dubious cases, the cost of insurance will go up with little social gain, but if a court consistently fails to recognise legitimate claims, then the integrity of the entire insurance scheme is undermined. Since no one would want such a court to rule on their own insurance claim, they would want to vote the court “out of office”, as it were.
* Now that reputation is fully a currency that can easily be converted between reputation domains, what will stop people from voting themselves a share of reputation in a domain that earns real money from Zutha?
  + Well actually it was already possible to do this before, only now it is more obvious that it is possible, which I think is probably a good thing as it will make people more conscious of preventing the abuse of this ability.
  + Some examples of possible abuses:
    - Someone who has earned a reputation for making delicious dessert recipes could potentially create a junk item that they label a contribution to Zutha's specification and invest all their Zuth in that item.
    - This would give them some reputation in the Zutha Specification domain which would cause them to start earning an income in real money in proportion to the size of the investment they were able to make after the expensive conversion from Cooking to Zutha Specification domains.
    - To take an even more alarming example, a group of teenagers could get together and start their own court which they register on Zutha. They could use whatever reputation they can get their hands on from other domains and convert it into reputation in Zutha Law which might enable them to be assigned to hear a case.
  + The fact that conversion of reputation between distant domains is expensive will be an important, but not sufficient barrier to prevent such abuses.
    - People with a small enough reputation for their activities to attract little attention are unlikely to be able to accrue a substantial reputation in domains for which they have not earned it.
    - The conversion rate between Cooking and Zutha Specification, for example (which is the same as the conversion rate between the Universal Domain and Zutha Specification) might be something like 50:1 because it would involve going through several layers of domains like Reasoning, Computer Science, Specification Design, Zutha Architecture, Communication Skills etc. each of which might shave off 50% or more upon conversion from a higher level domain. ½^5=1/32 so 50:1 seems like it could be a realistic number.
    - If someone with a modest 1000 reputation in some domain totally unrelated to Zutha Specification Design bought themselves reputation in that domain, it would cost them their entire reputation to get 20 reputation in Zutha Spec. Design.
    - Worse than the immediate loss of net reputation from converting between distant domains would be how vulnerable you would become to losing that expensive purchase of undeserved reputation.
    - When users who are actually knowledgeable about Zutha Specification Design come across the (presumably junk) item that was invested in, they certainly be unappreciative and will probably oppose the investment with negative investments of their own. Since investing in items in the Zutha Specification Domain is costless to them, there is not much stopping this from happening and several forces inducing it to happen, including the pride of legitimate members of a reputation domain in keeping their reputation in that domain meaningful.
    - Once the cook's reputation in Zutha Specification Design goes negative they have no way of getting their investment back.
      * If they keep it it simply offsets the negative investments somewhat leaving them with a slightly less negative reputation in that domain.
      * If they sell the investment, it will be received initially as reputation in a domain that is currently negative, which means it will be completely absorbed offsetting the negative reputation balance and none will be left over to convert back into other domains or re-invest.
    - So someone that tried to earn an income from Zutha by buying reputation in a well-paid Zutha Contribution domain would likely earn very little before losing their expensively-bought reputation entirely.
    - But there is certainly a chance that someone could pull off acquiring an undeserved reputation with special privileges and keep it for long enough to cause problems. How do we stop a bunch of teenagers becoming a self-appointed court of law for example?
  + Reputation domains that provide important privileges, like the one for being a court of law (which I'm dubbing Zutha Law for now), should have transparency and peer review requirements in place before the special privileges become active.
    - The creation of a new Zutha Court for example, would require a minimum reputation in the Zutha Law domain and would require the endorsement of a much larger weight of Zuth in that domain, as well as a long and very high-exposure review period, before it would become active.
    - So, for example, a bunch of teenagers could vote on their own comments in the Zutha Law domain for a while to accrue sufficient reputation in that domain to create their own Court item. But for that Court item to become an active and meaningful representation of their membership of the Zutha court system, they would need to get their hands on a much larger amount of Zuth in the Zutha Law domain, as well as survive the scrutiny of the entire Zutha Law community for weeks or months during which time their self-investments in their Court item could easily be countered by users with much more power in that domain for which squashing their kangaroo court would be quite cheap.
* Zutha software platform paradigm
  + Generalise the GNU model of lots of specialised tools that take text on the command-line. Pass Zutha objects (items) around instead.
  + Programs will know what kinds of items they can act on and they can export this information as Zutha semantic data which could be read by client software to provide useful feedback to users as they interact with programs. As a simple example, a client could provide an auto-complete feature for sending commands to a program, analogous to an IDE.
  + As an example, consider a program that does automated renaming of files/items. It would take in a collection of items to rename (which could be created manually or by sending a query to a different program) and a program to process each name. It would then pass each item's name in turn to the name-processing program and change that item's name to the result. The name processing program might be constructed using a regex-processor program that takes a set of replacement regular expressions as input and outputs a program that applies those replacements to its input and returns the result.
  + Direct text input would be just one kind of message that could be sent to a program so programs already written with the Linux paradigm could be made available with this more primitive interface without any changes being necessary.
  + The best way to apply this paradigm would be to have all programs written in a standardised, Zuthanet integrated language, and hosted on the Zuthanet
    - The Zutha-language implementation of the program would act as the canonical form that is used to verify the correctness of Hosts that run it.
    - There may be multiple different optimised versions made of the canonical version, though, and hosts would be free to chose from among these if they trust that the one they choose is correct.
    - The implementation of the Zutha language on host machines would also involve lots of experimentation with optimising compilers. This would be a more scalable approach to achieving greater efficiency and would likely be the way most programs on the Zutha platform are run.
  + The experience of a user should run something like the following example:
    - Scenario: a user wants to create a movie clip of their favourite superhero with their own face replacing that of the original actor who played the superhero.
    - They search the Zuthanet for photos of their own face, possibly within a private, encrypted section of the Zuthanet that their client would have decrypted and made seamlessly available within the wider knowledegbase.
    - They search for a program that does automated face replacements in movies. There might be many. They choose the one with the highest reputation.
    - They read the summary of how the program works which is the default view of the item that represents the face-replacement program. It says it requires a pre-processed face object as input, as processed by a specific other program, which it links to.
    - The user goes to the face-construction program and it explains that it can take any image object as input (as it will use 3rd party programs to convert to the format it prefers).
    - They press the execute-program button (or uses whatever interface their client has employed for interacting with programs) and are presented with a series of questions, about what input parameters they want to give to the program. Their client reads the available input fields from the program's specification and presents them to the user according to the client's implementation and user preferences. The interaction might be presented as a GUI with traditional browser-like input fields, as an old-school command-line interaction, as a brain-machine-interface communication or whatever.
    - Some inputs may be numeric parameters; some might be entire configuration objects for tweaking the behaviour of the algorithm of which the user could create their own or select from configuration objects created by other users; one input field in this scenario will be the image to extract a face from. The user might enter this parameter by entering its ZID number, by graphically dragging the object from another part of their client's visual interface, or by recalling it from their short term memory and allowing their brain-machine-interface software to associate the thought with the recent object they were examining when they first stored that pattern of associations in their short-term memory.
    - Many parameters are likely to be optional in this case, and most of these should be packaged in configuration objects that allow the user to select from high-level functionality instead of needing to be exposed to the low level details of the algorithm.
    - In this case the user just enters their image into the program and selects all the default settings and is promptly returned a new object that represents the algorithm's attempt (using default settings) to extract the facial information from it. The user's client might display the returned object in some graphical way that the user can immediately interact with (by zooming, rotating and moving it around their display for example) or it might just show a simple textual link to the new Zutha item that represents the object.
    - The user now returns to the face-replacement program and executes it. When asked for a face object, they enter the one just created, perhaps by dragging it from where they left it hovering in another part of their visual interface.
    - They are also asked for a film clip to insert the face into. It turns out other people have already gone to the trouble of processing the film clip this user wants to use, so all they need to do is give the face-replacement program a meta-data rich film-clip item in the format it wants. If they had needed to process the film themselves, the face-replacement program might have suggested a program that could do that, or it might have offered to send the raw film clip to that program itself using the default settings so that the user wouldn't need to do anything except pass the raw film and photo to the face-replacement program (the same could be done with the face-extraction algorithm).
    - The face-replacement program does its work and returns a new video object with the user's own face in place of the superhero's. The object is stored as a semantic entity in the knowledgebase, possibly with a name the user gave it when running the program. The user's client is simply returned the ZID of the resulting object and can do with it whatever its makers have programmed it to do – such as display a floating interactive object in the user's display that the user can drag into containers and categories and into messages to send to other users.
  + Processing can either happen on the user's client or can be distributed across many hosts.
    - A user would need reputation in order to ask hosts to run resource intensive processing tasks for them, but with sufficient reputation they could get access to an on-demand distributed supercomputer.
  + Programs are passed around a bit like Javascript in today's web; as code that is automatically downloaded and processed without the user needing to take any action other than to request execution of a program they find. There would be none of the antiquated concept of “installation”, let alone compiling, adding directories to your classpath or editing of configuration files in a text editor.
  + Programs would not come bundled together as enormous monolithic monstrosities like Microsoft Office or Photoshop; every function would have a distinct identity and only the functions that are actually going to be executed would be downloaded by your client (if you are going to run the programs on your client that is – you might instead just send an execution request to the network and have the results returned to you).
  + Setting up an entire application server and getting all the different components to talk to each should, in most cases be a simple matter of finding a pre-configured environment made by someone else that is similar to what you want and downloading it to your client (which is your server connecting to the Zuthanet as a client).
    - You could then tweak your environment by modifying settings items that would appear in the context of their place in the Zuthanet, alongside documentation and a real-time channel of communication with the community.
    - Adding a component would involve simply referencing that component from the program that would call it.
    - For example, if you wanted to connect to Redis database from a Lift web application, you would simply create a semantic object in the Zuthanet representing a new Redis database and make the call to that Redis database in your program – not as a plain text name, but by making a programmatic reference to the Zutha object representing the Redis database.
    - You would specify as part of the Redis Database item that it is located on the machine you are configuring. This action would require the permission to modify your machine because it would be synonymous with actually putting the program on your machine.
    - The Redis code would be downloaded as it is needed by your machine and no further configuration would be needed on your part. You could modify Redis settings of course, but this would just be a matter of modifying setting items associated with the semantic Redis Database item you created.
    - The code you write that references your Redis Database would just be Zutha data (possibly private and seen only by your client, but still forming a seamless part of the Zuthanet from your perspective.) The link to your Redis database would be just like any other Zutha association. Your code would be labelled as being stored on the machine you are configuring, and so it would be.
    - Your entire configuration would be just a graph of interconnected items that you construct and make semantic links between. This semantic structure could be taken and used by other users in the same way you started with a pre-configured server environment.
    - In fact, the structure could be “installed” on multiple different machines at once simply by associating it with each of those machines.
    - If you wanted to modify something about one machine but not all the others, the configuration structure could be modified in the way that immutable data structures can be modified to minimise duplication of data without modifying any of it. So for example, a new top level configuration item would be created, and the items directly between it and the modified item would be duplicated, but everything else would simply be linked to by the new structure.
  + Every entity that we interact with could conceivably be virtualised as an item in the Zuthanet which we would interact with in proxy of the actual thing. Now that wouldn't be desirable in all cases (women come to mind for me for example, though there already appear to be plenty of people who find satisfaction in virtual sexual encounters), but in the case of scenarios like application server configuration, the ability to interact with more abstract objects than actual files on a hard drive, command-line arguments or TCP connections could be a blessing.

# Sat 3-Mar-12

* Distributed Insurance again
  + If a group's liability is determined by a decreasing function (like log or sqrt) of its total risk, then member of larger groups will pay much less than members of smaller groups
  + Consider a community Z that is a member of the global community G
    - Z has a population of 10,000 and a total risk of 100,000
    - G has a population of 1,010,000 comprised of Z and 1000 other communities of population 1000 each with a risk of 10,000
    - The average risk per person of every top-level community is the same (10 per person)
    - If Z's share of the liability is based on its total risk compared to that of the other communities than its share would be:
      * liability\_G(Z) = log(100,000)/(log(100,000)+1000\*log(10,000)) = 0.00124843945
      * liability\_G(others) = log(10,000)/(log(100,000)+1000\*log(10,000)) = 0.000998751561
      * 0.00124843945/0.000998751561 = 1.25
      * So Z would pay 1/4 more than each of the other communities despite being 10 times larger than them.
    - So each member of Z pays, on average, 0.125 times the amount paid by users on the other communities of G despite having the same average risk as all the members of the other communities.
  + It is going to be inevitable that communities differ in size significantly so this problem needs to be solved
  + The scaling function (log, sqrt etc.) could be applied to the per-member risk instead of the total risk of a community
    - So in the above scenario, Z's share of the liability of G could be:
      * liability\_G(Z) = log(10)\*10,000 / (1\*log(10)\*10,000 + 1000\*log(10)\*1000) = 0.0099009901
      * liability\_G(others) = log(10)\*1000 / (1\*log(10)\*10,000 + 1000\*log(10)\*1000) = 0.00099009901
      * this time Z's liability is 10 times larger as it should be and members of Z have the same liability as the member of the other communities of G
    - Now what happens if Z's risk per capita increases, let's say from 10 to 20?
      * liability\_G(Z) = log(20)\*10,000 / (1\*log(20)\*10,000 + 1000\*log(10)\*1000) = 0.012843206
      * liability\_G(others) = log(10)\*1000 / (1\*log(20)\*10,000 + 1000\*log(10)\*1000) = 0.000987156794
      * Z's liability is now 0.012843206/0.000987156794 = 13.0103 times that of the other members of G
      * Z's liability per capita is therefore 1.3 times that of the other members of G (because Z is 10 times more populous)
    - The total risk of G has now risen from 10,100,000 to 10,200,000. What happens to the absolute risk worn by each member of Z and Z' ?
      * (I'm calling the other members of G than Z, Z')
      * let's call the risk actually paid for by an entity its “loss”. This is distinct from its liability, which is its percentage share of the loss experienced by its parent, and risk, which is the risk it contributes to the pool.
      * The loss amounts before the increase in Z's risk (time 0) are as follows
        + loss\_0(Z) = 10,100,000 \* 0.0099009901 = 100,000
        + loss\_0(Z-member) = 100,000/10,000 = 10
        + loss\_0(Z') = 10,100,000 \* 0.00099009901 = 10,000
        + loss\_0(Z'-member) = 10,000/1000 = 10
        + So loss equals risk when every community has equal risk. That's as it should be.
      * After Z's risk increases from 100,000 to 200,000 (time 1), loss amounts become:
        + loss\_1(Z) = 10,200,000 \* 0.012843206 = 131,000.701
        + loss\_1(Z-member) = 131,000.701/10,000 = 13.10
        + loss\_1(Z') = 10,200,000 \* 0.000987156794 = 10,068.9993
        + loss\_1(Z'-member) = 10,068.9993/1000 = 10.069
      * So loss increases be a hefty 31% for Z-members as a result of their 100% increase in risk, but it also increases for all the other members of G by a smaller 0.69%.
      * So the group responsible for the increase in global risk, Z, pays a heftier percentage increase in loss, but they actually only pay 31,000 of the 100,000 unit increase in risk they cause. The other 69,000 is distributed across all the other members of G (Z'). Each community in Z' only experiences an increased loss of 0.69% though because there are 1000 communities of 1000 members (each with 10 loss already) to distribute the loss of 69,000 across.
    - This method seems to solve the problem of larger groups being favoured

# Sun 4-Mar-12

* Nodes can verify another node's possession of a large amount of data cheaply by inventing a unique hash function and asking the other node to hash the data with it. If the verifying node has possession of the data themselves then they can compare the returned hash with what they get themselves. If not, they can send the unique hash function to multiple independent nodes that are expected to possess the data. Assuming a majority of these nodes are not colluding, any odd response could be treated as an attempted bluff and could be punished.
  + Without such a verification policy, even independent hosts entrusted with low popularity items could end up forgetting data by their individual incentives. If a host was only required to sign off on data packages as they are passed to new entrants to the controller group, they could get away with not storing the data themselves and simply relying on another host to actually send the data itself to the new entrant. Clearly such a lenient model would be degenerate.
  + All that is required to create a new hashing algorithm is to add a unique salt to input data before applying the hash. The salt could simply be the current timeframe key, which cannot be predicted in advance.
  + Hosts could verify their possession of data by hashing it after appending the current timeframe key.
  + The best way for the verification process to work would be for the timeframe-specific hashes to be posted publicly as part of a periodic standard procedure.
  + If every host had to sign the same hash, then they could all copy the hash posted by the first. So the salt added before hashing should be host-specific; probably just the timeframe key and their public key. Hosts would have no reason to voluntarily post someone else's hash. If one did through an agreement then this would be no problem because all the hosts would still be taking responsibility for ensuring they can get access to the data (if they can get access to the correct timeframe specific and host specific hash then they can get access to the data required to generate it through the same channel.)
* Items could accumulate a lot of associated baggage in the form of history and auditing data. It would be inefficient to pass this immutable data around every timeframe to new entrants to controller groups.
  + The recent data would need to be examined and verified by the new entrants but the old data should be archived.
  + The top of the hash tree of old data would be part of the newer verification data so the veracity of the archive could always be confirmed by the live head of the auditing hash tree.
* Archives should have their own mechanisms to avoid tampering.
  + The hash of the whole archive should be protected by a fast moving controller group even though the holders of the data itself would change slowly.
  + An archive should be a bucket consisting of all archived items whose zid hash falls within a particular range.
  + The archive will have a slow-changing storage controller group and a fast changing auditing controller group.
  + The storage controller group would be required to post host and timeframe specific hashes to the auditing group each timeframe along with the simple hash which would only change when an item is added or removed from its archive.
  + The auditing group would simply keep a history of all the hashes. It would need to change quickly to ensure that no one can get control of it and alter its hash history to.
  + Control of all members of both the audit and storage group would be required to modify the archive. For example, to wipe all traces of an item from the archive, an attacker would need to have all the hosts that were in the archive's storage group at the time the item was added sign an archive that omits the item. They would then need to get all the hosts that have been in the storage group since to sign hashes of all the alternate archives that omit the censored item. Since the audit group has kept a hash tree of the correct progression of hashes, the attacker would need to control all the members of the audit group back to the time of archive of their target for censorship and have them sign the alternate hash tree consisting of the hashes they just had re-signed by the storage group members.
  + The event items that make up the revision history of archived items, including the one that changed the item's status to archived, would all be stored in different archives. This means there will be a trail that can lead to an attacker's fraudulent censorship of an item even if the attacker manages to remove access to the item by looking up its zid at the appropriate archive.
  + Audit groups should include signatures from the controller groups that participate in the event of archiving an item. In fact the audit group of an archive should simply form a part of the standard global event hash chain. The event that archives an item has bi-directional dependency with the archive state and the archive's audit group manages the archives event hash chain.
  + New entrants to an archive storage group should verify the signed hashes of the other members. It would be a serious offence for a host to caught faking (or falsifying) a data possession verification hash.
* The punishment of a host for violating the standard should be a zero-sum game. The agent (user or node) who first reports it will receive reward equal to the transgressors punishment.
* The separation of storage managers and auditors for archives could be applied to all items.
  + All revisions are immutable. So every revision event item could automatically be archived. Since the revision items store all the data in the whole network this means all data would be archived.
  + For efficiency reasons, item controller groups should store the full latest version of their item, but their main responsibility should be to process and verify modification requests. They could delegate much of the data serving task to archive groups which would just serve static content.
* Stress testing protocols
  + There should be a provision for high reputation users to secretly organize a security/stress test on the network. They would sign a document in advance explaining their intentions and then present it to hosts that they ask to participate in a simulated attack.
  + An example stress test could be to ask the hosts that will be involved in a planned transaction to misbehave in some way, such as by allowing a double spend to occur. The test would be how long it would take hosts outside the conspiracy to detect and report the inconsistency.
  + The hosts that participated in the simulated attack would be automatically punished upon detection. A provision in the standard specifically for stress testing though would allow them to present the document signed by the initiators of the stress test and divert the default proceedings to a special resolution process specificity for this scenario. The resolution process should involve a very public debate and vote by the community to decide if the actions performed for the stress test were acceptable.
  + If the initiators use this stress testing provision to convince hosts to let them generate millions of dollars in profit for themselves than they, as well as the hosts that agreed to participate should be punished.
  + If they cause a double spend of $1 however, then this would clearly be in the spirit of an honesr stress test.
  + Regardless of whether the stress test is judge honest or malicious, its effects should be reversed as would be normal practice when inconsistencies are discovered.

# Fri 16-Mar-12

* Censorship Protection
  + All immutable data should be stored in blocks of uniform size, B.
  + When a small piece of data is uploaded, it should be combined with other small pieces from a queue to make up the size to B, with any left over space filled with random data.
  + When a block has been prepared, it should be XORed with a randomly selected existing block and the result should be what is actually stored.
  + The queue of filler data should be constructed by randomly selecting existing data fragments and creating new mirrors of them. These mirrors should be of the original file, not one of the XORed versions stored in other blocks. This original version needs to be reconstructed in the normal way (with appropriate XOR operations) because it will not be stored anywhere.
  + It would be best if the block size is small so that retrieval of small files does not incur excessive overhead.
  + If the block size is small though, then there will be lots of files larger than one block that will not be mirrored automatically by this policy.
  + Perhaps every block should devote half its space to mirroring existing blocks.
  + This would be awfully wasteful of space. But then, all data will be stored with massive redundancy. Why not add a little more in cryptographically obfuscated form to confound censorship?

# Fri 30-Mar-12

* Data representation
  + Field Sets should be conceptual items with a ZID based on the ZID of their parent
  + A property set ZID might look like: 06X3.07BB2 for the property set of property type 07BB2 for item 06X3.
  + An association field set ZID might look like: 06X3.04A33:01K91 where 04A33 is the role and 01K91 is the association type.
  + A decision market can be associated with a field set to determine its permission level just by referencing its ZID
  + The internal representation of a field set item need not be specified. All its features may be catered for purely via indexes because a field set will not have properties.
  + I also want to be able to specify permission level at the granularity of properties of a type and scope associated with a parent item. To implement this using ZID variants would be very messy e.g. “06X3.04A33:01K91@(088G,01FX)”
  + I think the representation of field sets will need to be virtual, or at least on-demand
  + I think I was already on the right track with the idea of using associations to specify field sets by their parent and type.
  + A field set association can be created on demand when a particular field set needs a specialized permission level
* For field set permissions to work with the decision market permissions model, field sets need to have a notion of revisions

# Sat 31-Mar-12

* Field set revisions
  + There needs to be an item associated with a field set so that its controller group can maintain the hash chain of events that modify the field set.
  + A field set item should be created when the first event that affects it is created.
  + The field set should be like a document containing a set of fields (associations or properties).
  + Field set items should be modified implicitly by an event that adds a new property to an item or adds an item to an association
  + There should be scope-specific field sets
    - But should there be a field set item for every possible combination of scopes?
    - For example, with the name property, should there be a property-set for name in the English scope, the Science scope and the (English, Science) scope? Or should compound scopes be excluded?
    - There may be need to specify permissions on any one of the possible combinations of scopes so it seems like it will be necessary to have a field set for every combination of scopes, but that could become unwieldy for fields with complex scopes because the number of field sets required would be combinatorial in the number of scope items in the field’s scope.
    - Perhaps scope-specific field sets could be virtual and only be created if they actually have a lock placed on them
* Decision markets should be associated with event items
  + Actually no; the locus of permissions should be field sets, not events. If you put a lock on an event that created an association would this mean that it locks only modification of that association or of all association field sets it affects?
* When a user locks an event with an investment of Zuth in its decision market, he may choose whether the investment should be sold automatically as soon as it is overridden, or remain invested for some period of time (perhaps infinite) after it has been overridden by another event.
* Field Set permissions
  + A Decision Market can be associated with a field set (possibly scope-specific).
  + The field set is virtual and is defined by the parent, the field type and the targeted scope
  + Once created, a decision market on a field set lasts forever. It may remain dormant/empty for an arbitrary period of time after it is first used, but it will be reused the next time someone wants to place a lock on that field set
  + The candidates of the Decision Market are events. Each candidate item has a part-whole association with the parent Decision Market and a link to the event item it represents.
  + For an Event to be accepted, it will need to win all of the Decision Markets it is involved in. So if an Event will create an association that will add fields to two association field sets, the Event needs to be accepted in the Decision Markets of both association field sets before it will be executed.
  + I think this will create the possibility for a kind of deadlock: if Event X is winning the decision market for field set A and Event Y is winning the decision market for field set B and X is just behind Y in B and Y is just behind X in A, then what should happen? Both X and Y might be beating the current lock (the most recently executed Event) for both A and B, but neither is the clear winner in both decision markets so it is ambiguous what should happen.
* An Event should embody the diff information to transform the current state to the desired state but should not depend on a specific starting state to be valid.
  + This will allow independent Events that act on the same data to be executed out of order (perhaps because they took different amounts of time to gain sufficient permission weight to be executed)
  + For example, two events that both create associations that will add an association field to association field set X should simply contain the instruction to create their respective association. The two events may be entered into the hash chain of X in either order because they don’t depend on any state that is changed by the other.
  + If an Event E0 is held back from being accepted and other Events are accepted before it, E0 can still be executed once it is accepted provided it does not depend on any state that was changed by the intermediate events.
* Permission system Lock paradigm
  + Field Sets can have a lock placed on them. The lock will be a decision market itself with a single binary decision to make: higher or lower lock strength.
  + Fields can also have a lock placed on them in the same way.
  + Event items that are obstructed by a lock need to have a decision market started on them which is also a binary decision: increase or decrease lock override weighting.
  + An event that will modify a field set will pass as soon as it receives a lock override weighting at least as high as the lock on the field set
  + An event that will modify a field must be given a weighting at least as high as the lock on the field in order to succeed.
  + After an Event succeeds, all the Zuth invested in its decision market is refunded
  + If a user has enough zuth in the relevant domain to cause an event to succeed, no decision market need be created – the event will immediately succeed.
  + A lock on a field locks both modification and deletion of the field
  + A lock/weight can also be applied to a specific value of a field. This kind of lock does not carry over to the new value of a field once it has been overridden. Instead, it remains an opposing force to any other value the field might take.
  + If a field has weights on specific values then to change the value of that field requires that the weight on the new value exceed the greatest of the weights on other values plus the lock size on the field as a whole.
  + This doesn’t reconcile the difference between the event and a field’s specific value, which may have been set by an event. This is because an event may affect more than one field.
  + I think it would be most elegant if decision markets on events always behave the same way: once they reach the required threshold for the event to succeed, they are dissolved and are never needed again.
  + A value-specific lock should be an association linking the field and the event that changed the value of the field to the desired value.
  + In order for an event to succeed its weight must exceed the biggest of the locks obstructing it and its weight in any value contests it is involved in must be the winner
  + If I assume that an event might involve other actions than the one modifying a locked value, then it is hard to define what should happen if a field value that was once overridden subsequently becomes the wining value again. Should the whole event it is associated with be re-executed?
  + I don’t think I actually need anything more than the simple lock on a whole field. If there is a dispute, nothing is really helped by allowing an ongoing competition between candidate values to swing back and forth. The only thing that can really be done in the case of a high intensity dispute (where a lot of zuth is put behind both sides) is to reduce the volatility of the field in dispute by increasing the weight on its lock. If the amount of influence behind both sides of the dispute really is evenly matched, then the only way the issue can ever be resolved is through debate or by the intervention of more powerful users who can decide on a value for the field and then increase the lock beyond the reach of the opposing force.
* An item’s controller group should be responsible for managing the hash chains of all its field sets.
* There should be a hash chain for every field set, but not for scope-specific field sets
  + The constraints on scope-specific field sets should be incorporated into the validation on actions affecting the whole field set, which does have a hash chain

# Fri 6-Apr-12

* I think it may be necessary to provide an anonymous cash service through Zutha in order to protect Zutha’s funders from government pressure
  + I don't think it would be possible to do this without also providing a money laundering service. This will of course mean that providing such a service will bring even more government animosity down on Zutha.
  + It should be possible to turn an anonymous cash service on and off (at least the official one supported by the whole Zutha network). This would make it possible to make it available only if it turns out to be necessary in order to protect crucial (for Zutha) financial entities from pressure.
  + It is already possible to be an anonymous payer with the current financial model. At least, it is possible to be as anonymous as you can be with Bitcoin. That means you can either earn all your money through the Zutha/Bitcoin network, making you completely anonymous, or buy Zutha/Bitcoin currency with other currency which makes your identity vulnerable to the involved financial institutions. After you have an anonymous identity with currency, you can use it without providing any information except that identity. With Bitcoin, every public key is an identity and the norm is to collect many of them. The same behaviour is supported on the Zuthanet via the creation of many anonymous users.
  + With both Bitcoin and Zutha, all transactions are visible and thus money can be traced end-to-end through the network, where endpoints are any transactions which can be connected to a corresponding trade in the outside world such as a transfer of money from a bank to a Bitcoin exchange.
    - A Bitcoin exchange could conceivably conceal which Bitcoin public key it transfers money to after it receives a payment from a bank, but in order to permitted to receive money from banks it may be forced to submit to government-imposed requirements that it maintain such records and provide them to the government on demand.
    - Zutha’s transactions with outside banks are inherently public because there needs to be a way to verify that a money handler has received payment so that Zutha Dollars can be transferred from their account. The network is responsible for this verification and the network activity is public. The Zuthanet thus does not currently permit the anonymous exchange of external currency for Zutha currency.
  + A money launderer could try to use the Zuthanet to conceal the identity of transactors
    - If they could get hold of Zutha currency anonymously, perhaps by trading real world cash for Zutha Dollars with a wealthy Zutha user, then they could make a whole lot of complicated transactions to get the money to a 3rd party.
    - The most obscure kind of transaction would be investments, but this can be made a very dangerous process for them as I have discussed earlier.
    - Normal trades of Zutha currency between users are completely transparent though so it would always be possible to keep track of which users it goes to.
    - If a government traced one side of an illegal transaction (purchase of weapons, say), then they could easily trace back the money to all users involved until they found a user they could prosecute. If the money starts and ends in the real world, then there must be a user at both ends of the money flow that can be traced to the real world.
    - In the case of the wealthy Zutha user who exchanges Zutha Dollars for physical cash, that is a kind of transaction that is already possible in the real world. If a government can trace it, then Zutha doesn’t provide any additional anonymity because they can follow the path of the money through the Zuthanet to its destination. If they can’t trace this transaction then Zutha still hasn’t provided any additional anonymity service because the money launderers would have evaded government policing anyway; they could have asked the receiver of their cash to pay their payees directly.
    - Actually, if the wealthy Zutha user has managed to earn his wealth anonymously, then he can provide a service to the money launderers that wouldn’t otherswise be possible. If this user is untraceable then there would be no one to target at this end of the deal. It may also be that the other side of the transaction is a whole lot of legitimate purchases, none of which raise suspicion. By buying Zutha Dollars with cash though, the money launderers are able to convert dirty money into clean money. The wealthy Zutha user who bought their dirty money has to be able to use it though, so that would be how a government could thwart this scheme – by following the dirty money until they found the cleaner, who they could then follow into the Zuthanet.
  + The real problem for governments will come when criminals can make trades of illegal goods and services directly with the consumers of those services for Zutha (or Bitcoin) currency. Every transaction can be made with independent anonymous users. This collection of anonymous users can then make many other innocent purchases to finance the illegal operation.
    - It will still be possible to find patterns in this though. The criminal organisation will have to make some large purchases which give away the connection between large numbers of anonymous users and this could be used to identify patterns in the consumers who financed those anonymous users.
    - All those anonymous users would be like a whole lot of dirty cash, only far less anonymous.
    - A criminal organisation would still be far better off using cash.
  + It seems that Zutha’s financial system might provide very minor services to money launderers, mostly in the form of a trade-off between convenience and anonymity. It would probably provide more potential for anonymity than government policed financial institutions though, and since criminal organizations regularly use those for the convenience, it seems clear that the Zuthanet would be used by criminals if it existed.
  + Regardless of whether the Zuthanet will already provide a haven for money launderers, an anonymous cash system would be their holy grail.
    - In a world where drug dealers could trade drugs with gadget savvy consumers who pay with anonymous cash via a direct over-the-air transfer of cryptographic numbers, the concept of money laundering will no longer be useful for law-enforcement because it will be completely untraceable.
    - The drug buyer could have obtained the anonymous cash legitimately by buying it from the Zutha network using honestly earned money. Even if he hadn’t, there would be no way for a government to know that he had provided money to the drug dealer.
    - Similarly, the drug dealer would obtain electronic cash that he can spend at any honest institution without them even needing to be concerned that they might be accepting dirty money because no one would no they had received such money.
  + I will need to investigate anonymous eCash further before I can be sure of this, but I think it will be possible for the Zutha network to coordinate the creation of eCash and its redemption for standard Zutha Dollars in the open.
    - A user should be able to request x anonymous dollars from the network; x Zutha Dollars will be removed from their account and they will obtain the cryptographic information representing x anonymous dollars which they can transfer to anyone else directly without any intermediation.
  + The main use case for anonymous transactions (from the Zutha network’s perspective) would be to enable hosts and other service providers to be paid while remaining anonymous.
    - The anonymity of hosts might be essential to protect them from persecution by governments and corporations who want to censor content that is provided on the Zuthanet.
    - With anonymous cash it would be possible for hosts to be paid by sending them private messages containing cryptographic coins which they could then redeem untraceably with an account that is unconnected to their host account.
      * Actually this wouldn’t work because the payment must happen automatically and be performed in public by the network so there is no way to keep secret who is being paid
      * Even private payments to an anonymous user do not protect the payee because the payer will know which user ultimately spends their coin (the coin will be posted publicly to prevent double spending)
      * It isn’t even the payment of the hosts that needs to be kept secret though. A host’s user is already known to be a host – it is known that they are being paid for it. The anonymity needs to come when the host wants to convert that Zutha currency into the real world.
      * A host can buy anonymous cash, transfer it to another user they own that is unconnected with their host identity and use that user to make transactions that might expose their real identity. The real person/people controlling the host would thus be connected with the Zuthanet, but there would be no way to trace their ownership of the host, only to the account they use to make purchases using Zutha currency.
* Anonymous electronic cash system
  + Use a group blind signature scheme where the group manager (which has the power to reveal signer) is itself a group with a threshold signature. I'm not sure if this is possible yet.
  + Users sign their requests for cash with a group signature that can only be traced to them with the participation of a minimum number of trusted Zutha hosts. These will only reveal a user for a double spending attempt or criminal activity.
  + A merchant can receive cash from a customer and verify its validity with the issuing bank before giving up goods to the customer. If the customer has tried to double spend, the bank will know and prove this to the distributed customer group manager, which will reveal the customer's identity so that he can be punished.
  + It may be that the bank will only be needed to sign (blindly) coins upon creation. When a coin is spent this action can be verified publicly and the coin publicly marked as spent. There wouldn't be any need to contact the bank to know whether a coin is valid.
  + If it is possible to construct blind threshold signatures, the bank could be a randomly selected collection of hosts - more than the threshold.
  + With public withdrawals and deposits there may not be any need for a customer group manager. Double spending would be prevented by having all transactions be online. A transaction would be complete only after the merchant has deposited the cash in his account and the cash has been publicly marked as used/depleted.
  + I don't think there's any need for a blind threshold signature. Upon cash withdrawal, a random selection of hosts will be chosen to sign the withdrawer's random 'coin'. This exact number of signatures must be demonstrated by a depositor for a coin to be accepted. This would mean a sufficient number of hosts could collude to make false signatures.
  + Zutha could just provide the service of enforcing the contracts made by money handlers when they sign cash. This way, a host is liable for the agreed upon worth of the cash it signs and will only rationally sign a coin in exchange for something of equivalent worth. The issue that remains is that the host will go bankrupt (or at least his Zutha account will be depleted and he refuses to replenish it). This system is somewhat better than a lone business or individual promising to back its issued cash, but it doesn't benefit from the same redundancy as the main Zutha financial system. Users can spread their risk across many hosts still though.
  + The risk could actually still be taken off the shoulders of cash purchasers though if money handlers' accounts are allowed to go into the negative. If this happens it effectively means the rest of the money handlers are bearing the load of the bankrupt money handler's unpaid debts. Because of this structure, perhaps money handlers should have more influence than other users in assessing the credit-worthiness of their peers.
  + It may still be desirable to use the group manager scheme so that money launderers can be exposed with the cooperation of enough trusted hosts
  + The payer of an anonymous coin would be able to detect which user spends it. If there turns out to be a need to be able to keep the identity of a payee secret from the payer then I think another anonymity scheme will be required.
  + I think basically all anonymity requirements can be provided by the ability of an agent to transfer money anonymously between users he owns.
    - If agent A wants to transfer money to agent B whom he knows through a channel outside of the Zuthanet (such as by meeting him face to face in the real world), and both parties want to retain complete anonymity on the Zuthanet, then agent A should create a dummy user account Ua1 and transfer cash anonymously from his main account Ua0 to Ua1. Agent A should ask agent B to provide a dummy account Ub1 to which agent A can then directly transfer funds using Ua1 (no need for anonymous cash here since these are both throwaway accounts. Agent B can then transfer the money from Ub1 to his main account Ub0 using anonymous cash.

# Mon 9-Apr-12

* Associations should only be able to have properties with cardinality exactly 1
  + This is so that Association Properties will not need unique id numbers.
  + If an Association could have multiple properties of a particular kind then each one would need a unique id number so that it could be referenced for modification or deletion

# Tue 10-Apr-12

* Human-friendly query languages should be implemented on clients
  + Hosts should only be expected to understand an unambiguous json encoded query language
  + Client writers can experiment with sophisticated inference algorithms to make querying easier for users
  + For example, a client query language might provide access to calculated properties like ‘age’ which are translated transparently into queries on a date-of-birth property before being submitted to hosts.
  + Client query languages could also allow ambiguity (in natural language processing for example) because the client will translate the query into an unambiguous form before it is submitted redundantly to many hosts. Thus the responses from the hosts will still be deterministic and can be judged correct or incorrect robustly.
  + Clients with sufficient resources (or insufficient reputation to ask hosts to process their queries for them) will do all query processing locally and will thus have no need for a sophisticated standard query language at all; they will simply ask for whole field sets from the network and run set operations on them locally.
* Sometimes the goal of punishing other users will be a stronger motivation to invest negatively than the prospect of profiting personally from the investment
  + For example, if a bunch of jokesters are pooling their reputation to modify a highly visible item for the sake of a prank, then those users with the will to oppose them are likely to be motivated by righteousness or disgust and a desire to punish the hooligans rather than a desire to make profit by anticipating similar emotions on the part of other users.
  + It could be helpful therefore if investing negatively had some kind of direct undesirable effect on the positive investors in the same item.
  + Actually it may be better if the ability to punish the opposite investors is an optional feature rather than an always-on feature because there will be many occasions when an investment does not represent a criticism of the behaviour of the opposite investors but merely a disagreement with them.
  + All that is needed is a special item whose ownership is defined proportionally by investment (either positive or negative) in a target item.
    - If you disapprove of those who support a candidate event E (such as one to vandalize or play a prank on a high visibility item), then you can create an item A representing this action (investing in E) and then invest negatively in it. All those who invest negatively in A will reduce the reputation of all those who have an investment in E.
    - The ability to create such an item as A could be supplied by more basic building blocks. For example, ownership of an item could be allowed to be defined in terms of a function that returns a set of users and a corresponding weight. The ownership function of A would return the set of all investors in E with the weight of each owner being the size of their investment in E.
* Zuth invested negatively should continue to contribute to the investor’s reputation while also reducing the reputation of the owner of the item. This will create an incentive for users to invest negatively in users or communities that hoard their reputation because it will not disadvantage the investor but will decrease the total reputation in the economy and will thus slightly increase the proportion of reputation owned by the investor.
* Modification Locking
  + Every field and field set can have zero or one associated Field Lock item
  + A Field Lock item is an ordinary binary decision market where positive investment increases the size of the lock and negative investment decreases the size of the lock
  + When a user wants to perform an action that is impeded by one or more lock:
    - If they have more zuth in their account than the size of the biggest lock impeding the action, then they can perform the action immediately.
    - If they do not have sufficient zuth to override the biggest lock, then they can create a candidate event item (which is a decision market) and invest as much as they desire into it. When the size of the candidate event item’s decision market exceeds the size of the biggest lock impeding it, then the event is performed (providing that no conflicting events have passed in the meantime) and the decision market is dissolved (all Zuth is returned to investors).
  + A candidate event should simply be an event with a property marking it as not yet committed.
  + The owners of an event should be, in proportion to investment size, the investors in the event at the time it passes. This share of ownership should be frozen for all time.
  + Instead of dissolving the decision market when an event passes, it should simply become an investment market which rewards or punishes the owners of the event.
  + At the instant that an event is committed, the reputation of its owners should not change because their investments will go from decision market investments which continue to contribute to their own reputation, to investment market investments which contribute reputation to the owners of the item of which their share of ownership is identical to their share of the investment in it.
    - Actually any negative investment in the event item will reduce the reputation of the event owners from the moment when it is committed
* There may still be need for host-managed calculated properties
  + One example would be the current worth of an investment or decision market
  + There needs to be a controller group with responsibility for maintaining and indexing important calculated properties
  + I don’t think there is any justification for calculated names though. Names which are simply a compilation of a few pieces of easily accessible data should be left to clients to calculate in the format they desire.

# Fri 13-Apr-12

* Eliminate the distinction between clients and hosts
  + If you want to be able to receive automatic updates for field sets you subscribe to you have to be a registered host and respond to requests from the network yourself
  + The default way to perform queries should be to receive the latest data from the relevant field sets and calculate the query yourself
  + Anyone should be able to ask an automatically selected host group to process a query for them
  + The efficiency of the ad-hoc network design depends on the number of hosts being relatively small compared to the total number of nodes. I think this will still be an essential design feature so maybe requiring every node to be a host won’t work after all.
* The network would be more efficient if queries involving particular items were consistently sent to the same group of hosts
  + There also needs to be a way to reward hosts for responding correctly to lots of requests without opening up the reward system to gaming
  + Requests that involve just one field set should always be directed at the controller group responsible for that field set. But responses to such requests need to be rewarded so that there is an incentive for hosts to service them. But the reward system needs to be structured so that hosts will only benefit sustainably by responding to requests from strangers – as opposed to dummy requests they send to themselves to game the reward system.
  + If it costs more to the requester than is earned by any one host then it would not be profitable to create lots of dummy requests to reward a target host. However, this would require that requesters have reputation so they have some currency to spend on the request. It is essential that both anonymous users and users without reputation can make requests to the network though and provided that they can, there will be no need for users with reputation to give any of it up to make requests.
  + The limiting factor in a host’s reward should be its computational resources; not the quantity of requests it happens to receive. What is the problem with a host that is being underutilized sending dummy requests to itself to show off what it is capable of?
  + A host could be scored on the total number of requests it responds to, but then it could optimize itself to respond to queries that it knows are coming, thereby obtaining an advantage over other hosts in the network that are behaving as desired and responding to unanticipated requests.
  + It could simply be left up to the Zutha community to determine the reputation of hosts via investment
    - As long as there is a dedicated enough influential subset of the community who police abuse of this system by opposing both unfair boosting and sabotaging of host reputation, this could be viable.
    - It would be up to individual users to judge the contribution of hosts to the network. Their goal would be to predict future changes in a host’s reputation so they can invest in that host (positively or negatively) before other users catch on, thereby making a profit.
    - Users involved in this market could perform various tests on hosts themselves, measuring response times to various requests under various conditions like anonymity, time of day and current network load.
    - Investors would have an incentive to share their explanations for their investment decisions so that other users might follow suit and thereby increase the value of their investment.
  + Zutha should pay a proportion of its income to hosts in proportion to their reputation in the Zutha Host domain.
  + Real money investment could also support hosts in just the same way that it is designed to reward arbitrary contributors of value to society.
  + If an investment market determines host reputation, then there would be no need to randomize the selection of request handlers.
    - Requesters could make their request to whichever hosts they choose. They may very well choose randomly for some kinds of requests, though because that would decrease their chances of being duped by an attacker.
    - Some kinds of requests are likely to be made frequently to the same hosts because those hosts are known to be better equipped to answer them efficiently.
    - I’m of course talking about view requests only here. Modification requests are deterministically handled by the appropriate controller groups.