HIDING IN THE DARK

ANONYMOUS DISTRIBUTED DATA STORAGE

HOWDY.

HOWDY.

HOWDY.

HOWDY.

OUTLINE

- Monetas
- An Overview of Open Transactions
- An overview of Bitmessage and what it can do for the decentralized ecosystem.
- The “Holy Grail post”
- The “Bazaar” marketplace and key concepts.
- What we want to accomplish and what our limitations are.
- Distributed Storage and key concepts.
- Requirements for the design and approaches.
- State of progress and plans for the immediate future.
- Q&A
- References

Monetas

Instant. Global. Secure. Private.

LEGACY FINANCIAL SYSTEMS

- World’s existing financial systems are centralized, creating single points of control and failure.
- Centralized systems are inherently vulnerable to abuses of power and catastrophic failure.
- “Legacy” systems are expensive, slow, complicated, insecure and prone to privacy violations, fraud and corruption.

LEGACY FINANCIAL SYSTEMS

- World’s existing financial systems are centralized, creating single points of control and failure.
- Centralized systems are inherently vulnerable to abuses of power and catastrophic failure.
- “Legacy” systems are expensive, slow, complicated, insecure and prone to privacy violations, fraud and corruption.

HISTORICAL RECAP

- Decentralized (People-based) trust systems start to fail as market systems exceed Dunbar's Number (~150 relationships).
- This led to the evolution of market systems controlled through a top-down model.
- Printing press was the beginning of a return to Decentralized Systems.
- Most recent developments in this direction have been the Internet and Bitcoin (Blockchain based systems).

PILLARS OF A DECENTRALIZED SOCIETY

1. Decentralized Communications
2. Decentralized Law
3. Decentralized Production
4. Decentralized Finance

SMART CONTRACTS

- Computer protocols that facilitate, verify, or enforce the negotiation or performance of a contract, or that obviate the need for a contractual clause.

- Claimed to have begun with the “Agoric Computing” movement in the 1970s to bring market mechanisms (such as Auctions) to computational resource management.

- Coined by Nick Szabo around 1993.

- Anything humans can agree to do together, can be described in a smart contract.

- Can enable Wallets, Decentralized Exchanges, Merchant Services.

SMART CONTRACTS

- Computer protocols that facilitate, verify, or enforce the negotiation or performance of a contract, or that obviate the need for a contractual clause.

- Claimed to have begun with the “Agoric Computing” movement in the 1970s to bring market mechanisms (such as Auctions) to computational resource management.

- Coined by Nick Szabo around 1993.

- Anything humans can agree to do together, can be described in a smart contract.

- Can enable Wallets, Decentralized Exchanges, Merchant Services.

OPEN TRANSACTIONS

OPEN TRANSACTIONS

OPEN TRANSACTIONS

A financial crypto and digital cash software library that operates as a centralized transaction system.

Complementary to Bitcoin in that it provides features that Bitcoin cannot, such as untraceable anonymous transactions, no latency and more.

Featuring:

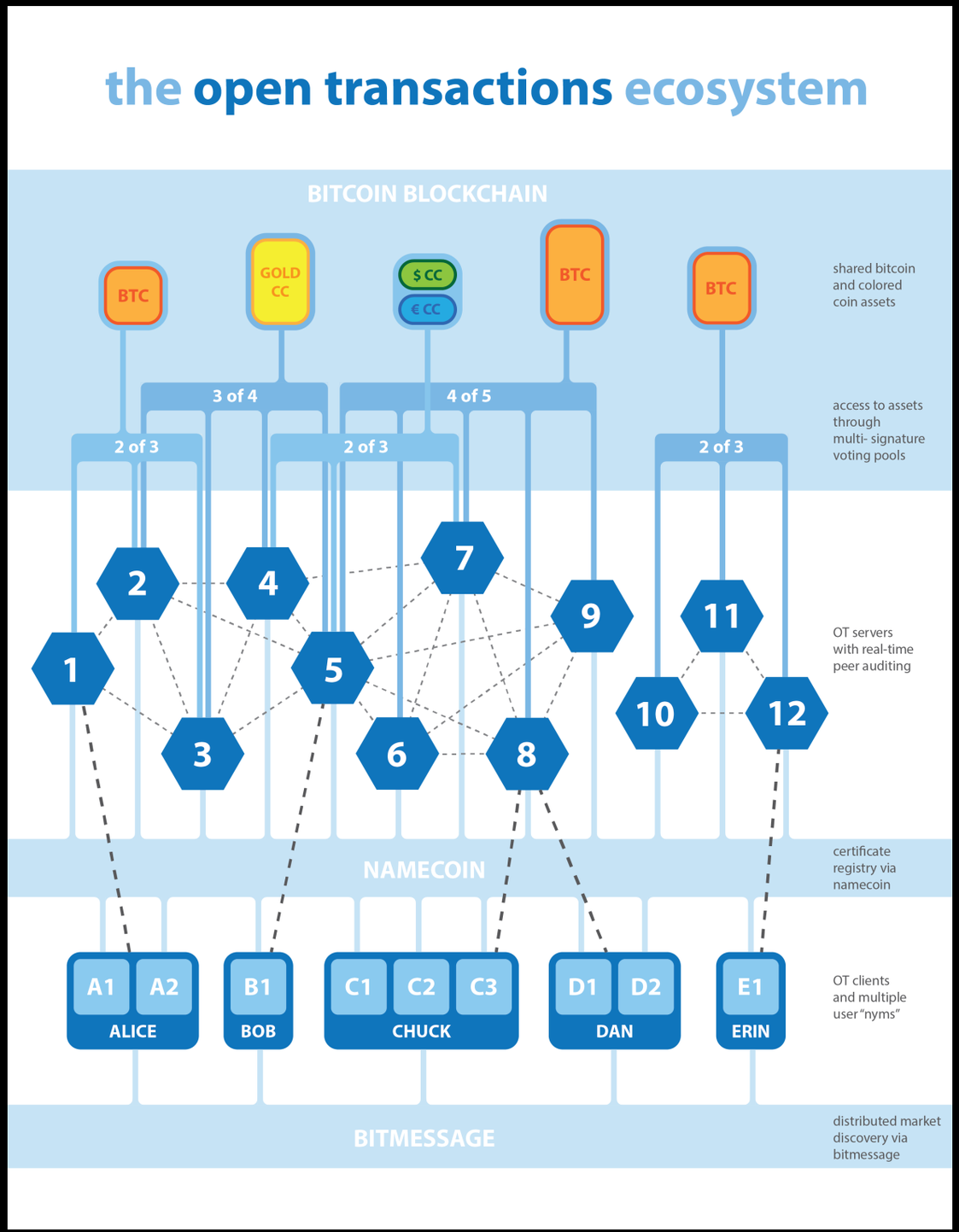
- Untraceable Digital Cash (real blinded tokens)
- Anyone An Issuer (Ricardian-style Contracts)
- Bearer-only, Fully-Anonymous (when used cash-only)
- Pseudonymous User Accounts (user account == PGP key)
- No Account History (asset account == the last receipt)
- Many Financial Instruments (cheques, cash, vouchers, invoices...)
- Basket Currencies (10 "baskets" == 5 gold, 3 silver)
- Markets with Trades (stop, fill-or-kill, limit orders...)
- Payment Plans
- Includes an API, server and client.

Moneychanger

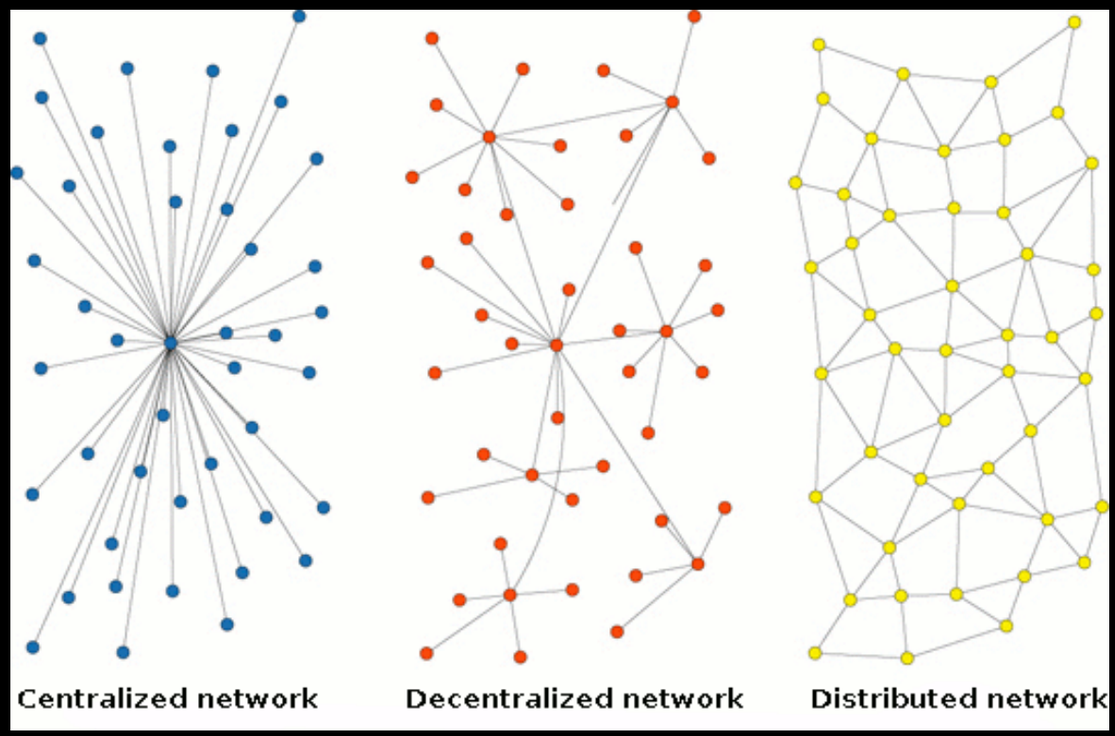
- A reference implementation of a currency accounting application that accesses the Open Transactions API.
- Integrates with Bitcoin, Namecoin and Bitmessage.

OPEN TRANSACTIONS

OPEN TRANSACTIONS



WHAT IS BITMESSAGE?



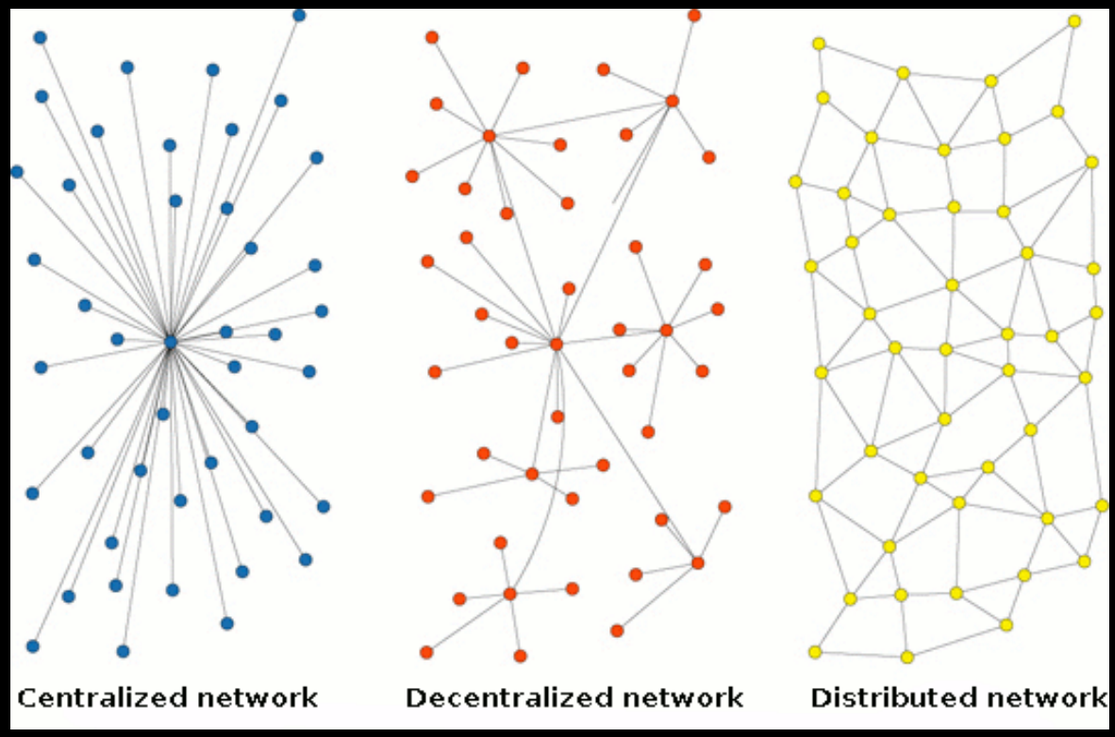
WHAT IS BITMESSAGE?

- Bitmessage is a peer to peer communications protocol used to send encrypted messages to one person or many subscribers.

- It is Distributed and Trustless, meaning that you need-not inherently trust any entities like root certificate authorities.

- It uses strong authentication, which means that the sender of a message cannot be spoofed.

- “non-content” data such as sender and receiver information is hidden from eavesdroppers, such as those running warrantless wiretapping programs.



BITMESSAGE OVERVIEW

- Messages are encrypted and sent across network, where all nodes are guaranteed to receive it and hold it for a set period of days.[1]
- A broadcast message can be opened by all subscribers. Requires owner to be online.
- A channel can be created using the password as the deterministic address key.
- Messages are limited to 256kb[1]
- There are ways of using custom drivers to speed up the proof of work calculations in Bitmessage by 40%.

[1] THIS IS A NEW FEATURE IN BITMESSAGE V3

HIDING IN PLAIN SIGHT

HIDING IN PLAIN SIGHT

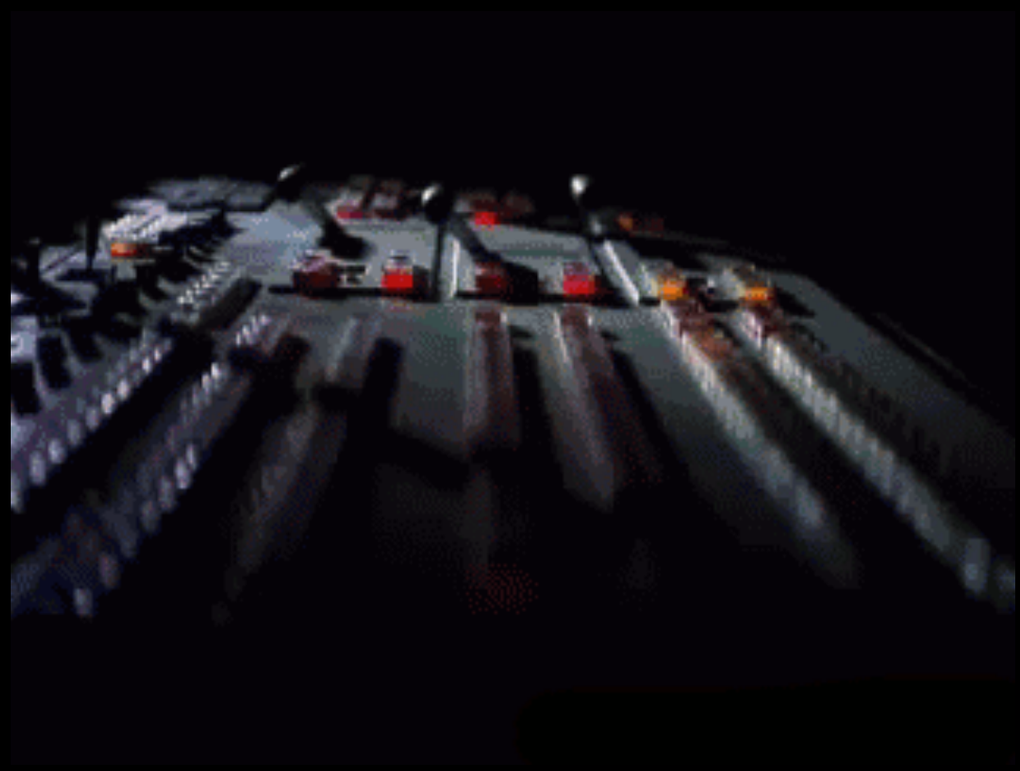
HIDING IN PLAIN SIGHT

HIDING IN PLAIN SIGHT

HIDING IN PLAIN SIGHT

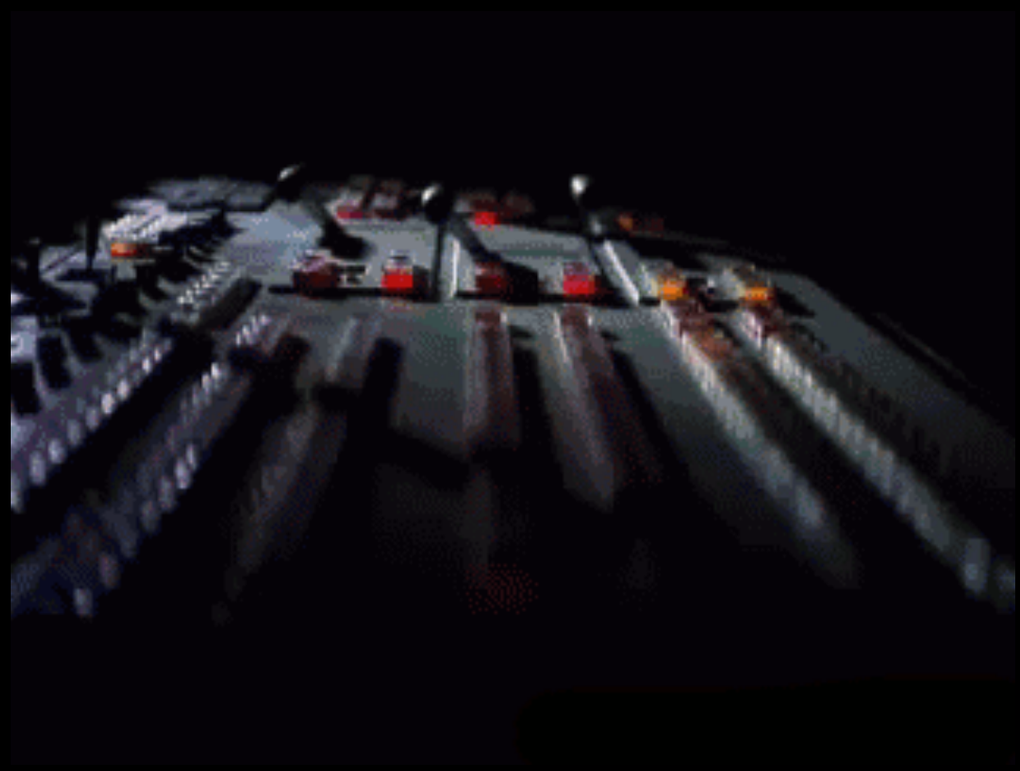
- The ability to discretely broadcast information is part of what makes this attractive for OpenBazaar and our own Bazaar project.
- Binary data doesn’t have to sit on the network permanently. It only needs to stay alive for transfer.
- Nodes can’t arbitrarily open messages.
- Messages can still be encrypted on top of what is already performed by the network.
- Binary data can be transferred by base64 encoding it first.

WHAT CAN IT DO?



- Decentralizing markets from the bottom up.
- “BitCloud” distributed fault-tolerant file storage (plus it’s free!)
- Proof of work security with speed boost.
- Secure order announcements.

WHAT CAN IT DO?



- Decentralizing markets from the bottom up.
- “BitCloud” distributed fault-tolerant file storage (plus it’s free!)
- Proof of work security with speed boost.
- Secure order announcements.

THE HOLY GRAIL

Bitmessage solves discovery across federated OT servers.

Bitmessage makes possible fully-decentralized p2p markets, as well as p2p escrow across OT federated servers, easy p2p and server-to-server wiring of funds and conversion of currencies, both within OT and also between OT and the conventional banking system.

THE HOLY GRAIL

THE HOLY GRAIL

Holy Grail Post - https://bitcointalk.org/index.php?topic=212490.0

Using Bitmessage with OT to effect server-to-server wiring of funds: **http://pastebin.com/NjQgDarx**

- The wiring protocol is all about Alice trying to discover Bob so she can move her money from one server to another (and Bob trying to discover Alice so he can make a profit by moving money from one server to another.)

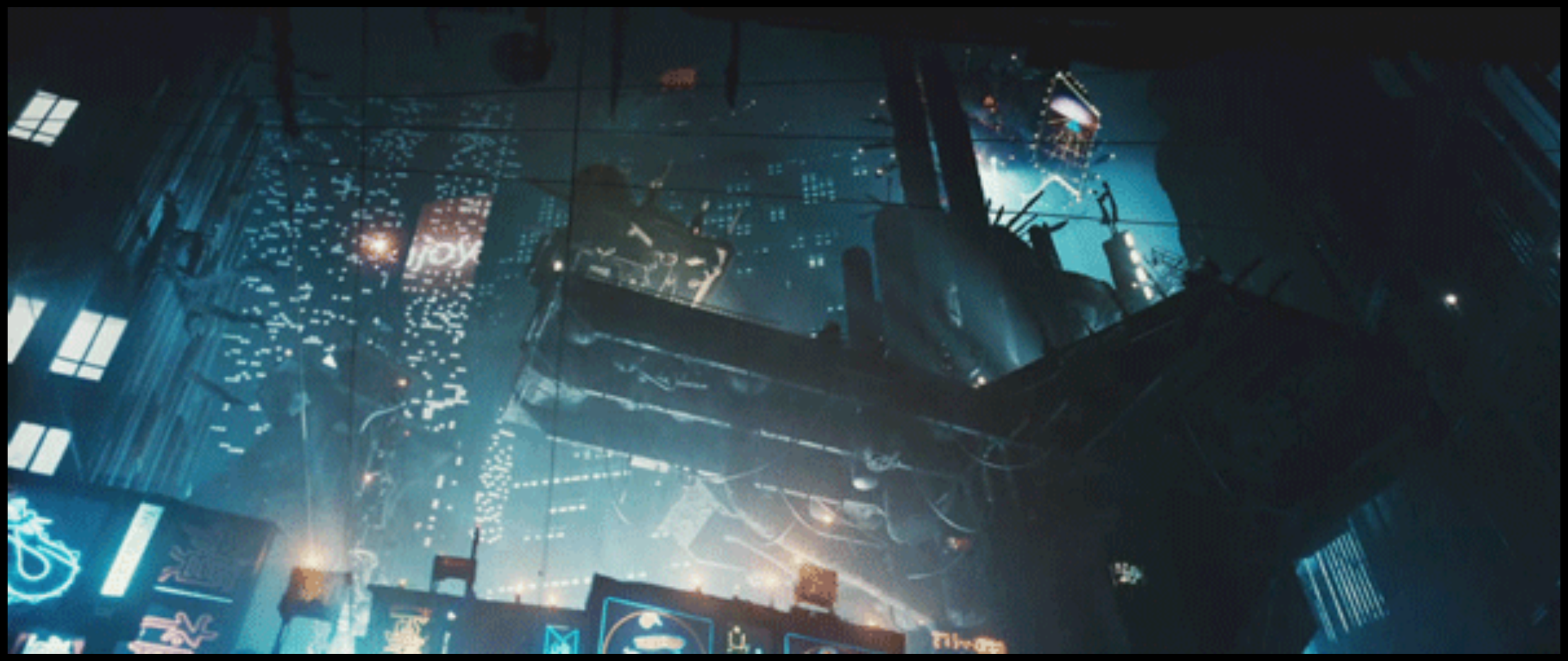
Using Bitmessage with OT to effect escrow-based conversion of currencies across OT federated servers: **http://pastebin.com/S1W5guAQ**

- The currency conversion protocol is about Alice and Bob being able to choose a server they can agree to meet on so they can trade one currency for another inside OT. (For cases where they aren't already trading on the same OT server.)

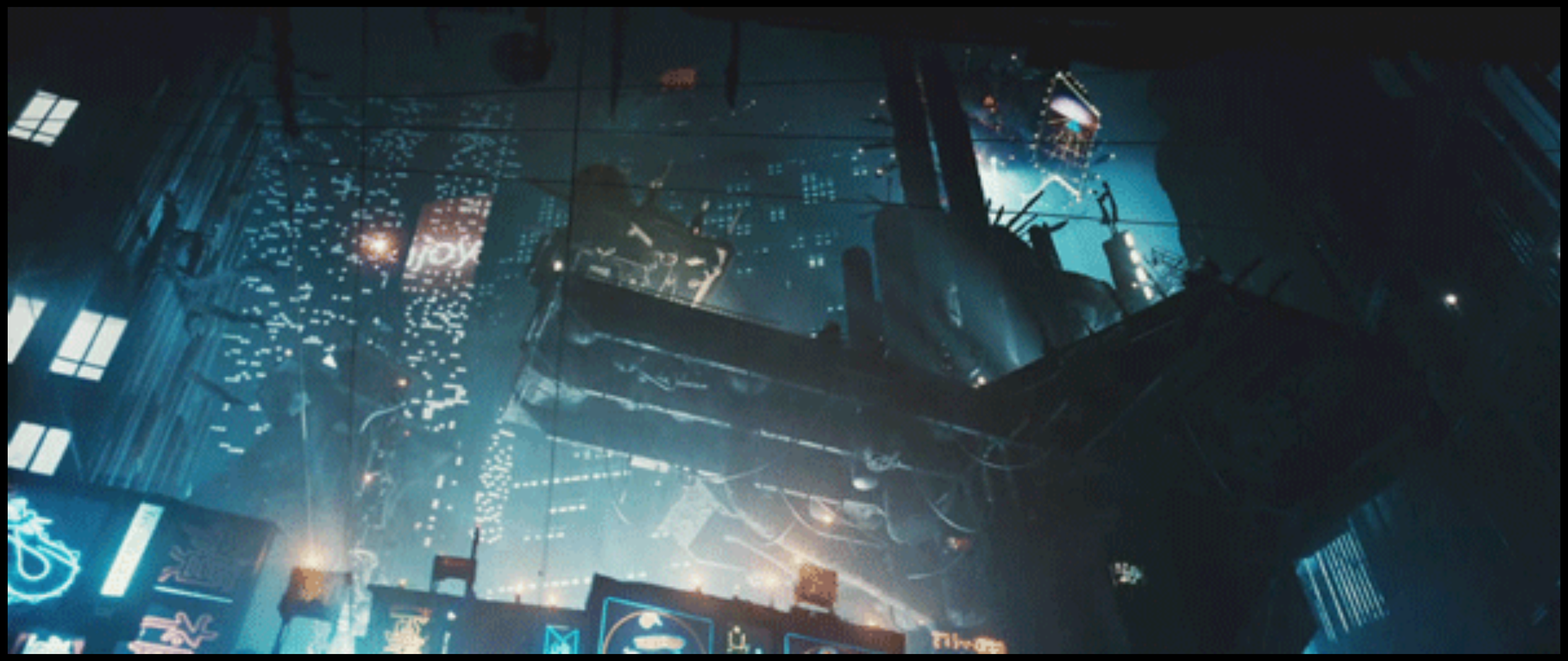
Using Bitmessage with OT and SEPA so that Alice can p2p send any currency which Bob receives as Euros in his Euro account: **http://pastebin.com/SsLrxVP6**

- The SEPA transfer protocol is about Alice being able to send Silver Grams, which Bob receives as Euros in his Euro bank account. It's also about Jorg earning a profit in silver grams, by sending a SEPA transfer to Bob on Alice's behalf.

THE BAZAAR



THE BAZAAR



BAZAAR TERMS

BAZAAR TERMS

Vendor

- Creates advertisements and submits them to marketplace for approval.

BAZAAR TERMS

Vendor

- Creates advertisements and submits them to marketplace for approval.

Marketplace

- Marketplace is who broadcasts the advertisements for the clients to browse.
- Approves / revokes vendors.
- Approves advertisements.
- Tracks and provides ratings info for all parties.
- Does NOT adjudicate disputes.

BAZAAR TERMS

Vendor

- Creates advertisements and submits them to marketplace for approval.

Marketplace

- Marketplace is who broadcasts the advertisements for the clients to browse.
- Approves / revokes vendors.
- Approves advertisements.
- Tracks and provides ratings info for all parties.
- Does NOT adjudicate disputes.

User

- Adds marketplace contracts to his client.
- Listens to Bitmessage channel for each marketplace.
- Answers advertisements by contacting Vendor directly.

BAZAAR CONTRACTS

BAZAAR CONTRACTS

Marketplace Contract [1]

- Contains Bitmessage address for private communications.
- Contains info on requirements for payment and dispute mediation.
- Hashed ID is deterministic seed for Bitmessage channel.
- Optionally can contain a URL, or Freenet or I2P URL where ads are posted.
- Client software should ignore any broadcasts in the marketplace's channel unless they are signed by the marketplace.
- Any user can ask the Marketplace (via his private BM address) to provide ratings data on any vendor. Similarly, any vendor can ask for ratings data on any user.
- Any entity can supply ratings to the marketplace, as long as the rating is from a vendor regarding a user, or a user regarding a vendor, and as long as the rating contains a copy of the vendor's original signed acceptance of an order, and optionally a copy of the payment receipt or dispute mediation receipt.

[1] Signed OT contract.

BAZAAR CONTRACTS

Marketplace Contract [1]

- Contains Bitmessage address for private communications.
- Contains info on requirements for payment and dispute mediation.
- Hashed ID is deterministic seed for Bitmessage channel.
- Optionally can contain a URL, or Freenet or I2P URL where ads are posted.
- Client software should ignore any broadcasts in the marketplace's channel unless they are signed by the marketplace.
- Any user can ask the Marketplace (via his private BM address) to provide ratings data on any vendor. Similarly, any vendor can ask for ratings data on any user.
- Any entity can supply ratings to the marketplace, as long as the rating is from a vendor regarding a user, or a user regarding a vendor, and as long as the rating contains a copy of the vendor's original signed acceptance of an order, and optionally a copy of the payment receipt or dispute mediation receipt.

Vendor Contract

- Contains Bitmessage address for private communications.
- Includes payment info.
- Marketplace and Vendor contracts are similar to OT asset/server contracts, since they contain the public key of the signer, and since the contract ID is the hash of the contract.

[1] Signed OT contract.

BAZAAR CONTRACTS

Marketplace Contract [1]

- Contains Bitmessage address for private communications.
- Contains info on requirements for payment and dispute mediation.
- Hashed ID is deterministic seed for Bitmessage channel.
- Optionally can contain a URL, or Freenet or I2P URL where ads are posted.
- Client software should ignore any broadcasts in the marketplace's channel unless they are signed by the marketplace.
- Any user can ask the Marketplace (via his private BM address) to provide ratings data on any vendor. Similarly, any vendor can ask for ratings data on any user.
- Any entity can supply ratings to the marketplace, as long as the rating is from a vendor regarding a user, or a user regarding a vendor, and as long as the rating contains a copy of the vendor's original signed acceptance of an order, and optionally a copy of the payment receipt or dispute mediation receipt.

Vendor Contract

- Contains Bitmessage address for private communications.
- Includes payment info.
- Marketplace and Vendor contracts are similar to OT asset/server contracts, since they contain the public key of the signer, and since the contract ID is the hash of the contract.

Advertisement Contract.

- Must be signed by both marketplace and vendor.
- (Alternately, Vendor signature is sufficient as long as Marketplace has approved Vendor.)
- Contains timeout field.
- Contents text content.
- Alternately contains a URL or Freenet or I2P URL where ad content is posted.

[1] Signed OT contract.

BAZAAR ORDERS

BAZAAR ORDERS

User order

- Includes copy of advertisement.

BAZAAR ORDERS

User order

- Includes copy of advertisement.

Vendor acceptance of order

- Includes copy of user order, which includes copy of advertisement.
- Includes payment request. (OT escrow, OT invoice, BTC address, BTC multisig, etc.)

BAZAAR ORDERS

User order

- Includes copy of advertisement.

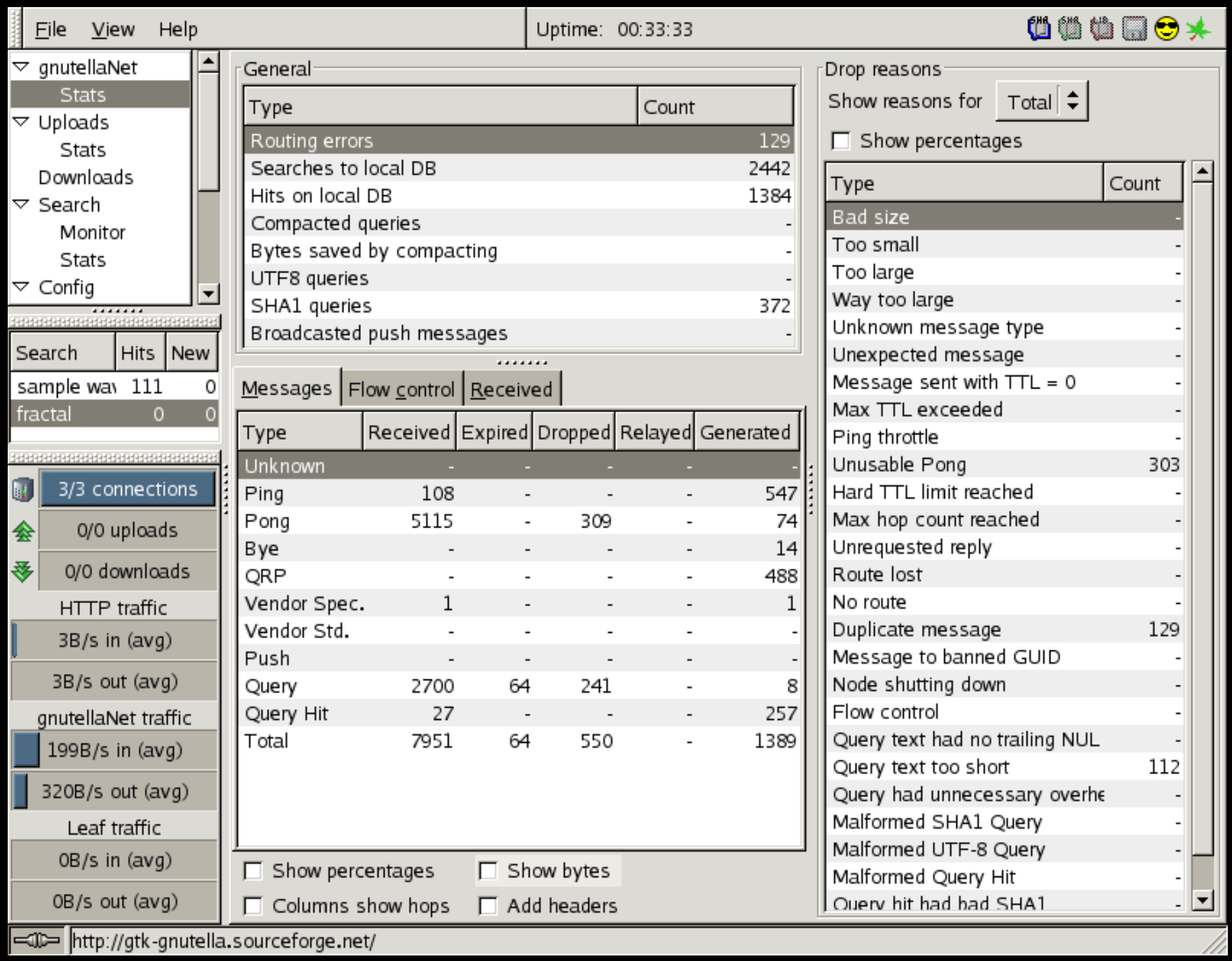
Vendor acceptance of order

- Includes copy of user order, which includes copy of advertisement.
- Includes payment request. (OT escrow, OT invoice, BTC address, BTC multisig, etc.)

Order Process

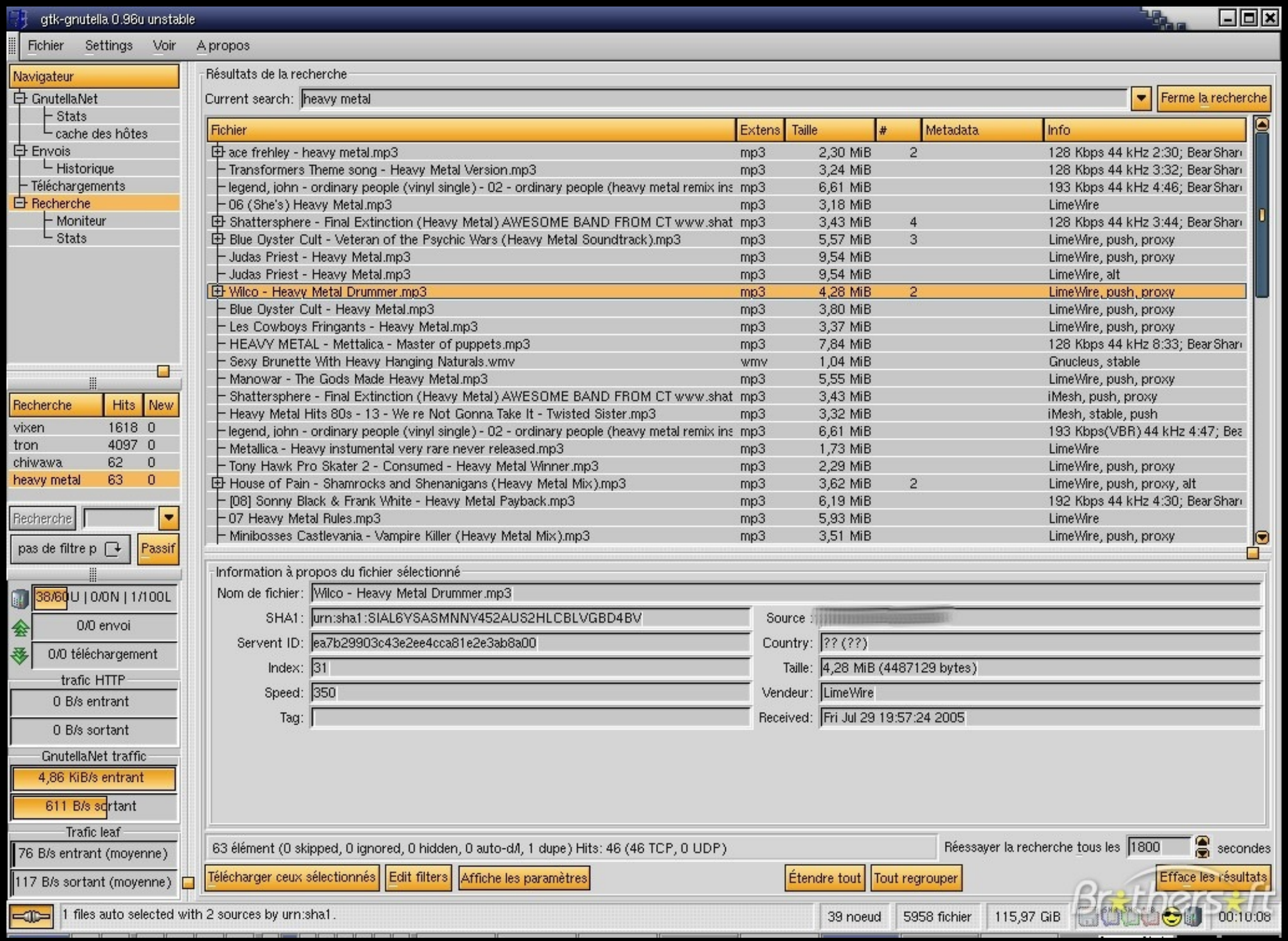
1. The Vendor sends his vendor contract to the marketplace for approval.
2. The vendor sends his advertisements to the marketplace for broadcast.
3. The marketplace signs and broadcasts the advertisements.
4. Users receive and organize the advertisements. They either read them directly from the channel broadcasts, or from a URL located in the marketplace contract.
5. Users answer advertisements to vendor directly. (Placing an order.)
6. Vendors can ignore, or reject, or accept.
7. Marketplace tracks ratings, and “Judge Judy” adjudicates disputes. (She is a third party to an escrow contract.)

UNIFIED INTERFACES



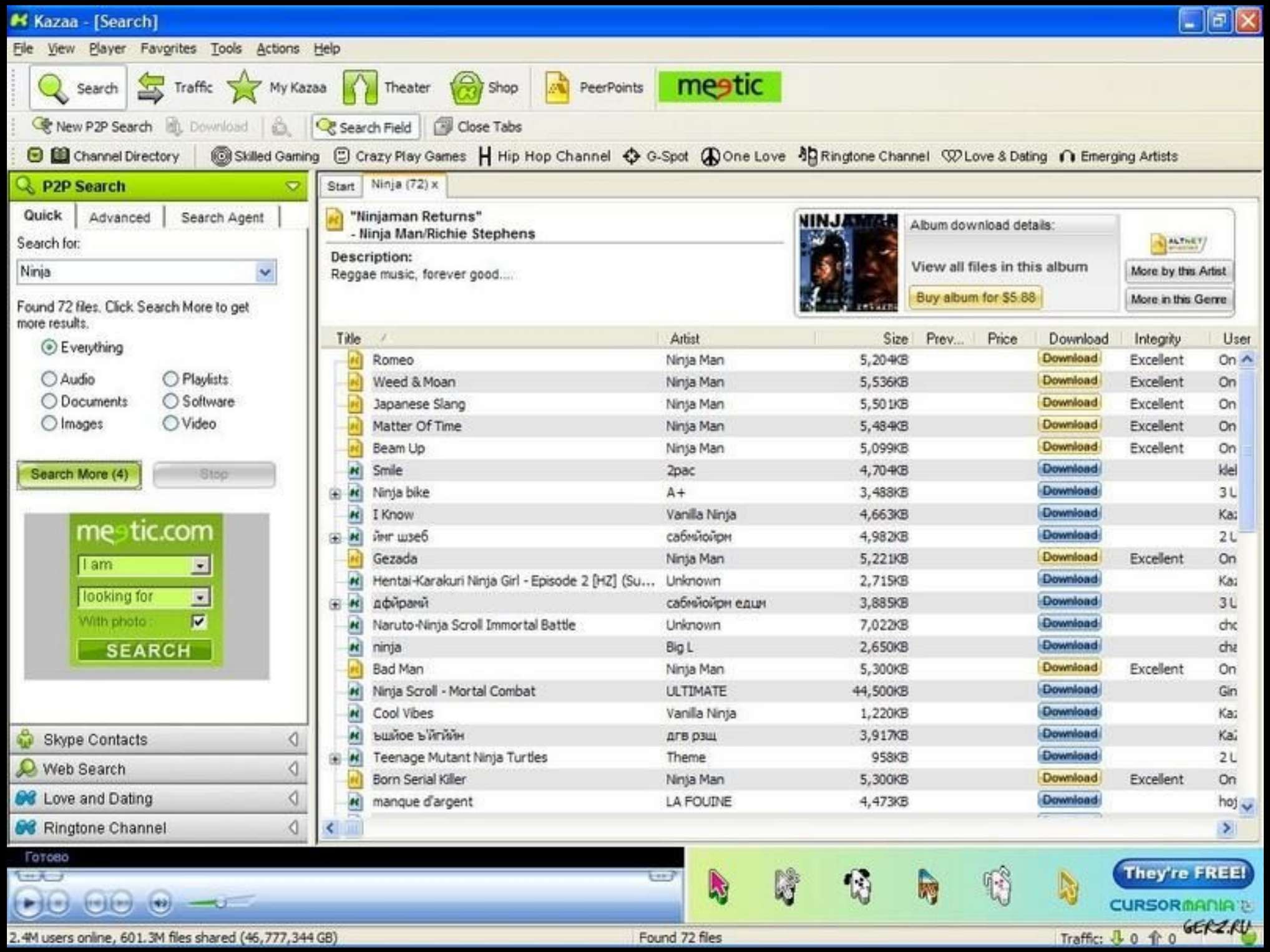
UNIFIED INTERFACES

UNIFIED INTERFACES



UNIFIED INTERFACES

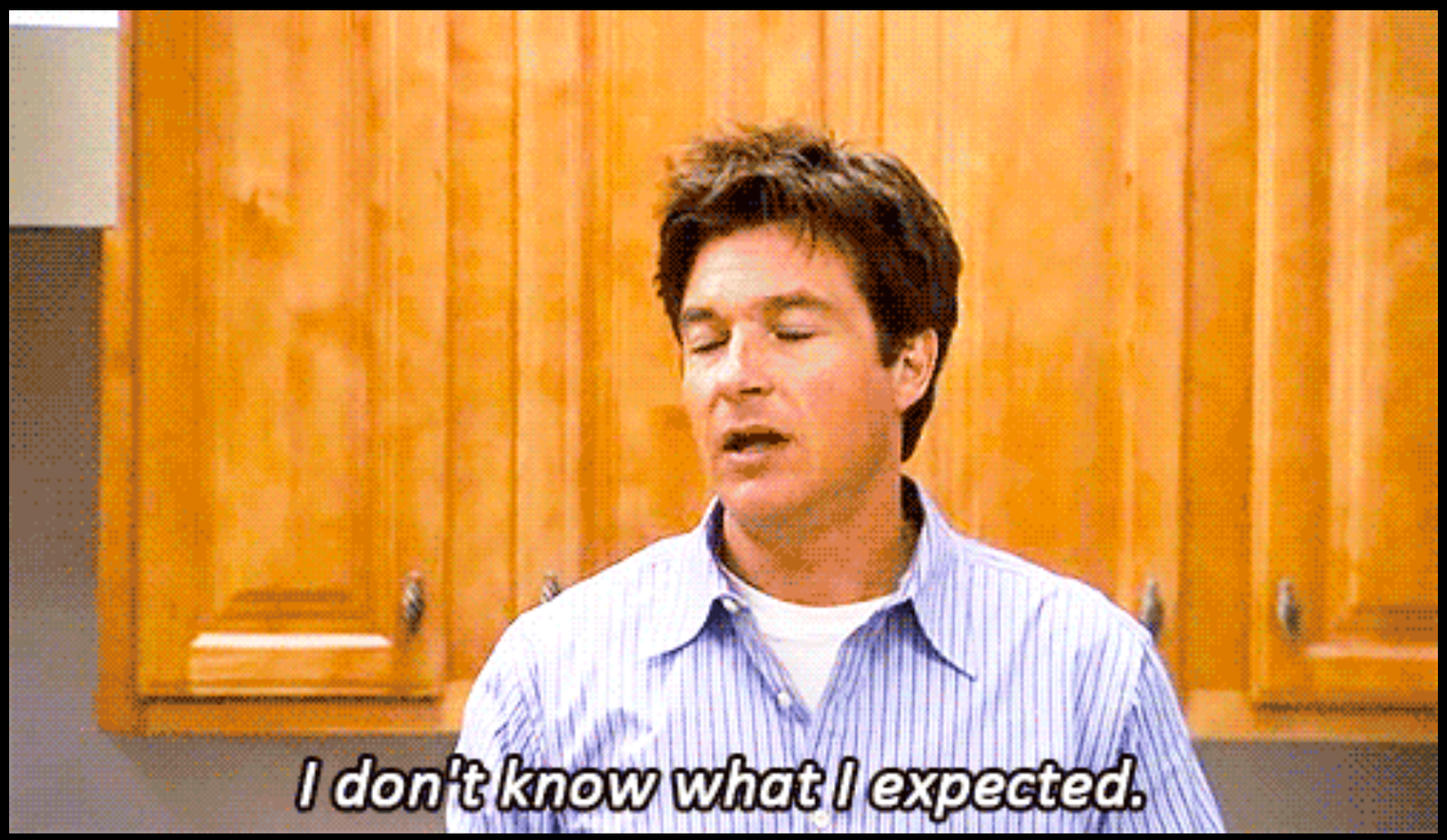
UNIFIED INTERFACES



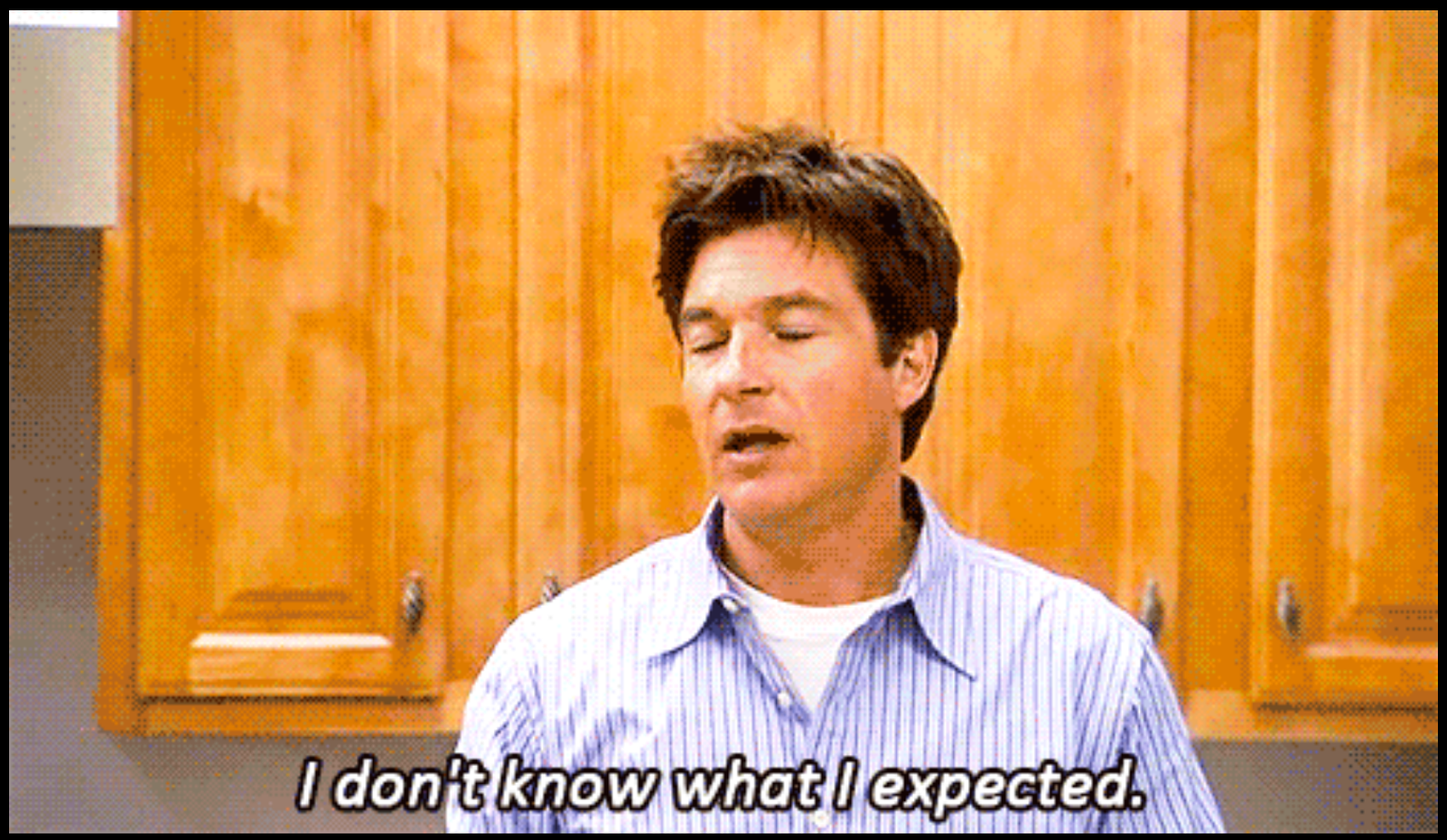
BITMESSAGE UNFRIENDLY TO BINARIES

- The larger the message, the longer it takes to process.
- Large messages can congest the network as a byproduct of it’s features.
- 256kb limit on messages prevents most binary data transfers.
- Makes data retrieval costly in time.

TESTING THE NETWORK



TESTING THE NETWORK



TESTING THE NETWORK

TESTING THE NETWORK

- Composing a message larger than 1mb took about 15 minutes to complete.
- Complaints from users on the network about large messages.
- Bitmessage v3 will limit messages to 256kb, I was likely partially responsible for that change.
- Large messages would freeze the client.

TESTING THE NETWORK

TESTING THE NETWORK

DILEMMA

- We know that we don’t want Advertisements to be extraordinarily large. This will complicate the situation more.

- We don’t want users to have to leave the application they’re in to use another tool to view the details from the contracts. That can destroy anonymity.

- We could use tags in advertisement information for things like images and audio or video and store data with peers, however…

- If we store data in single places that may give information away about the users and create bottlenecks and failure points for data.

DILEMMA

- We know that we don’t want Advertisements to be extraordinarily large. This will complicate the situation more.

- We don’t want users to have to leave the application they’re in to use another tool to view the details from the contracts. That can destroy anonymity.

- We could use tags in advertisement information for things like images and audio or video and store data with peers, however…

- If we store data in single places that may give information away about the users and create bottlenecks and failure points for data.

DILEMMA

- We know that we don’t want Advertisements to be extraordinarily large. This will complicate the situation more.

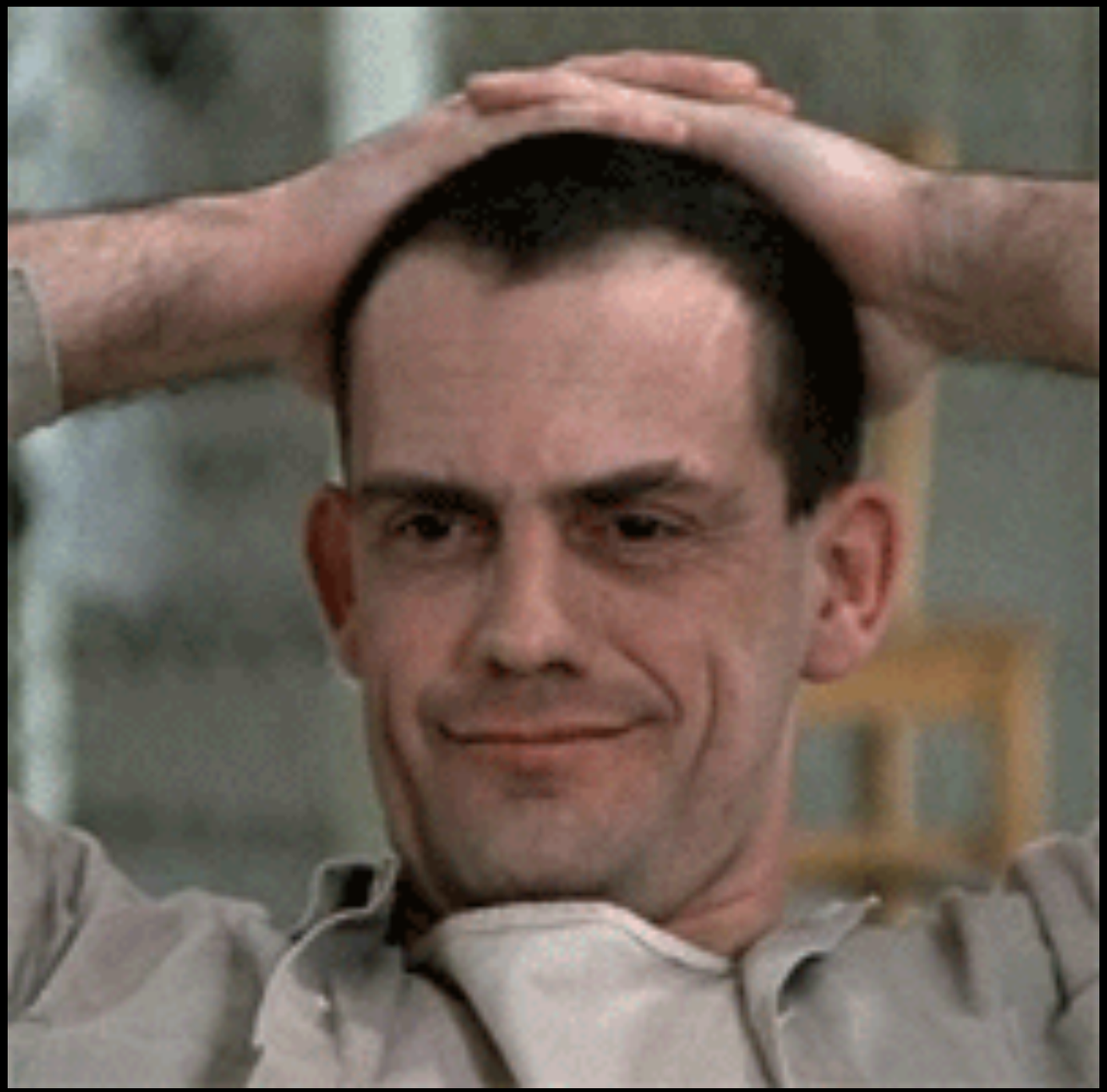
- We don’t want users to have to leave the application they’re in to use another tool to view the details from the contracts. That can destroy anonymity.

- We could use tags in advertisement information for things like images and audio or video and store data with peers, however…

- If we store data in single places that may give information away about the users and create bottlenecks and failure points for data.

DILEMMA

DILEMMA



DISTRIBUTED ANONYMOUS STORAGE

DISTRIBUTED ANONYMOUS STORAGE

ERASURE CODE

- An erasure code transforms messages of *k* symbols into longer messages with *m* symbols such that that original message can be recovered from a subset of *m* symbols.
- Optimal codes have the property that any *k* of *m* is sufficient to recover the original message.
- Optimal codes are costly in terms of CPU and memory usage when *m* is large.

ERASURE CODE

- An erasure code transforms messages of *k* symbols into longer messages with *m* symbols such that that original message can be recovered from a subset of *m* symbols.
- Optimal codes have the property that any *k* of *m* is sufficient to recover the original message.
- Optimal codes are costly in terms of CPU and memory usage when *m* is large.

VENDERMONDE MATRIX

1	1	1
1	2	4
1	3	9
1	4	16
1	5	25
…	…	…

L	O	V	E
F	R	O	M
D	A	W	N

=

L+f+d	o+r+a	v+o+w	e+m+n
L+2f+4d	o+2r+4a	v+2o+4w	e+2m+4n
L+3f+9d	o+3r+9a	v+3o+9w	e+3m+9n
L+4f+16d	o+4r+16a	v+4o+16w	e+4m+16n
L+5f+25d	o+5r+25a	v+5o+25w	e+5m+25n
…	…	…	…

As long as the second column has no repeated elements, the matrix is invertible.

REASSEMBLY

? x = ?

REASSEMBLY

? x = ?

As long as we know which rows were chosen

REASSEMBLY

? x =

As long as we know which rows were chosen

REASSEMBLY

? *x* =

L+f+d	o+r+a	v+o+w	e+m+n
L+2f+4d	o+2r+4a	v+2o+4w	e+2m+4n
L+4f+16d	o+r4+16a	v+4o+16w	e+4m+16n

As long as we know which rows were chosen

REASSEMBLY

? *x* =

L+f+d	o+r+a	v+o+w	e+m+n
L+2f+4d	o+2r+4a	v+2o+4w	e+2m+4n
L+4f+16d	o+r4+16a	v+4o+16w	e+4m+16n

As long as we know which rows were chosen, we can multiply both sides by the inverse of the Vendermonde matrix.

REASSEMBLY

$$x =$$

L+f+d	o+r+a	v+o+w	e+m+n
L+2f+4d	o+2r+4a	v+2o+4w	e+2m+4n
L+4f+16d	o+r4+16a	v+4o+16w	e+4m+16n

As long as we know which rows were chosen, we can multiply both sides by the inverse of the Vendermonde matrix.

REASSEMBLY

1	1	1
1	2	4
1	4	16

X =

L+f+d	o+r+a	v+o+w	e+m+n
L+2f+4d	o+2r+4a	v+2o+4w	e+2m+4n
L+4f+16d	o+r4+16a	v+4o+16w	e+4m+16n

As long as we know which rows were chosen, we can multiply both sides by the inverse of the Vendermonde matrix.

REASSEMBLY

1	1	1
1	2	4
1	4	16

X =

L+f+d	o+r+a	v+o+w	e+m+n
L+2f+4d	o+2r+4a	v+2o+4w	e+2m+4n
L+4f+16d	o+r4+16a	v+4o+16w	e+4m+16n

As long as we know which rows were chosen, we can multiply both sides by the inverse of the Vendermonde matrix.

Using the result, we can solve for X , giving us

REASSEMBLY

1	1	1
1	2	4
1	4	16

X =

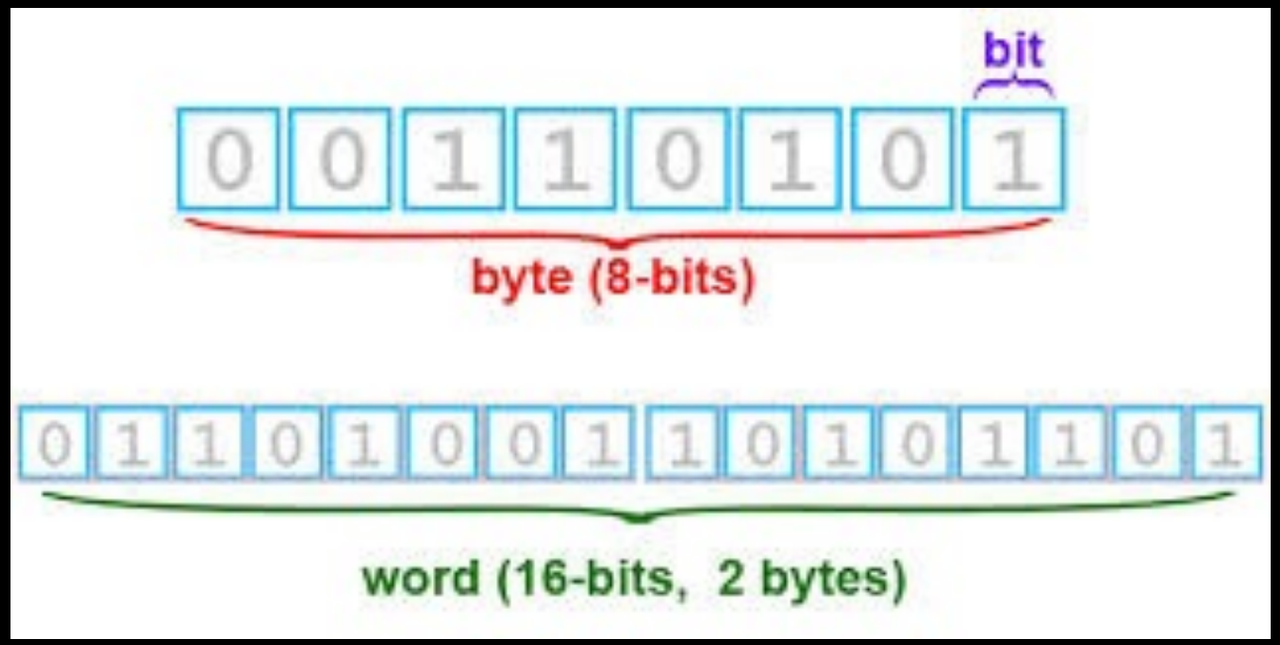
L+f+d	o+r+a	v+o+w	e+m+n
L+2f+4d	o+2r+4a	v+2o+4w	e+2m+4n
L+4f+16d	o+r4+16a	v+4o+16w	e+4m+16n

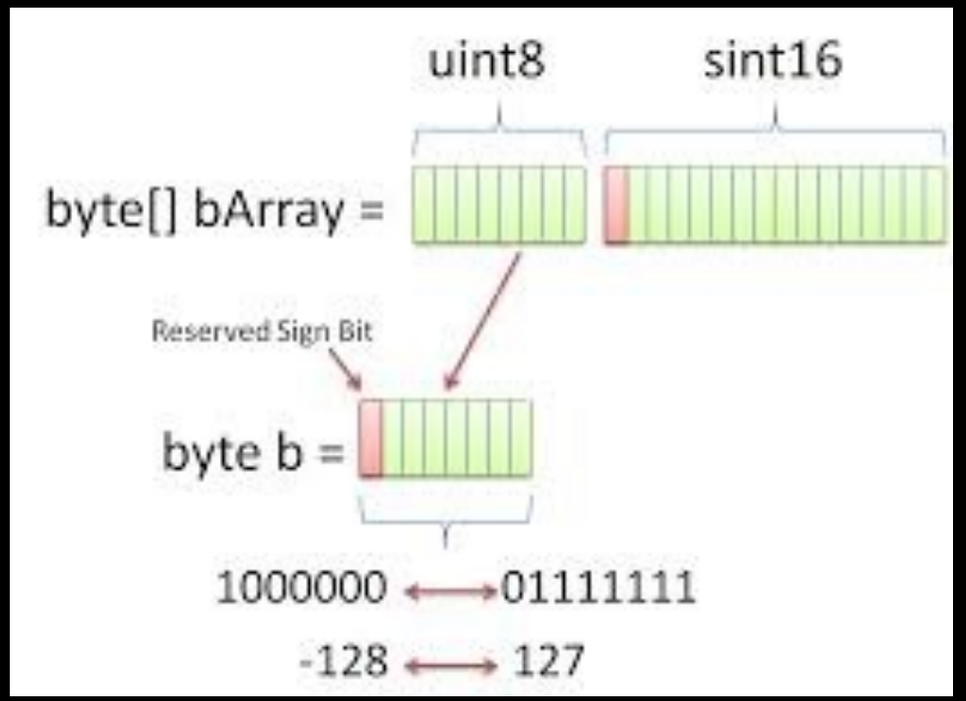
As long as we know which rows were chosen, we can multiply both sides by the inverse of the Vendermonde matrix.

Using the result, we can solve for X , giving us

Lovefromdawn

BYTE ARRAY LIMITATIONS





- A byte is the smallest addressable space in most computer architectures.
- A byte is 8 bits, thus the binary value [1111 1111] == 255 in decimal.
- Using standard integer arithmetic, the resulting matrices could end up with values larger than 255, which can’t be stored in a byte array.

FINITE FIELD THEORY

ÉVARISTE GALOIS (25 OCTOBER 1811 - 31 MAY 1832)

- A “Field” is a mathematical structure that has addition, a 0, multiplication, a 1, and a reciprocal for every non-0 element.
- A “Finite Field” is different from standard integer arithmetic, in that there are a limited number of elements in the field, and all operations performed in the field must result in an element in that field.
- A finite field with p^n elements is denoted as GF(p^n), and is also called the “Galois Field” There exists a special field that resolves our limitations GF(2**8)[1]
- In this field, 0 is the 0, 1 is the 1, addition is bitwise exclusive-or (so every number is its own additive),
- Once we operate over this field, all matrix values are guaranteed to be in the range [0, 255], and our method for bytewise encoding and decoding becomes easy.

[1]ALSO KNOWN AS G(2^8)

GF(2^8) FINITE FIELD IN C

Add, subtract, and multiply numbers in Rijndael's finite field:

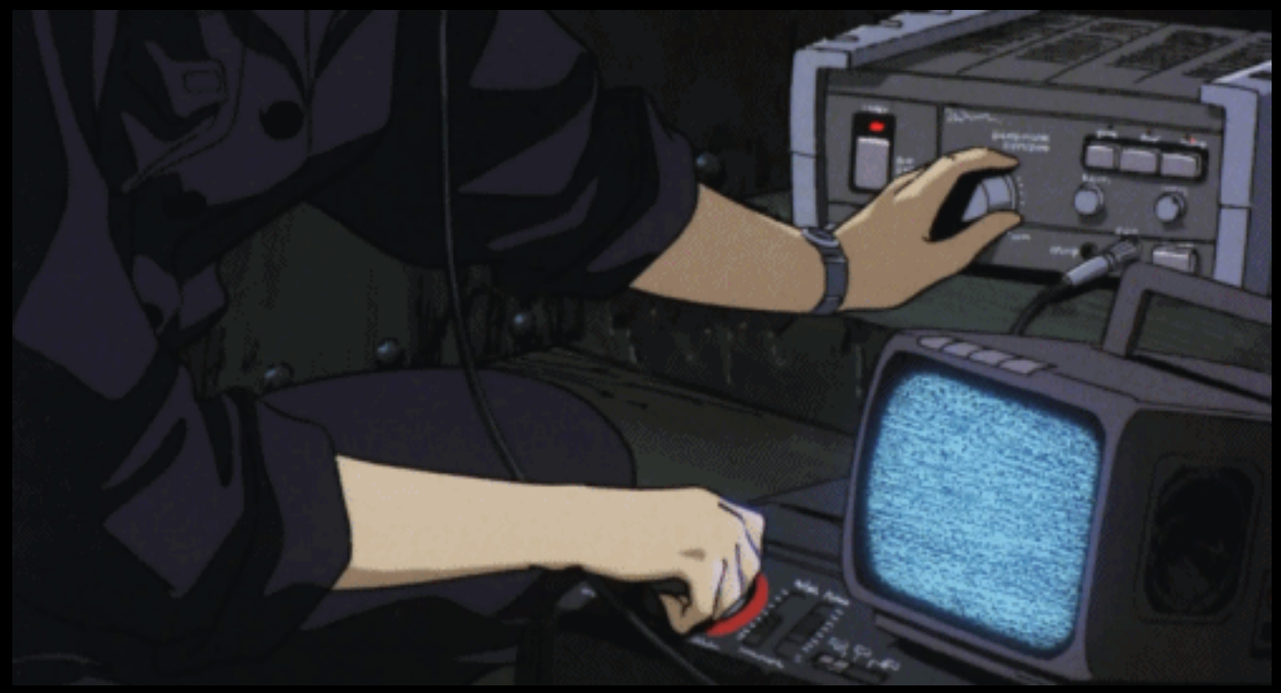
```c
/* Add two numbers in a GF(2^8) finite field */
uint8_t gadd(uint8_t a, uint8_t b) {
	return a ^ b;
}

/* Subtract two numbers in a GF(2^8) finite field */
uint8_t gsub(uint8_t a, uint8_t b) {
	return a ^ b;
}

/* Multiply two numbers in the GF(2^8) finite field defined
 * by the polynomial x^8 + x^4 + x^3 + x + 1 = 0 */
uint8_t gmul(uint8_t a, uint8_t b) {
	uint8_t p = 0;
	uint8_t counter;
	uint8_t carry;
	for (counter = 0; counter < 8; counter++) {
		if (b & 1)
			p ^= a;
		carry = a & 0x80;  /* detect if x^8 term is about to be generated */
		a <<= 1;
		if (carry)
			a ^= 0x1B; /* replace x^8 with x^4 + x^3 + x + 1 */
		b >>= 1;
	}
	return p;
}
```

GF(2^8) FINITE FIELD IN C

Add, subtract, and multiply numbers in Rijndael's finite field:

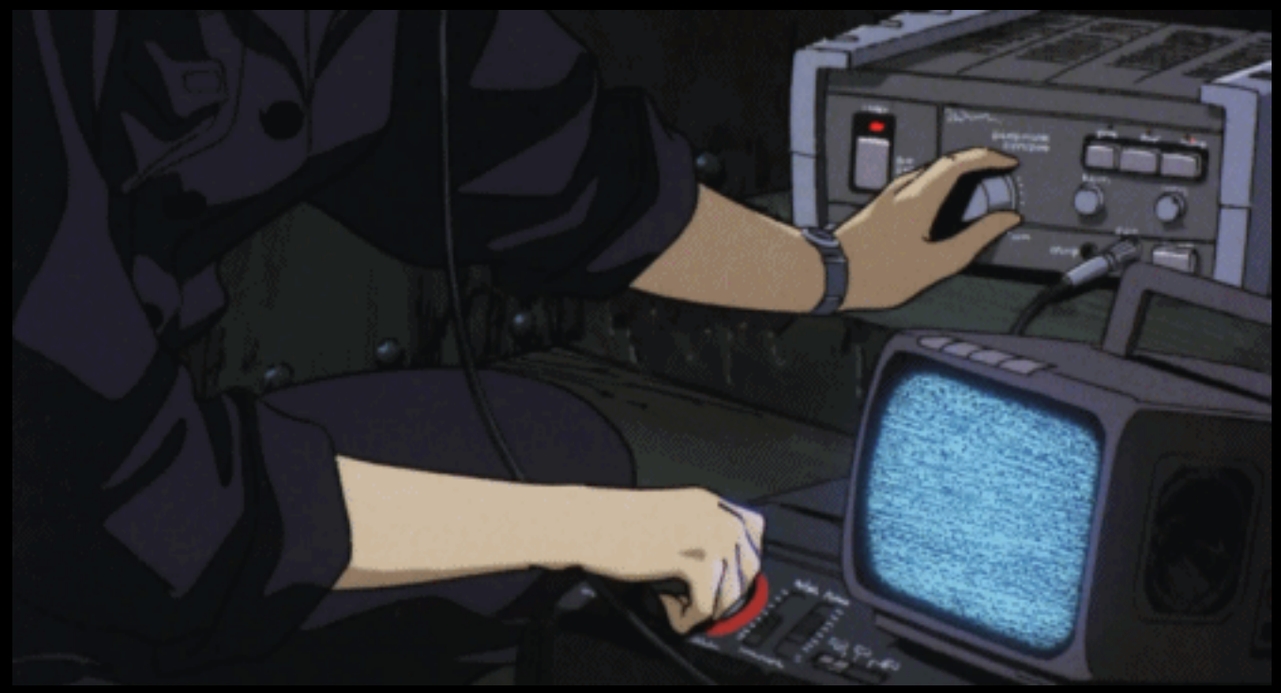
```c
/* Add two numbers in a GF(2^8) finite field */
uint8_t gadd(uint8_t a, uint8_t b) {
	return a ^ b;
}

/* Subtract two numbers in a GF(2^8) finite field */
uint8_t gsub(uint8_t a, uint8_t b) {
	return a ^ b;
}

/* Multiply two numbers in the GF(2^8) finite field defined
 * by the polynomial x^8 + x^4 + x^3 + x + 1 = 0 */
uint8_t gmul(uint8_t a, uint8_t b) {
	uint8_t p = 0;
	uint8_t counter;
	uint8_t carry;
	for (counter = 0; counter < 8; counter++) {
		if (b & 1)
			p ^= a;
		carry = a & 0x80;  /* detect if x^8 term is about to be generated */
		a <<= 1;
		if (carry)
			a ^= 0x1B; /* replace x^8 with x^4 + x^3 + x + 1 */
		b >>= 1;
	}
	return p;
}
```

LETS RECAP



- We have a transport protocol that maintains anonymity.
- We want to be able to broadcast binary data with a size constraint of 256kb per message.
- We have the algorithms available for being able to split files into pieces with fault tolerance.

LETS RECAP



- We have a transport protocol that maintains anonymity.
- We want to be able to broadcast binary data with a size constraint of 256kb per message.
- We have the algorithms available for being able to split files into pieces with fault tolerance.

PIECING IT ALL TOGETHER

- Open Transactions Contracts
- Moneychanger as a reference client
- libbmwrapper for Bitmessage integration.
- libnmcrpc for Namecoin integration.
- C++11 threading facilities.
- fecpp for Forward Error Correction with SSE2 support.

PIECING IT ALL TOGETHER

- Open Transactions Contracts
- Moneychanger as a reference client
- libbmwrapper for Bitmessage integration.
- libnmcrpc for Namecoin integration.
- C++11 threading facilities.
- fecpp for Forward Error Correction with SSE2 support.

CONCEPTS

CONCEPTS

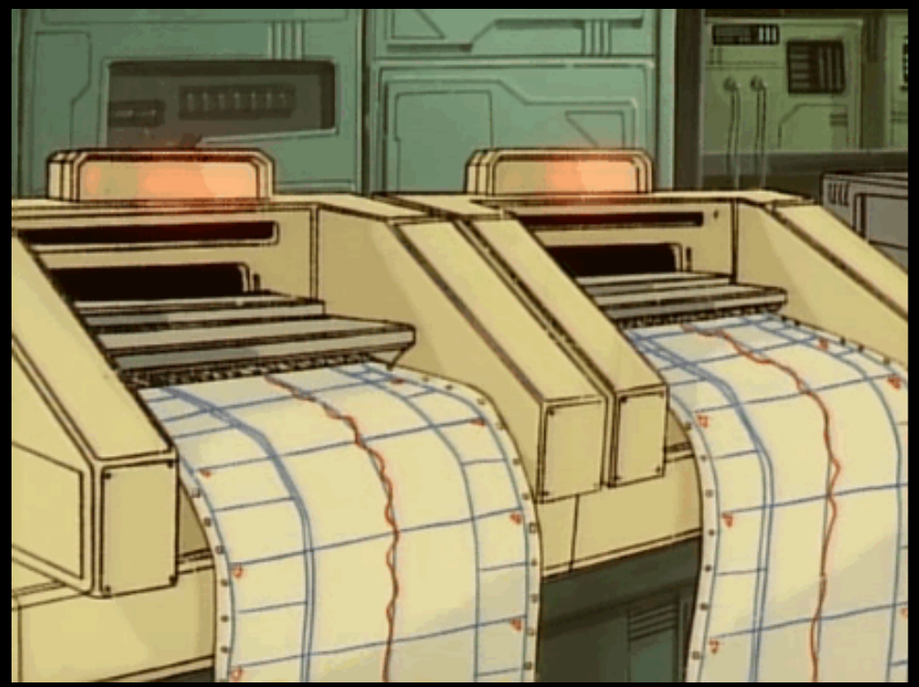
UPLOADING DATA

- We don’t want every host to have every piece of every file, This defeats the purpose of splitting it up.
- We know that we only need a subset of the total set of data to reassemble our file.
- We can split up our uploads across channels, which are created in a deterministic way.
- Files that already exist on the network should not be uploaded twice.
- Files can be appended with a header describing what file they belong to using a Merkle Tree as a solution to determining which pieces belong to which files quickly.
- Uploading a file will still take the longest to complete, but the 40% “mod” will make this negligible.

UPLOADING DATA

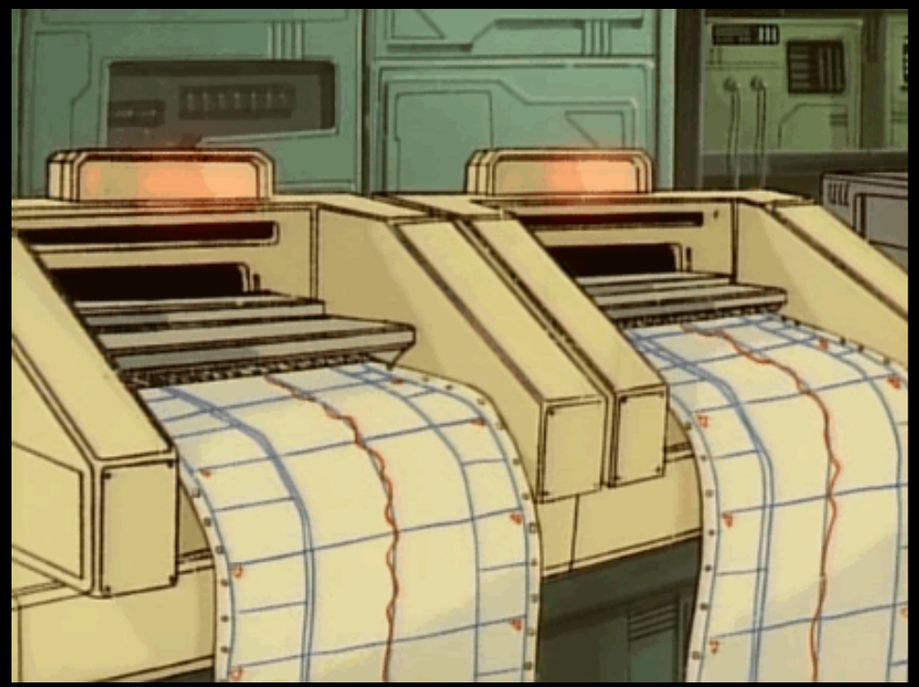
- We don’t want every host to have every piece of every file, This defeats the purpose of splitting it up.
- We know that we only need a subset of the total set of data to reassemble our file.
- We can split up our uploads across channels, which are created in a deterministic way.
- Files that already exist on the network should not be uploaded twice.
- Files can be appended with a header describing what file they belong to using a Merkle Tree as a solution to determining which pieces belong to which files quickly.
- Uploading a file will still take the longest to complete, but the 40% “mod” will make this negligible.

RETRIEVING DATA



- A requester should only have to send two messages - once when a file is requested, and again when they have all the pieces they need.
- A broadcast request will contain the requesters address for the file, so that the application knows how to file it accordingly and should only be used once.
- The more pieces a client has, the faster it can reassemble it.
- If a client has sufficient information to request a file, they will have the information necessary to know how many pieces they need.
- Clients can’t stop an upload in progress with the reference client.

RETRIEVING DATA



- A requester should only have to send two messages - once when a file is requested, and again when they have all the pieces they need.
- A broadcast request will contain the requesters address for the file, so that the application knows how to file it accordingly and should only be used once.
- The more pieces a client has, the faster it can reassemble it.
- If a client has sufficient information to request a file, they will have the information necessary to know how many pieces they need.
- Clients can’t stop an upload in progress with the reference client.

WHAT WE DON'T KNOW

- How many pieces is the file broken into.
- How many pieces they have received.
- What pieces they have received.
- How many pieces they need to reconstruct the file.

WHAT WE DON'T KNOW

- How many pieces is the file broken into.
- How many pieces they have received.
- What pieces they have received.
- How many pieces they need to reconstruct the file.

CLEANING UP

- Requesters should broadcast when they are done with reassembly.
- This benefits both uploaders and the requester who may otherwise be flooded with data they don’t need.
- Users should set a short TTL on file requests, and rebroadcast as needed.
- Uploads should have a TTL for network cleanup, with users being able to accept or deny files that exceed certain sizes.

CLEANING UP

- Requesters should broadcast when they are done with reassembly.
- This benefits both uploaders and the requester who may otherwise be flooded with data they don’t need.
- Users should set a short TTL on file requests, and rebroadcast as needed.
- Uploads should have a TTL for network cleanup, with users being able to accept or deny files that exceed certain sizes.

CURRENT STATE

- Moneychanger building against Monetas opentxs
- libbmwrapper fully compliant with BitMessage v2 and is thread-safe.
- BitMessage works in Moneychanger
- Forward Error Correction implemented and tested over network with binary files (mp3 and images).
- fecpp converted to CMake build system.
- Command Line GUI using ncurses
  (cross compiles to SDL) - “bmfec” on Github.
- Binary pieces are binary compatible with the Tahoe-LAFS filesystem.

IN PROGRESS

- How many channels to keep open for data broadcasts.
- Formatting the header structure of a broadcast.
- Automate filesystem handling of stored file pieces.
- Determine best options for files behind the scenes (*k* of *m*).
- Testing on Linux
- Converting libnmcrpc to CMake build system.

IN PROGRESS

- How many channels to keep open for data broadcasts.
- Formatting the header structure of a broadcast.
- Automate filesystem handling of stored file pieces.
- Determine best options for files behind the scenes (*k* of *m*).
- Testing on Linux
- Converting libnmcrpc to CMake build system.

PLANS

- A headless user agent similar to ssh-agent or gpg-agent
- Transitioning OT Client features to use this user agent, ie. Moneychanger would talk to the agent.
- Keeping Moneychanger up to date with the latest opentxs and updating the stable-client branch of opentxs as tests are completed.
- Updating libbmwrapper to be fully compliant with Bitmessage v3
- Setting up nodes on cloud instances to act behind the scenes on the Bitmessage network.
- Integrate “Bitcloud” features into Moneychanger (one as a file manager, another for the Bazaar).

CONCLUSION

CONCLUSION

QUESTIONS?

yamamushi (at) gmail.com
PGP: 6E1DAA32

QUESTIONS?

yamamushi (at) gmail.com
PGP: 6E1DAA32

REFERENCES

- http://en.wikipedia.org/wiki/Dunbar's_number - Dunbar’s Number
- http://opentransactions.org/ - Open Transactions Wiki
- https://github.com/Open-Transactions/ - Open Transactions Projects
- https://bitcointalk.org/index.php?topic=212490.0 - “Holy Grail” post.
- http://lasindias.com/the-p2p-mode-of-production - Decentralized vs. Distributed
- https://bitmessage.org/ - Bitmessage homepage.
- https://openbazaar.org/ - OpenBazaar homepage.
- http://namecoin.info/ - Namecoin homepage
- http://en.wikipedia.org/wiki/Merkle_tree - Merkle Trees
- http://en.wikipedia.org/wiki/Erasure_code - Erasure Codes
- http://blog.richardkiss.com/?p=264 - Richard Kiss’s blog article covering zfec and the Vendermonde Matrix
- http://en.wikipedia.org/wiki/Vandermonde_matrix - Vendermonde Matrix
- http://www.amazon.com/Hackers-Delight-Edition-Henry-Warren/dp/0321842685 - Hackers Delight by Henry S. Warren, Jr.
- https://www.tahoe-lafs.org/trac/tahoe-lafs - Tahoe-LAFS homepage.
- http://static.benet.ai/t/ipfs.pdf - IPFS decentralized filesystem
- https://github.com/yamamushi/libnmcprc - libnmcrpc
- https://github.com/yamamushi/bmfec - Bitmessage Forward Error Correction
- http://www.reddit.com/r/reactiongifs and http://www.reddit.com/r/cinemagraphs - For all those gifs