**A Decentralised Social Network**

Keeping the web social, open and free from corporations and government.

Rhys Laval

**Abstract**

This project covers the creation of a social network, which is free, open source and decentralised (peer to peer). Alongside will be the research of security, privacy and the social impact. Looking into how the technology will work and to create a basic implementation based on my findings. The deliverables will be a code project that will be open source. The first implementation will include a basic text module to act as a proof of concept. This will be used to show the modularity of the system.

UFCFR4-45-3 - Computing project

BsC (Hons) Computer Science, 11014931

University of West of England, 2014/15

Word count: 21,694

**Acknowledgements**

I would like to thank my parents and friends for their patience whilst also pushing me to finish this project and continuing with education.

Also I would like to thank Frank Maddix for his help over the past year and

Jeff Graham for his great idea of the brand ‘Yella’ which one-day I hope to bring to fruition with the product.

‘Our freedom can be measured by the number of things we can walk away from.’

Vernon Howard

Table of Contents

2 Introduction 6

2.1 The state of the current Internet 6

2.2 The Technological Side 7

2.3 The Social Impact 8

2.4 Privacy 10

2.5 Aims and objectives 11

3 Background & Motivation 13

3.1 Other social networks 13

3.1.1 Facebook 13

3.1.2 Diaspora 14

3.1.3 Ello 14

3.1.4 Comparison 15

3.2 Peer-to-peer networks and ideas 16

3.2.1 Napster 16

3.2.2 Tor 18

3.2.3 OpenBazaar 18

3.2.4 BitTorrent 19

3.2.5 BitCoin 20

3.2.6 Storj.io 21

3.2.7 MaidSafe 21

3.2.8 Peer-to-peer investing, banking 22

3.3 Motivation 23

4 Research 24

4.1 Research strategies 24

4.2 Questionnaire 24

4.3 Survey Results 27

4.4 Peer to peer (P2P) methodologies 30

4.4.1 Bootstrapping 30

4.4.2 Distributed Hash Table 30

4.4.3 Chord 31

4.4.4 Kademlia 32

4.4.5 Tapestry 35

4.4.6 HyperCuP **Error! Bookmark not defined.**

4.4.7 CAN (Content addressable network) **Error! Bookmark not defined.**

4.4.8 Issues 35

4.4.9 Conclusion 37

4.5 Social aspects 37

4.5.1 Depression 38

4.5.2 Addiction 38

4.5.3 Envy 40

4.5.4 Other issues 40

4.5.5 Conclusion 41

4.6 Privacy 41

4.6.1 Conclusion 44

4.7 Security and Web of Trust 44

4.7.1 Pretty Good Privacy (PGP) 44

4.7.2 Web of Trust 45

4.7.3 Blockchain 45

4.8 Open Source Licensees 45

5 Requirements & Analysis 47

5.1 Introduction & Overall Description 47

5.1.1 Purpose & Product Scope 47

5.1.2 Product Perspective **Error! Bookmark not defined.**

5.1.3 Product Functions 47

5.1.4 User Classes and Characteristics 47

5.1.5 Operating Environment 48

5.1.6 Design and Implementation Constraints 48

5.1.7 User Documentation **Error! Bookmark not defined.**

5.1.8 Assumptions and Dependencies 48

5.2 External Interface Requirements 49

5.2.1 User Interfaces 49

5.2.2 Software Interfaces **Error! Bookmark not defined.**

5.2.3 Communications Interfaces **Error! Bookmark not defined.**

5.3 System Features 49

5.3.1 Peer-to-peer DHT 49

5.3.2 Social Network 50

5.3.3 Security 50

5.4 Other Non-functional Requirements 51

5.4.1 Performance Requirements 51

5.4.2 User Interface Requirements 52

5.4.3 Safety Requirements 54

5.4.4 Security Requirements 54

5.4.5 Software Quality Attributes 54

5.5 Other Diagrams 55

5.5.1 Use Case Diagram 55

5.5.2 Sequence Diagram 55

5.5.3 Communication Diagram 56

6 Methodology 57

6.1 Mono C# 57

6.2 Client/server side communication 58

6.3 Client Side JavaScript 58

6.4 Local Database 60

6.5 Testing 60

6.6 Peer 2 Peer Libraries 60

6.7 Chat 60

6.8 Other Methodologies 61

6.9 Development Methodology 61

7 Design 63

7.1 Company Brand 63

7.2 Component Design of the Backend 63

7.3 Component Design of the Angular.JS 64

7.4 Class Design 65

7.4.1 Yella 66

7.4.2 DHT 66

7.4.3 StaticFileServer 69

7.4.4 WebSocketHost 69

7.4.5 Encryption 69

7.5 Sequence Diagram 70

7.6 User interface Design 71

7.7 Protocol Messages 73

7.7.1 Find Node 73

7.7.2 Find Value 73

7.7.3 Store 73

7.7.4 Ping 74

7.7.5 Node/Key ID’s 74

7.8 Stored Data Types 74

7.8.1 Example User Profile 74

7.8.2 Example Status 75

7.8.3 Example Linking Data Structure 75

8 Implementation 76

8.1 Version Control Software & Build Server 76

8.2 Self-hosting Server 77

8.3 Server-side Web sockets connection 77

8.4 Website 78

8.5 Complete Website Designs 80

8.6 Angular.JS 87

8.7 Ui Router 87

8.8 LocalStorage 88

8.9 Module System 89

9 Testing 93

10 Conclusion 96

10.1 Future Work 97

10.2 Evaluation 97

10.3 Reflection 98

10.4 Closing Statement 98

11 Appendix 100

11.1 Appendix – Test Cases 103

12 Bibliography 113

# Introduction

Websites available for social networking at the moment are generally closed source. They are run by large corporations, which use the data they collect to pay for the hosting and the development of their websites. These websites are meant to be social and should follow human rights, which privacy is a key factor. Why not use the social aspect to design, create and host the site, which has shown to work well on many Open Source projects?

## The state of the current Internet

As cloud technologies are developing and are now the buzzwords of our generation. A few large centralized websites are now dominating the Internet space, which are ran by a few corporations. These large companies such as Amazon, Facebook and Google are now taking a hold of the Internet. Data centres are using more than 1.3% of the world’s bandwidth and using as much power as a small country (Newman, 2011). Out of the top 100 websites, only one is not a large corporation, which is Wikipedia. It has no adverts and runs on donations.

Has the current progress of the Internet started to stagnate? Do the current users like to consume the content they have available in the means they do? With more and more adverts hidden away in content, which is unknown to users. Our data is being harvested to collect statistics on who we are and to put us into groups of users to target specific adverts in the future.

The Internet was created as a decentralized web. Now it has changed into a web where the big players own a massive market share. They can push in any direction and the web and its users follow them.

In Tim Berners-Lee’s book (Berners-Lee, 1999), he mentions the first World Wide Web conference at CERN in 1994. Those technologists cannot leave the social and ethical questions to other people because the changes in technology directly affect us all. This is a starting to create change, putting the control into the peoples’ hands, which has worked well for millions of open source projects that have created operating systems, web browsers and languages which power the web. The Internet was created with the thought of “permission less innovation” where there is no central ownership or control but this is also under threat. The vision of Tim Berners-Lee had of the Internet is that it stays under no control and that our data is kept private (Berners-lee & Fischetti, 1999). An example is that by uploading images to Facebook you are giving them full rights to use the user’s images’ in advertising and other forms.

Does anything need to be done?

* Is the Internet going to turn into a dirigisme affair? It is about the idea of a capitalist economy with strong state links.
* Shall innovation for the Internet stagnate after the last ten years of massive changes?
* The powers that are increasingly taking over our privacy, destroying our basic human rights, should this happen?
* Should the WWW be kept free, respectful and balanced?

## The Technological Side

Innovation has slowed down recently but the nature of the web has carried on. A social network that shall not be reliant on classical server architecture topology but also a free, self-sufficient network that is social and open does not exist. There are more advantages where no one has control and the network is owned by its users and can be modified by its users. Instead of using huge server farms, which already consume a large proportion of the world’s electricity; why not use the idle computer time.

The Internet has grown far passed the use cases it was first created on. The protocols were designed with small networks in mind and IPv4 running out of address space shows this. On the other hand they were designed well and have lasted well over the years with limited changes needed.

The technology that could be used is already around. These are similar algorithms to how BitTorrent works with distributed hash table (DHT). Kademlia is a DHT for peer-to-peer communication which incorporates an algorithm. It calculates how to keep the data across the world in as little hops as needed and keeps the data up to date across the world. Napster used technology similar to this idea, except for the fact that searching was done centrally.

A disadvantage is that more data processing would be needed as the computers need to communicate more. In the future, if net neutrality is affected, it would affect peer-to-peer communication in an unknown way depending on how they filter and regulate the traffic.

## The Social Impact

If anyone mentions they do not have Facebook, they are immediately looked down as if to say, “How do you live without the social network”? That really demonstrates how it has ingrained itself in current society.

On the social and technological side of things a decentralized network is needed, similar to how the infrastructure is maintained. I think a neo-liberal approach would work well as a network, which can be modified by its members, and the views of all can be taken into account. A network, which is regulated by its members, similar to how other open source communities self-regulate themselves. This gives more accountability as everyone can see what the code does and how it works.

It could also combat censorship which is on the increase across the world. Censorship across the world is changing rapidly, from the great firewall of China to Iran where Facebook is blocked. FireChat was used at the Hong Kong protests during 2014, which was called the umbrella revolution (Meyer, 2014). It allowed the activists to communicate in a secure fashion without the need for any central server, which could be taken down in a moment’s notice by the firewall or the state. FireChat works by creating a mesh network through Wi-Fi or Bluetooth, so crucially even the Internet was not needed.

To keep the privacy of user, he/she should own the rights to the data and the network should not take ownership. Functionality could be created so that the user can port the data away in a format. This could be created as a standard for social networks. Data portability is a new idea which means to be able to transfer from one service to another service. It has been added to the EU’s new data protection proposals (European Union, 2014).

Most networks have struggled with keeping each profile as a real persona. In an open social fashion this would be harder to maintain. A person could even create a fake persona of themself.

Many sites work well such as Reddit but they suffer trolls which aim to cause trouble. Either by upsetting people, creating arguments or disrupting the normal flow. Would it need moderators? Who would they choose? What is allowed and what is not? Peer-to-peer networks generally have the wrong view as it is is used to infringe copyright laws. How would DMCA (Digital Millennium Copyright Act) take down requests work? Would the moderators be allowed to remove material that infringes copyright law? An example here is the “the fappening”, a scandal of a release of nude celebrity pictures which caused issues for 4chan. 4chan is an image-based forum launched in 2003 with no registration. Before the scandal, it had never had a DMCA page for requests. Sequentially, it had to add a page due to the celebrities taking legal action for abuse of their personal privacy (BBC, 2014).

Social media generally shows the good side of people’s lives and shadows the bad times. It is similar to the filter bubble and you are really shown this rose coloured sceptical view of the world that really has no fragmentation. People generally view the feed of latest posts, thinking their friends are having fun without them and this can cause envy and stress. This also creates an addiction, feeding from gossip, which travels through these networks. It is better to be with friends than creating a life around the computer.

Would a kill switch be needed and possible? What would happen if a huge Internet frenzy occurred on the network and how would this be controlled? If for example a virus or malware spread through the network, how could this be stopped and is it possible?

The social-technical gap needs to be filled with a network which looks more towards the social side than one which causes antisocial and angst. The Internet should be the driving force between bringing human civilisation together. In 2000 some looked at the Internet at a dystopian place which did not have trust; nowadays some would argue the trust is in the wrong places (Whitworth, 2006).

“To realize the social potential, software designers may need to recognize how societies generate nonzero-sum gains”.

(Wright, 2000)

The Internet has properties similar to a human brain, in the connected sense. The neurons are a user or a computer and the synapses are like the networks, which join the computers together. This analogy this could help us to understand the world better and to look at how we could communicate better to find a more enlightened world. When looking at a virtual wiring diagram of a rat’s brain researchers found similarities to a Rats brain. (Perkins, 2015).

## Privacy

A change in our privacy online is needed. On one side are the major Internet leaders saying “privacy is dead” and the common retort “if you are not doing anything wrong, you have nothing to worry about”. The opposite argument is that it is easy to defame the identity of a person or a corporation online. These views degrade our rights as humans and do not allow humans to veer off the line of the dominant culture. Computers have allowed creativity and more recently via the use of the Internet as a median to share the music. “Do we fight pirates, or do we learn from them?” a quote from the book ‘A Pirates Dilemma’ (Mason, 2008).

Cardinal Richelieu wrote, “Give me six lines written by the most honest man in the world, and I will find enough in them to hang him.” This shows that privacy is needed in the present times and more so now, as laws and the culture changes at a pace never before seen.

Government control is massive on the Internet and has been shown by the Snowden revelations that have happened recently such as PRISM. In 2001, the European Parliament issued a paper on ‘Echelon’. It finished with surveillance would be limited because of the amount of data created but now we have moved forward with technology which means this data can be monitored in huge amounts (Schmid, 2001). Advanced artificial intelligence can be used to pick and choose which data is important to be monitored via keywords.

Surveillance is used on the corporate side, which is called Data mining. It occurs on all large networks because it is a way to generate money. They push adverts relevant to the user by looking at relevant trends and which in turn increases the company’s profit because the users are more likely to click on the advert.

## Aims and objectives

Many social networks exist and more are popping up every day. Despite many of them having no unique selling point. The main aim of this final project is to turn the model upside down of social media websites and have every part of the data of the social network move around the user’s computers instead of central servers. In turn that means control is ironically in the users hands. My background research reveals that there is little regarding a distributed website, in a sense of that a user’s computer has part of the website. This dissertation will delve into the technical side mainly. It will also cover the research of the social, ethical aspect and the open source culture to power the forth mentioned.

Objectives:

* To research how a decentralised website would work that is not static. Also, how the data would be passed around to all clients to be highly available.
* To create a basic implementation of a social network with a basic interface via a website. Users can interact with each other like twitter.
* The project will be designed with no central server architecture.
* It will be created to be modular in the future. The prototype will contain one module of a status module but so it can be extended for images, video and other content.
* It will be created in an open source model under a licensee, which allows other’s to take the projects code to create other projects in the future.
* Research the sociocultural and anthropological side of the network.
* A further objective is to research into how security and privacy would work. How this could work with the data being spread over the network. The first development will be working on the data being unencrypted. Over time this will change into an encrypted model where users can select who can see the data for example, their friends or everyone.

This report shall cover all aspects of delivery of a software solution from start to finish, which will act as a proof of concept, which could inevitably be brought forward into a real world example. With a cornucopia of references, and appendices to support the evidence gained and interpreted in the author’s own way.

# Background & Motivation

The modern WWW (World Wide Web) requires to evolve continuously and to be open to change as to stop it stagnating. The social networks are a key part of present-day society.

## Other social networks

Here is a brief summary of social networks. I have covered the largest, newest and a network that is towards my aims but is not fully decentralised.

### Facebook

Facebook started as a social network for College students, began at Harvard University in Massachusetts. It is the only social network in the top 50 sites in the world. In 2004 it had 1 million members, which increased to 1250 million by 2014. This is 1 in 13 people on Earth! A film has been made which portrays how Mark Zuckerberg created the website. It showed the struggles Mark went through to create the site. One of these was a conflict with how ideas for the site were from the Winklevoss twins. Currently, the company is on the stock market but without a troubling start when it was floated. It has gained a huge audience and great popularity amongst first world countries and is now starting to take hold in Africa as mobile Internet is spreading quickly.

One of the reasons Facebook is cited to have grown so quickly is way they translated the website quickly into other languages. Users were asked to submit translations for words or phrases and then it is put forward for a vote and if enough votes were obtained it would be published. In 2009 they even filed a patent application, which was approved (Yishanl, Wong, & al., 2010).

“Doing a privacy change for 350 million users it is not the type of thing many companies would do, we decided that these would be the social norms now and we just went for it” said Mark Zuckerberg in the film “Terms & Conditions may apply”. This shows the attitude that large companies have as they know they have a large enough market share to make changes without issues (Hoback, 2013). This fact when put in context that when buying his house, he bought the four houses surrounding his house to ensure his complete privacy, for a total of $30 million (Evans, 2015). This shows his own personal views may be different.

### Diaspora

The word Diaspora means ‘the dispersion of Jews beyond Israel’ (Dictionary.com Unabridged). It also means the spread of people from their homeland. The word defines the website as it is a social network which is partly decentralized. There are pods, which you can join (the pod is a group of servers). Each of these pods may interact with other pods to create the full network or be a separate social network. As of 10th of November 2014, there are more than 1.1 million members spread across 246 pods (Robinson, 2014). They emphasise three words as a philosophy of the network, which are: decentralization, freedom and privacy.

It is an open source project, which was started, by Ilya Zhitomirskiy and 3 other students at New York University’s Courant Institute of Mathematical Sciences in 2010. In 2011 a kick-starter campaign was commenced which had $190,000 more than the $10,000 they wanted to pay for full time coders to get the minimum value product up and running.

### Ello

This relatively new social network has gained a lot of traction recently. Its motto is “Simple, beautiful & ad-free.”

The following quote was on their front page:

“Your social network is owned by advertisers. Every post you share, every friend you make and every link you follow is tracked, recorded and converted into data.”

(Ello, 2014)

Even without advertising, which is the general way that websites make money, Ello has managed to get Venture capital funding for their project. They said that they will never take on advertising but it is unclear on how they will generate funds. It was launched in March 2014 and already has 1 million members. At one point it was getting 30’000 members trying to join in one hour (Smith, 2014).

### Comparison

Here is a comparison of the sites mentioned above. This is to be taken lightly as these are subjective.

Table 1 - A brief comparison of the social networks reviewed.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Out of 10 | Privacy | Data Mining | Free Speech | Security | Average |
| Facebook | 1 | 3 | 5 | 9 | 4.5 |
| Diaspora | 8 | 10 | 10 | 8 | 9 |
| Ello | 10 | 8 | 7 | 8 | 8.25 |

* Privacy is measured on:   
  10 – meaning the user has complete ownership of data but also control over their data.

1 – Means the data ownership is transferred over to the network and that you lose control of who can see your data.

* Data mining is measured on:

10 – The data is not being used to tailor adverts towards users and the data is not being sold even if anonymised.

1 – Data being used for adverts or data being sold when not anonymised.

* Free speech is measure on:

10 – Any posts will be allowed within the law.

1 – Posts have been removed in breach of free speech.

* Security

10 – SSL is enabled and they have responded in a timely fashion when an issue has been found.

1 – Issues have been found that have not been fixed with a timely fashion.

## Peer-to-peer networks and ideas

Peer to peer networks are gaining strength from where they have only been used for torrents and data networks for many years. People and companies are now seeing the advantages of using them and are branching out to use them in other software solutions such as SETI@Home. This subsection will go into some of the background research covering where peer-to-peer examples have been implemented.

### Napster

Napster is known as the original peer-to-peer file sharing service. At its peak it has over 80 million users worldwide (Gowan, 2002). Napster is still enshrined in the Guinness Book of World Records as the fastest-growing business ever (Nieva, 2013). It was developed by Shawn Fanning who was only 18 at the time with John Fanning and Shawn Parker (who after went off to help Facebook afterwards). The company lodged many patents which cover how its network works such as US patent 7310629, which covers the “Method and apparatus for controlling file sharing of multimedia files over a fluid, de-centralized network” (Mendelson & Fanning, 1999).

Dispatch is a band, which mentions that their success is due to Napster and other file sharing sites in the 2000’s. They stayed independent and were amazed at the spread of their music for which they were astounded. They would play gigs on the other side of the United States with people coming who would be able to sing their music even though they had little airtime on the radio or TV (Winter, 2013, p. 42 minutes). This was unheard of at the time and showed the changing times.

Before long Napster had two cases against them; one with Dr Dre and the other with Metallica. Metallica being the ironic case, as the genre, Metal portrays an anti-capitalist view of the world. They later hand delivered 25’000 names to Napster to make a point and so that these usernames had to be removed from the network. Dr Dre also sent DMCA (Digital Millennium Copyright Act) requests for files to be taken down which were not taken down. The RIAA (Recording Industry Association of America) started a court case against Napster which they lost. It was then brought to the Ninth Circuit where it was over ruled. Afterwards, it was challenged by the District Court that it must restrict access but unable to comply Napster had to shut down. This is still seen as one of the landmark cases in the history of the Internet.

“So let’s get to the heart of the matter. This hearing is about the future of profiteering on the mass distribution and duplication of intellectual property. It is a do-or-die situation, but the future is unclear, so let’s hear the good news first.

…

Really, the Internet is the Holy Grail of distribution channels. It is a zero marginal cost distribution channel. That means that it costs the same to transfer one copy of intellectual property as it costs to transmit 10,000 copies or 1 million copies or 10 million copies. This is truly the Holy Grail of distribution channels. There is no physical media, there is no marginal cost. We don’t have to print CD’s; we don’t have to ship CD’s. We don’t have to mine the aluminium, make the CDs, and destroy our environment.

…

Our goal is to try to establish a method by which artists, whom I call the intellectual property profiteers, can profit side by side with these people exercising their rights to listen to music in a convenient format. The question is how are we going to harness the Internet, how are we going to make the Internet work for us, us as consumers and us as intellectual property producers and owners.

…

Napster and Gnutella are but the first of a succession of technologies which will make it increasingly difficult to control the distribution of   
intellectual property.

…

If laws are enacted against these technologies, the ensuing replacement for these technologies would only be more difficult, if not entirely unfeasible, to police. This is only the beginning.

…

The bad news is that old-world tactics may no longer work on the Internet. This is the new economy."

Gene Kan, opening statement for the hearing “Music on the Internet: Is there an upside to downloading.” (US Senate, 2000)

It really shows why ‘Torrenting’ became so popular; they could shut down the sites but not the protocol. Sites start popping up left, right and centre. Recently, for profit organisations, have tried to tackle how to use the Internet as a distribution channel for music. Whilst allowing the intellectual property owners to profit from their ownership. These are programs such as Spotify, Pandora & Rdio. Even Spotify used a peer-to-peer model to start with, to keep the costs of running the company low as fewer servers would be needed. Only 8.7% of music came from Spotify's servers while the median playback latency is only 265ms (Kreitz & Niemelia, 2010)!

### Tor

It started out as a US Naval research laboratory project. They first developed onion routing in 1990’s. Onion routing is a method of dispersing the traffic through other users whilst encapsulated in layers of encryption like an onion. In 2004 it was released under free license. Then in 2006 the Tor project was founded. Tor for the most part is used to allow users to go onto the Internet whilst keeping their browsing private. This is very useful for countries where censorship is in place. There are dark places on Tor where illegal content/items is stored and sold.

The most high figure use is Edward Snowden, using the network to send the documents regarding PRSIM using a secure drop service. It has been claimed that Tor can be cracked but this requires large amounts of manpower and this will only provide details on a very small percentage of Tor users. There is lot of the examples where users have used Tor but have been arrested because of human error. Not because Tor has been hacked. An example of this, is the first version of Silk Road, which stands on a piece of evidence which says that FBI got the owners details based on a coding error where one of the Captcha images was not being provided correctly through Tor which lead to the arrest.

### OpenBazaar

OpenBazaar is described as “if eBay and BitTorrent had a baby”. This gives the idea of a decentralized marketplace for private trade online, which runs over protocols similar to BitTorrent. The protocols can be made to be protected against censorship. It started as a hack day entry at the BitCoin Expo in Toronto. At the time it was called the Dark Market with the idea of making a transparent Silk Road.

Silk Road is the most well-known black market place which is a part of the Tor network. It was first shutdown in 2013 after the FBI raided it and then it was said to be opened again with a decentralized backend. This was found not to be the case after it was closed down again in 2014 as part of a worldwide operation, which closed 414 hidden tor services.

OpenBazaar is open source with the code located on GitHub. It is licensed under the MIT license. The program connects to the OpenBazaar network, which uses Tonika and Kademlia as algorithms behind the peer-to-peer network. It uses BitCoin as its currency, which is covered later on in this chapter.

They have released a lot of the theory work behind the system. One of the interesting theory pieces which can be viewed online is “A pseudonymous trust system for a decentralized anonymous marketplace”. It would be useful to use the ideas in this paper as part of the research for the security element of the network. It covers how they are going to use the web of trust, the most well-known concept built upon Pretty Good Privacy (PGP). By authenticating with a public and private key, the web of trust works well for protection of both sides (Zindros, 2014).

### BitTorrent

The BitTorrent protocol has been around since 2001 and is used to share large files between users. They are the company behind the software and specification which powers the protocol. They released a press release about how they have more than 150 million users (BitTorrent, 2012). If you included the many other pieces of software that could be used to connect to the protocol you could gauge the network at over quarter of a billion users.

Commonly a tracker is used to search the millions of files available to be downloaded. A tracker tracks the clients on the network and the files, which are available. In the UK the largest trackers are generally blocked via our local ISP’s because of a High Court order. The largest tracker called ‘The Pirate Bay’ is now closed after many years of fighting legal battles. In 2009 the founders were found guilty for assisting copyright infringement. After many legal issues with trackers being linked to the search sites. In the last few years, they have been split into two separate domains to separate the legal issues regarding the tracking sites linking the torrent to users IPs. The first domain is the search site now just gives out the id (called the info hash) of the torrent. This is given to the public tracker which only knows about the IPs and torrent ID of the torrent which limits legal issue. PublicBitTorrent is an example of one of the open tracker projects. These new trackers do not keep the Meta data just an info hash (which is like a torrent ID) and the peers which are seeding the torrent.

In 2005 a DHT was built inside of the protocol to support distributed trackers. This means you could find peers without the use of a tracker to bootstrap the list of initial peers. This is the example of the largest DHT in the world, which the specification can be found here[[1]](#footnote-1).

### BitCoin

BitCoin was the first cryptographic currency and is still the largest in terms of size based on transactions and amount of users. It is not known who creator is. All that is know is the pseudo name he used was called Satoshi Nakamoto (Nakamoto, 2004). This could either be a collective or a single person. He kept contributing till 2010 and then slowly faded away but it is believed he is in possession of a million Bitcoin’s, which currently have the value of $1.1 billion.

The Blockchain is a key part of how BitCoin works; it is a ledger of all transactions that have occurred through the network. Anyone can access the ledger or see confirmed transactions to verify that transactions have occurred. Bitcoin’s similarly to OpenBazaar use the web of trust idea to keep the wallets (wallet is like a physical wallet, it is used to keep the coins in) safe.

The transactions whilst viewable cannot be linked to a person. There are currency exchanges that operate to allow users to exchange Bitcoin’s into fiat currency. Exchanges operating in certain countries require collecting personal information to comply with money laundering laws.

Mining is the creation of more Bitcoin’s. They help to keep the Blockchain complete by verifying new transactions and by adding them into the Blockchain tree to be verified in the future by other nodes/users.

### Storj.io

Storj.io is taking the idea of Dropbox and creating a decentralised version of it. It is using the BitCoin protocol and peer-to-peer network topology to store and retrieve data. For users giving space the reward is their own currency called SJCX which could then be converted to BitCoin’s. It is currently on the second round of testing before they release it as a beta to the public.

### MaidSafe

MaidSafe is a company based on Scotland, which is at the forefront of a decentralised Internet. They have many peer reviewed journal papers covering peer-to-peer networks, a distributed file system, authentication across a decentralized network, and also a new DHT algorithm. All the papers, patents and code are fully available for anyone to use. The code is under GPL licensee because they want no-one person, company or organisation to own the code and also want audit-ability. They are welcoming people to use their key components to create applications based on their SAFE (Secure Access for Everyone) network (Lambert & Bollen, 2014). It is currently in beta trials.

### Peer-to-peer investing, banking

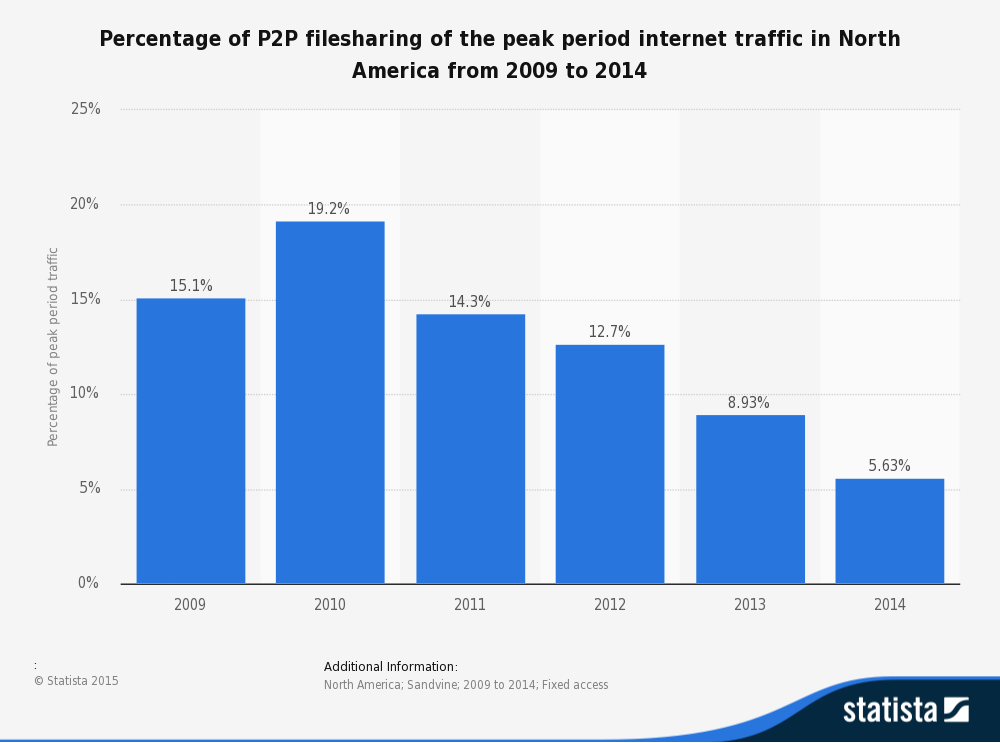
So far all the previous examples are relevant to the technological field. Peer-to-peer investing is another huge example, which is removing the large corporations. Banks have been through a troubling start to this century with the banking crisis and many scandals, which include wrongly selling payment protection insurance.

With savers losing out in an age of low-interest rates, a new form of lending money has emerged which is called peer-to-peer lending (The Economist, 2014). Zopa is one of the British platforms and which now has £663 million in loans. They cut out the banks and charge as little as 1% commission. Lenders can choose to take on the risk of bad credit themselves or pay for insurance to protect themselves from bad debts. One large difference is that there are no state-backed guarantee but this looks to change with the Financial Services Authority (FSA) moving in on the sector.

The above are all examples of peer-to-peer solutions, which work towards a goal of using the technology to join the users together for the greater good. The solutions are in all areas of life such as banking, privacy, security and communication.

Figure 1 shows the change in P2P landscape over the last five years. This could be put towards the legal challenges faced by torrent sites, which were a huge contributor to the traffic usage because the large amounts of data that are passed around. Now this has changed into corporations such as Netflix and YouTube having large proportions of network traffic.

Figure 1 - Percentage of P2P file sharing of Internet traffic in North America

(Sandvine, 2014)

## Motivation

The motivation for continuing this report is that the Internet needs to move forward as it has rather stagnated and has moved away from the basic tenets, which first created the World Wide Web. There are a lot of ideas floating around but these needs to be put into fruition.

# Research

## Research strategies

The research strategies will include both primary and secondary research.

Primary research is research, which is collected through the course of this thesis which will be original data. This research can be collected via interviews, observation, questionnaires and focus groups. A questionnaire will be issued to see if the points made in the introduction are valid. One of these was “Does anything need to be done?” in the context of privacy, security through the social network. This should be shown through the questionnaire. It will show if the issues are valid and that privacy is important to users of social networks.

Secondary research methods include looking at research, which has already been collected by someone else. Google Scholar and the UWE library which are great places to find already researched papers, which will be written by experts in their relevant fields.

## Questionnaire

The high-level information that will be gathered through the questionnaire, will act as primary research:

* A few questions of usage of the social media sites? To gauge the user’s use of social media, this may inform the implementation later on down the line.
* Would users use these sites if they knew how the owners talk about their own privacy and the user’s privacy? Use quotes from the background section to see if users opinion changes on if they would leave.
* Do users know about how the social media sites generally take over copyright of what they upload?
* The social issues regarding social networks and how angst can be develop through the use.
* To inform the requirements, if the user would be happy to use a native program instead of a browser to access the social network?

The piece of software used to publish the survey was Qualtrics. The reasons are that questionnaire looked very professional and how it was well regarded as the standard application used for university research. The questionnaire link was messaged to people online on Rhys Laval’s Facebook. Trying to get a wide selection of people, targeted was people who were not just well versed in technology. The majority were aged 19-25 though but there are a few entries of age.

Figure 2 – The questionnaire software through a browser.

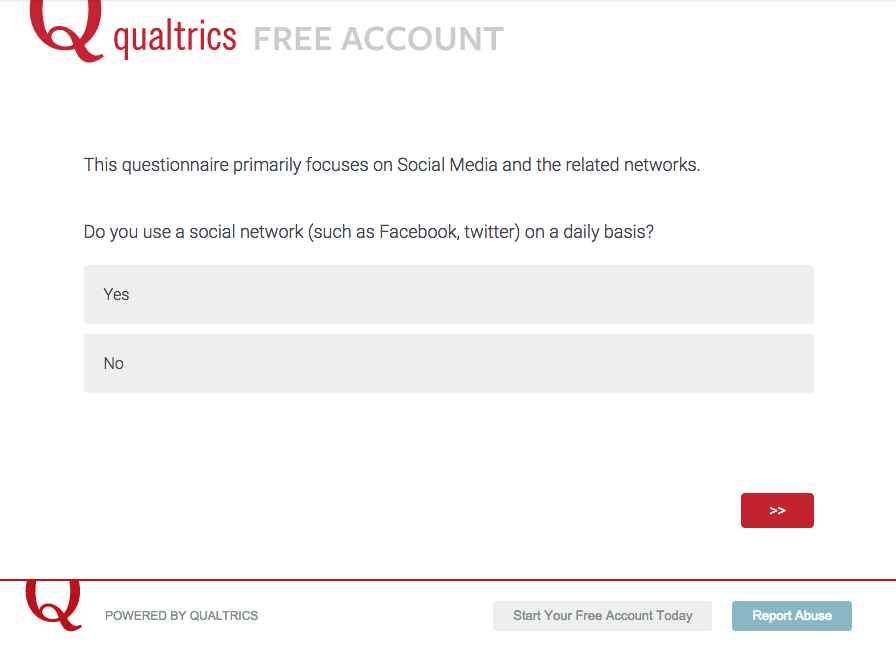
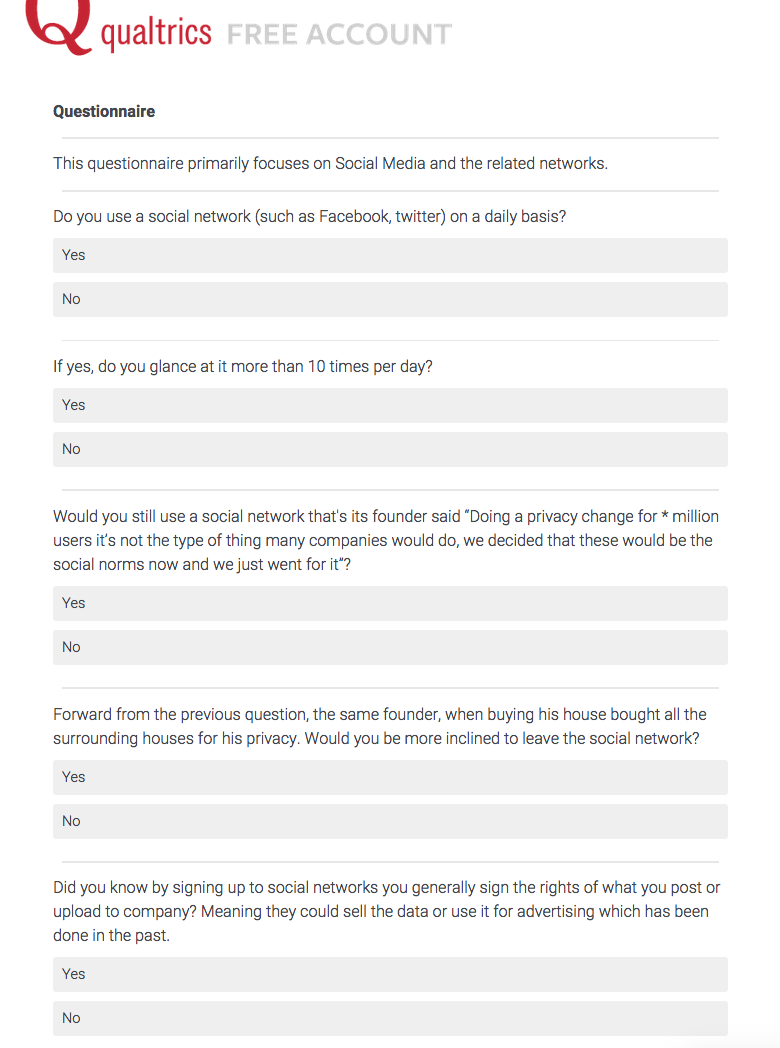
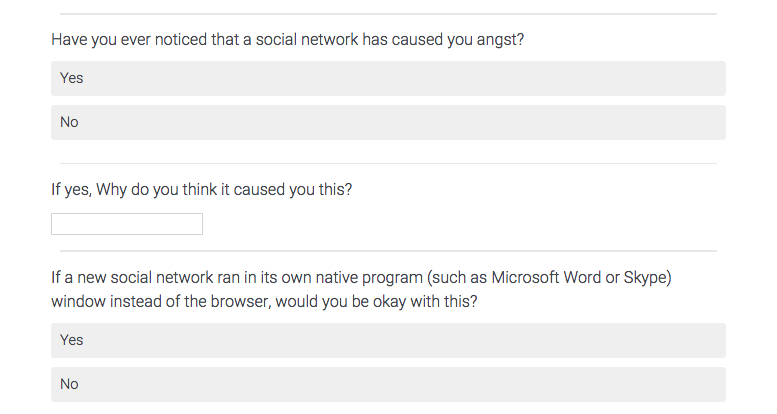


Figure 3 – The full questionnaire





## Survey Results

#### Do you use a social network (such as Facebook, Twitter) on a daily basis?

Table 2 - Results for Question 1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| # | Answer | |  |  | | --- | --- | |  |  | | Response | % |
| 1 | Yes | |  |  | | --- | --- | |  |  | | 23 | 88% |
| 2 | No | |  |  | | --- | --- | |  |  | | 3 | 12% |
|  | Total |  | 26 | 100% |

#### If yes, do you glance at it more than 10 times per day?

Table 3 - Results for Question 2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| # | Answer | |  |  | | --- | --- | |  |  | | Response | % |
| 1 | Yes | |  |  | | --- | --- | |  |  | | 14 | 61% |
| 2 | No | |  |  | | --- | --- | |  |  | | 9 | 39% |
|  | Total |  | 23 | 100% |

The above two questions show the high usage of social networks. Out of 26 people 14 of them access the site more than 10 times a day. This shows that for a new social network to gain mass usage it needs to work on mobiles and tablets so users are able to access the site anywhere such as on the move, in a waiting room or in the pub.

#### Would you still use a social network that's its founder said “Doing a privacy change for \* million users it’s not the type of thing many companies would do, we decided that these would be the social norms now and we just went for it”?

Table 4 - Results for Question 3

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| # | Answer | |  |  | | --- | --- | |  |  | | Response | % |
| 1 | Yes | |  |  | | --- | --- | |  |  | | 15 | 58% |
| 2 | No | |  |  | | --- | --- | |  |  | | 10 | 42% |
|  | Total |  | 26 | 100% |

#### Forward from the previous question, the same founder, when buying his house bought all the surrounding houses for his privacy. Would you be more inclined to leave the social network?

Table 5 - Results for Question 4

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| # | Answer | |  |  | | --- | --- | |  |  | | Response | % |
| 1 | Yes | |  |  | | --- | --- | |  |  | | 16 | 68% |
| 2 | No | |  |  | | --- | --- | |  |  | | 10 | 38% |
|  | Total |  | 26 | 100% |

These question show that a large proportion of people are willing to use a social network when the founder has a contradicting view of his own privacy to others. On the other hand it shows that one third of users use a network which they may not want to use in future due to the statements expressed by the founders.

Afterwards, from the feedback of the survey, it was found that the two statements were quite well known within groups of people who take an interest in technology.

#### Did you know by signing up to social networks you generally sign the rights of what you post or upload to company? Meaning they could sell the data or use it for advertising which has been done in the past.

Table 6 - Results for Question 5

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| # | Answer | |  |  | | --- | --- | |  |  | | Response | % |
| 1 | Yes | |  |  | | --- | --- | |  |  | | 24 | 92% |
| 2 | No | |  |  | | --- | --- | |  |  | | 2 | 8% |
|  | Total |  | 26 | 100% |

This question shows that the users are aware that their data is given over to the company. It is good that this is known. To further to carry on the research it would be interesting to see if users were concerned.

#### Have you ever noticed that a social network has caused you angst?

Table 7 - Results for Question 6

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| # | Answer | |  |  | | --- | --- | |  |  | | Response | % |
| 1 | Yes | |  |  | | --- | --- | |  |  | | 15 | 58% |
| 2 | No | |  |  | | --- | --- | |  |  | | 11 | 42% |
|  | Total |  | 26 | 100% |

#### If yes, why do you think it caused you this?

Table 8 - Results for Question 7

|  |
| --- |
| Responses |
| People are irritating |
| Getting to much in other people’s business |
| Reading (political) news, seeing people post negative/hate-fuelled things when they don't know all the facts or are just hating on something different rather than showing compassion |
| Game invites |
| Everyone's highlight reel vs. my life |
| Overload of "input", too much to keep a handle on |
| Because bored |
| Social pressures and obligations. |
| I assume it would be able to have better features |

The above two questions look into the social aspect of social networks. In the background section of my research it was mentioned that social media and networks cause angst among members. The survey results fully support this. Nearly 60% is a very high number of users that have received angst from using a social network.

#### If a new social network ran in its own native program (such as Microsoft Word or Skype) window instead of the browser, would you be okay with this?

Table 9 - Results for Question 8

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| # | Answer | |  |  | | --- | --- | |  |  | | Response | % |
| 1 | Yes | |  |  | | --- | --- | |  |  | | 13 | 50% |
| 2 | No | |  |  | | --- | --- | |  |  | | 13 | 50% |
|  | Total |  | 26 | 100% |

There was an even split for a question that was meant to lead the requirements in terms of how the users would interface with the program. As 50% of the users would prefer it to stay in the browser it should remain the browser.

## Peer to peer (P2P) methodologies

This chapter covers all aspects of how peer-to-peer communication works with no central architecture.

### Bootstrapping

Bootstrapping by definition is starting a process by itself without any form of help from the outside. The Internet is designed so that it only allows unicast communication. On a local area network (LAN) you can use multicast to detect other devices. This allows easy detection of other users in a LAN environment (V, 2015). On the Internet this multicast system is not allowed.

All peer-to-peer solutions require one IP address and port to start the bootstrapping of the node. The BitTorrent protocol has many ways to start this. One-way is that in the Torrent file is a list of trackers which will then give out a list of IP’s in the swarm. UTorrent is a commonly used torrent program has its own tracker which will connect to get a list of IP’s in the swarm. Once it has found an online node, that node will then inform the sender of other nodes locally.

Unfortunately, this is one of the flaws in how the system works. Once the first time it has been started and has learned of nodes to connect to. These can be used in future. With time though these nodes will disconnect.

### Distributed Hash Table

DHT is otherwise known as a Distributed hash table. DHT store is a simple key-value decentralised database. Any computer (known as nodes) can update a piece of data or retrieve the value from passing a key. The keys must unique.

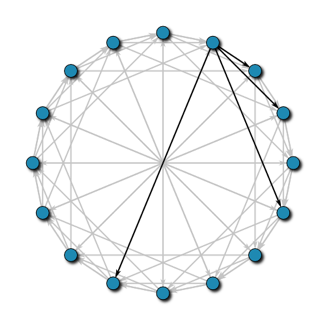
Many of the DHT overlay networks following this chapter incorporate algorithms, which are needed, as there is no central coordination. Whilst keeping the data evenly spread across the network with the idea of having high reliability, efficiency and scalability. There are many issues behind this, which the following algorithms try to solve. For example, when a node leaves or joins, how do you redistribute the keys evenly? Another is that node’s should be loaded equally in terms of bandwidth and storage. The idea is to split the keys evenly across the nodes with say for example *k* nodes having each key for reliability. An example of an overlay network is if the network had 3 nodes and 15 keys numbered 1-15. Node 1 would contain all keys with an ID of 3n; node 2 had ID’s with 3n+1 and etc. This is a very simple example and does not scale very well. This shows another issue that exists. As is it possible to have nodes join or leave without rehashing each key to its new node. This example outlines the common approach taken by the algorithms and they build on top of this. Can you put a node in the middle of two nodes and just have it communicate with just two local nodes to share the load? This problem has given the solution of content-based routing to find the right node to get the data.

Interestingly, even Amazon uses a DHT as an internal, highly available key-value storage system (DeCandia, et al., 2007). It is not publically available but does power some of Amazon’s Web services such as S3. The research paper behind it mentions how customers should be able to add/remove items from the shopping cart even when:

“Disks are failing, network routes are flapping, or data centres are being destroyed by tornados”. (DeCandia, et al., 2007)

### Chord

Chord is a protocol and algorithm for DHT. It was created in 2001 by MIT (Stoica, Morris, Karger, Kaashoek, & Balakrishnan, 2001). Chord normally works hand in hand with a storage mechanism such as DHash to keep the data spread over the nodes. A code example is located at [[2]](#footnote-2). Generally the file hash is then kept in the finger table for that node. Chord was one of the first to solve the problem of spreading the data across the network evenly and in an efficient manner. This is done via a variant of constant hashing which constantly assigns the same amount of keys across all nodes. Before constant hashing was introduced this created difficulties because the node would need to carry new information on every node to know how to spread the load evenly. This would cause scalability issues. The difference with chord is that the routing table is distributed too but also that it communicates with other nodes to have an efficient hash function.

Figure 4 - 16-bit Chord Network with the node having ‘fingers’ with 1,2,4,8,16. 

(Wikipedia, 2010)

Chord uses subtraction as its distance function, which allows it to circle the network. On figure 4 it shows the fingers, which are the nodes known about.

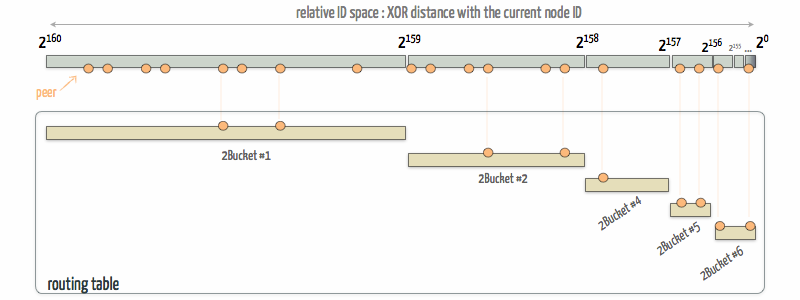
### Kademlia

Kademlia is one of the most well known algorithms, which came from three generations of ideas in the P2P research. It has the most used and is used by networks such as BitTorrent, KAD, and Gnutella. The word ‘Kademlia’ comes from the Turkish word for ‘lucky man’ and is also a name of the mountain peak in Bulgaria (Maymounkov, 2013). Kademlia holds the data and also stores the data at more than one node keeping the data more accessible even at times when large number of nodes is unavailable to the network. The main reason Kademlia works so well is because of its flexible routing table, which reduces the amount of administration communication, which is needed. This is due to the symmetrical design of the network.

The routing table is a constrained tree structure but as the network can be highly unbalanced the underlying data structure used needs to be able to cope. It contains *k*-buckets as its leaves, each which normally contain up to k contacts. Each bucket is allocated a range of the 160-bit space. A *k*-Contact is the data structure which holds the

* node ID
* IP:Port
* node state
* The last contact time.

Figure 5 – The routing table with example buckets.

 (Lachèze & Guilleminot, 2012)

On a node join, Kademlia’s flexible routing table works by replicating the keys of the local data pairs which are passed on to the new node. The data pairs are generally also re-enforced by re-publishing the keys at a set interval. There are many optimizations to this, which are outlined in the Kademlia paper.

XOR is a key component as this is used as the distance function.  
The distance function is:

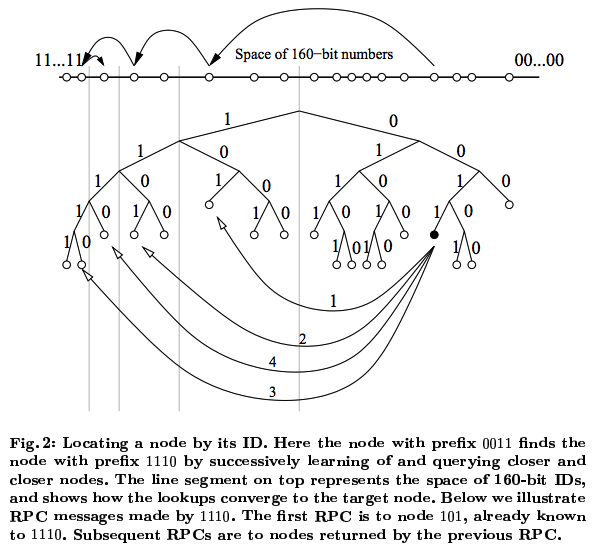
distance(A, B) = A XOR B

or

A XOR B = B XOR A

The above shows the mathematical formula which is a key part to the asymmetrical design.

Figure 6 - From the first Kademlia paper outlining the tree and locating a node



(Maymounkov, 2013)

The paper also outlines a few ideas for basic securities, which can be implemented to help with node, key and look up attacks.

Kademlia has 4 operations:

* PING – to see if a node is online and return local node information.
* STORE – keeps a key located with information of the file.
* FIND\_NODE – will either return information of the node or details of the closest node to the destination.
* FIND\_VALUE – Either returns the value if known by the local node, or details of the closest node known to know the value which in turn could just pass the command to another node.

The find operations are generally asynchronous and are performed *a* times in parallel to ensure speed. This allows stale nodes to be bypassed quickly.

There are many newer research papers which look into extending Kademlia to improve the base algorithm or the covering network:

* Tonika – By the original researcher of Kademlia, which addresses issues with long-range routing to shorten routes so that it always tries to use the shortest route available (Kelner & Maymounkov, 2009).
* Kaleidoscope – Extends by Kademlia by splitting the network by having nodes attached to a colour and each colour network knowing of only entry points to other coloured nodes (Einziger, Friedman, & Kibbar, 2009). It tries to solve the ‘hot spot’ issue of the protocol. The issue is where a piece of content becomes very popular which then creates a ‘hot spot’ around the nodes which have the piece of data. It tries to overcome this by caching content at other nodes.

Overall, Kademlia works very well because of Symmetry, uniqueness and unidirectional. It was the first algorithm that directly looked at all the important functionality that is consistency, performance and concurrency to ensure speed and reduces failures.

### Tapestry

Very similar to another network called Pastry which was developed at the Microsoft research facilities. Pastry has many of its idea’s from the Plaxton network. Tapestry has been used by a few large scale networks such as OceanStore. OceanStore was created to be a long-term archive facility. Tapestry uses a neighbour map data collection which is a layered store containing neighbours details put into levels. The closer the node is the higher the neighbour’s level. The locating of nodes/keys is similar to the principles Kademlia and it also has a flexible routing table.

### Issues

The Sybil attack is one of the most common attacks affecting peer-to-peer networks. It is not limited to just peer-to-peer networks though; it has been seen in mobile networks and in real life vehicle networks. It’s about forging the identity. In the Kademlia network you would forge the identity of node. This would require a highly customized client but it can easily be done causing large-scale attacks. It is harder when there is a trust chain such as PKI (explained later in this section). In a decentralised fashion this can be done with a Blockchain. Otherwise it can be complex for a fully decentralised system to prevent this. There are algorithms, which are used to detect honest users, which then pass on the results to other nodes, which in turn can push these incorrect nodes out. Unfortunately, these tend to push the valid node out which has the same identity as the fake node (Danezis, 2014).

The Eclipse attack otherwise known as Torrent poisoning is to intentionally put fake data out which causes issues for the routing tables of torrents which then is spread onwards. This is a common attack used by anti-privacy organisations. First they will generally collect IP’s which can be used as evidence later, after they will use many methods of poisoning the torrent to make it harder to download the files. These include decoy insertion, index poisoning, spoofing and eclipse attack. The eclipse attack is when they try and take over a peers routing table by faking the identities of all the peers in the table.

Distributed Denial-of-service attacks are common and have been used against websites to shut them down for hours at a time which are harder to stop as the peers can be worldwide. The peers are tricked into requesting the data from a peer which is the intended target of the DDoS. Torrents of 100’000 peers can easily be used against targets.

Bandwidth utilisation can be an issue as a node which becomes popular can cause massive bandwidth usage. Many ISPs have caps on usage but with time these will be eradicated and the speed will be increased. This can also be an issue on local networks though. For example Facebook started at one college in the US. If the same happened over here at UWE, would UWE’s network be ready to cope with the extra demand, which is not a genuine academic requirement. This issue can also be caused by rapid joining and leaving of nodes, which causes massive network usage.

Ideas to think about how to combat the above are:

* The querying node should be able to see what’s happening behind the look-up process.
* Be able to verify the node/file ID’s.
* Go against the single responsibility principle of OOP coding and have more than one point of responsibility.
* Continuously cross-check the nodes with random queries (Druschel, Kaashoek, & Rowstron, 2002)

### Conclusion

Kademlia has the upmost support and is the most used but it does not solve all the problems mentioned in the DHT chapter. Attacks are becoming more common such as the Sybil attack and the anti-privacy agencies have used this attack to remove illegally pirated torrents. More research is needed into the area to find the optimal solution but this is happening as decentralised technologies are on the rise. Hopefully soon new technologies will lead the Internet into the direction that the introduction talks about. The search for a DHT is not complete. As no solution has been found when the networks grow to millions of users. There is many ideas’ from keyword tree’s to using semantic webs. Unfortunately most require the use of broadcasting a message to a large number of nodes, which causes massive network issues at large scales.

## Social aspects

Social media and their related sites may seem harmless but as more in depth research is done more harm is found.

Social media research so far has led to evidence of loneliness, depression, anxiety, addiction and envy. These were already linked to problematic Internet use but this has increased with the high use of social media sites (Daria J. Kuss, 2013). In the following chapter, we will cover them and look at the associated research.

### Depression

The phrase ‘Facebook Depression’ has been coined by the news but there is no conclusive evidence found. A paper looking into the impact of social media in pre-teenagers linked large amounts of time spent looking at Facebook to symptoms similar to depression. It mentioned that acceptance online as in the real world is an important element of adolescent life (Gwenn Schurgin O'Keeffe, 2011). On flip side, Tumblr has shown to help users with depression by using it as a coping tool, letting out their personal problems into a space unknown to their friends (Anstiss, 2012). This leads to connectedness and reassurance that it can get better. Society as a whole is defamatory about depression as it seems as a weakness and because of this people generally hide it. Tumblr in the last few years has had controversy with romanticising the issues within this chapter. It has added many features to overcome these problems. Microsoft have done research into using Twitter to detect if an individual is depressed or may be in the future (Munmun De Choudhury, 2013).

### Addiction

Dopamine has been shown to be released when for example a user ‘likes’ a picture on Facebook or re-tweets a tweet. This shows that this can have an addiction potential similar to a drug such as Heroin. An example of a user who has an addiction is a 24 year old woman who the psychiatric clinic of the University of Athens School of medicine in Greece, gave the diagnosis of social media addiction. She had lost her job due to it and her symptoms were mild anxiety, sleep disturbance and loss of hobbies (Financial Times, 2013). This is one example which has been put forward for debate to see if Social Media Addiction should be included in the Diagnostic and Statistical Manual of Mental Disorders. It has even caused Thailand’s Department of Mental Health to warm over a selfie addiction on how it can cause loss of self-confidence and that it can cause a negative impact on life and work (Bangkok Post, 2014).

A scale called the Bergen Facebook Addiction Scale (BFAS) has been created by researchers in Norway (Andreassen C. S., 2012).

“It occurs more regularly among younger than older users. We have also found that people who are anxious and socially insecure use Facebook more than those with lower scores on those traits, probably because those who are anxious find it easier to communicate via social media than face-to-face

”

**These key facts were taken after the survey was used on 450 students.** (Andreassen, 2012)

The Bergen Facebook Addiction Scale is based on six basic criteria shown below. Using a scale of 1-5 with 1 being very rarely to 5 being very often

* You spend a lot of time thinking about Facebook or plan use of Facebook.
* You feel an urge to use Facebook more and more.
* You use Facebook in order to forget about personal problems.
* You have tried to cut down on the use of Facebook without success.
* You become restless or troubled if you are prohibited from using Facebook.
* You use Facebook so much that it has had a negative impact on your job/studies.

If scoring above 4, more than 4 times, may suggest that you are addicted to Facebook but this also can apply to any social network. They also created a Work Addiction scale.

A study of 1000 users in Sweden found that 85% of users said Facebook was part of their daily routine. Young users, 67% use it kill time. The most important fact from the study, which shows the addictive potential, is that 25% of users if they cannot login feel ‘ill at ease’ (Gothenburg Research Institute, 2012).

One of the statistics is “Women who use Facebook more are also reported to feel less happy and less content with their lives”. This demonstrates, similar to fashion magazines, have long been letting readers compare themselves to the magazine models. It has a similar affect but along with friends instead of celebrities or models.

### Envy

It is common for people to overestimate other people happiness but Facebook makes this easier. It only shows what users want it to show. They create an online image of themselves, which is only the better side of them. Other users comparing themselves compare themselves to this digital image of the person.

The theory is known as FOMO, which means Fear Of Missing Out. This is caused by the user sitting at home when their friends or others are holiday enjoying them on a golden beach for example.

Another human need is social interaction. It looks that Facebook should increase social interaction but it has been shown there is some negative psychological consequences (Kross E, 2013).

### Other issues

For university students it has shown that the level of use directly affects university grades. Although, it has shown to help social interaction in the first year of university. Later on Facebook encourages alcohol and substance abuse as more than 58% pictures of male users have pictures of these on their profile (Glassman, 2012).

Social overload can be an issue as social networking sites are bringing more frequent social interaction to the users. This in turn can show an increase in emotional exhaustion (Maier, 2012).

A breakup of partners can be painful. With our digital life online it can cause more issues as a third of us would find it hard to delete those digital memories. Keeping these stored memories make it harder to for us to move on.

### Conclusion

Facebook has been around for a century so far and no long term studies have been done yet on a user’s wellbeing. All of the above mention about over use of the social networks so it seems like the quote "All things in moderation" would be applicable. Should the networks themselves detect the time spent on the network to produce a warning? Similar to World of Warcraft which sends a message to the user after a specific period of playing. The other side of all of this is that people who struggle with real-world interaction it has been shown that it can be good to get some interaction through social media as it can help.

## Privacy

“A state in which one is not observed or disturbed by other people.”

(Dictionary.com Unabridged)

Privacy is now more important than ever with the countless leaks over the last few years. People are thinking of their own privacy more and more. The UK is known as the surveillance state of the world. We know little of how much we are really under surveillance? The Edward Snowden leaks show that we do not really know how much the Government collects and/or stores. Statistics collected by YouGov for Liberty, which shows that 77% of people believe that the UK has become a surveillance state (Liberty, 2009). This was taken before the Edward Snowden leaks and as a consequence this could even be higher today.

Privacy is a key component with Liberty in the human rights, which were recognised internationally in 1950s. CCTV in the UK is at a level of 1 camera per 11 people. Evidence has shown it works well for a period when it is first installed but the passage of time it becomes less effective (Armitage, 2002).

Government Control is on the rise with more than 10 countries having blocked Facebook for more than a month. These are blocked because of the scares of political activism or that the country is under totalitarian/communist control. It is understandable that Cyber warfare could the next big issue confronting Governments. Even the USA has looked into the president having the power to turn off the Internet when required. In a state of war or a civil disorder, for example. The government policies to shutdown access to sites have been shown not to work. The easy access of virtual private networks show this even with new technologies like deep packet inspection (Anderson, 2012).

“Mass surveillance has the potential to erode privacy. As privacy is an essential prerequisite to the exercise of individual freedom, its erosion weakens the constitutional foundations on which democracy and good governance have traditionally been based in this Country.” (House of Lords, 2009)

The increasing responsibility of these single providers could ultimately be closed down at moment’s notice by governments or even external influences.

Censorship is also placed on Facebook as it receives thousands of requests each year from Governments’. Data requests are frequent and Facebook has to spend time to ensure that requests meet countries regulations. Governments’ can also enforce the content to be restricted from view in their own country. An example of this is India, which many of these items were links or text relating to “criticizing the government”.

Table 10– Shows amount of government requests towards Facebook

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Country | Requests for User Data | User Accounts Referenced | Percentage of requests where some data produced | Content Restrictions |
| United States | 15,433 | 23,667 | 80.15% | 0 |
| India | 4,559 | 5,958 | 50.87% | 4960 |
| Germany | 2,537 | 3,078 | 33.94% | 34 |
| France | 2,249 | 2,599 | 30.24% | 22 |
| United Kingdom | 2,110 | 2,619 | 71.68% | 9 |
| Italy | 1,869 | 2,658 | 49.28% | 3 |
| Brazil | 1,307 | 2,269 | 35.20% | 0 |
| Australia | 610 | 650 | 60.33% | 7 |
| Spain | 514 | 860 | 36.58% | 0 |
| Portugal | 354 | 403 | 40.40% | 0 |

For the period between January – July 2014. (Facebook, 2014)

Identity theft has been shown to be on the increase since users are putting more and more information online. These sites allow anyone to easily make a profile in the name of anyone without any issue or check. Anyone can easily de-flame a user in an effortless manor. One of the key issues users create is that they are using the same password across services such as email, social networks and general sites. Once an attacker has a password they can use information across services to get enough details to use this to create bank accounts, spend money and ruin a person’s credit history. This is made worse by only 18% of people knowing that data they spread on social media can be used to cause identity theft (Action Fraud, 2011).

Social media can now be used as part of the recruitment process. A survey of 400 HR managers it was found that 45% use social media as a screening process of viewing applicants publically available data, which is only be a few clicks away (Andrea Broughton, 2013). Facebook and other social media sites have been shown to work well as recruitment method to. A quick Google search shows the large amount of people who have been dismissed because of a post on social media.

Data mining can be used for many reasons, from targeted advertising to selling user information. Facebook Beacon was the first version of this, which allowed websites to pass information along to Facebook without the user’s option. The information such as buying items, which would then be shown to other users on Facebook. This has now been removed in favour of Facebook Connect, which is an opt-in service. Facebook Connect service includes a Share button on many sites, which allows the user too easily to share pages they have visited. The share button placed on the page includes the cookie, which includes the data of the user so Facebook over time learns of where they have visited.

### Conclusion

From the primary research, it was learned that users do understand that by signing up to social media sites they generally sign over their data to the company. This is changing with companies like Ello making the change clear to the users. Surveillance is an issue which users are aware that the governments are trying to have more and more control.

## Security and Web of Trust

Most current public key infrastructures rely on a central service to work but the Web of Trust works around this. The hierarchical public key system works well when users can connect to a central server.

### Pretty Good Privacy (PGP)

Hans Zimmerman created PGP in 1991 and it has lasted the test of time. It is still use to today to send secure emails and to authenticate messages with digital signatures. It works with the use of two keys, one, which is public, and the other one, which is private. The public key can be used to encrypt data which then the private key is used to unencrypt the data. PGP is made faster by the use of an algorithm to make the data shorter first as the encryption algorithms can be very time consuming. It can also be used to authenticate messages via the creation of a digital signature which can be checked via the usage of the public key.

PGP can use the proprietary RSA or the DH algorithm for encrypting data. Diffie Hellman created the DH in 1976.

### Web of Trust

The Web of Trust is a concept to validate the user’s public key of the user. Public Key Infrastructure has a central server, which hands out these keys and these keys can be checked against the higher level keys. It is the concept used in PGP to have the private/public key association.

### Blockchain

To have a secure login/registration system you would need another layer on top of the DHT. Twister, MaidSafe and Storj.io work on using the Blockchain as this extra layer. It works by using the Blockchain as a proof-of-work system to keep hashing the data with the already hashed previous data. Each block would contain the details of a user or say if a user’s details were changed a new block would be placed in the chain. The block would contain the user’s details such as username, password, relevant Id and the hash of the public key so it can be checked against when it is being used.

The features, which are needed are: Login, Register and Check Key. For my project and the time constraints this is not possible as it would be another layer of complexity.

## Open Source Licensees

Open source licensees must comply with the open source definition:

“Denoting software for which the original source code is made freely available and may be redistributed and modified.”

(Dictionary.com Unabridged)

The three key licenses used are MIT, Apache and GPL. It is common for an open source project source code to have a file called ‘LICENCEE’ which contains which licence they have chosen.

The MIT license is the simplest license, which is also very permissive. Anyone can take the code under the license but they must credit the author. Importantly they cannot hold anyone liable. It permits all commercial use, distribution and private use. It was published by MIT (Massachusetts Institute of Technology) in 1988. It is commonly used and it is used by Mono, Ruby on Rails, Node.js and jQuery. OpenBazaar, one of the projects mentioned in the background section, uses the MIT licence.

Apache license is a very similar to the above but much longer and complex. It includes that when you put code under the licence you are granting the worldwide, no-charge use of a patent if held. It forbids the use of any trademarks. The license is authored by the Apache Software foundation and was last updated in 2004. All Apache’s software is licensed by this and also the Android operating system.

The GPL v3 license is the most Copy Left of them all. Copy Left is form of licensing which means that the work or software may be used and modified freely but that anything created from it has to be put under the same licensee. Examples of who use it are Linux, Git and WordPress. It is said to be the most widely used licensee (Wheeler, 2014). The licensees are released by the Free Software Foundation and were first written by Richard Stallman who is commonly known for launching the GNU project. Diaspora is an example of one of peer-to-peer projects using GPL v3.

This is more than just being open-source the aim is that these licenses bring the project further because these licensees lead to more people pushing their changes into the wild. The social side is also increased as people work towards as a collective towards a greater good.

# Requirements & Analysis

Below I’ve used the standard IEEE Software Requirements Specification Template[[3]](#footnote-3), slightly modified to exclude parts that have been covered previously in this document.

## Introduction & Overall Description

### Purpose & Product Scope

The purpose of the product is to decentralise the current architecture of a social network so that it can be free, open source and uncensored. There are many relevant benefits to different users. One of the fundamental reasons is that users data should not be used to make profit. The scope of the product is a proof of concept that it could show that it is possible. As the introduction talks about, bringing back the Internet into peoples hands and not controlled by large corporations but the people it self.

### Product Functions

A high level summary of the product functionality is:

* Keep data in decentralised fashion across the user base.
* Allow users to have a profile, which is explorable by other users.
* Users can post on their own profiles.
* The posts in future should be expandable from the default status module.

### User Classes and Characteristics

There is only one user class and this is the typical user of a social network. They should have access/control over their own data and should know that the data will be spread over the network. These users may not come from a computing background. It should be simple to use and to install. It should be clear, easy and familiar and efficient.

To ensure that we meet that all users can use the proof of concept, the designs at each stage will be shown to a variety of people who meet the user classes and their feedback will be sought on usability and design.

### Operating Environment

The operating environment for the first version of the software to be built which will include all common desktop platforms From this the operating systems to be supported will be Windows back to XP, Mac OSX above 10 and the three most common Linux systems available. This will lead the choice of the programming language, as choosing a programming language that compiles on three systems will work best. The language will also need to network functionality that can be easily used. As it is a peer-to-peer system, it must be able to co-exist with other peer-to-peer technologies.

Hardware will follow the operating systems basic requirements with the thought in mind of moving towards mobiles/tablets in the future but this is not a requirement.

Throughout the development will testing happen on different environments such as building for multiple environments every major commit of code.

### Design and Implementation Constraints

There are no constraints yet except the ones mentioned previous to this. Even though this is a proof of concept it should aim to be able to bypass censorship like other peer-to-peer networks currently do. A peer-to-peer interface will be used for the data sharing mechanism but this will be decided in the forth-coming work, which is an implementation constraint.

### Assumptions and Dependencies

At all times, the use of packages will be used if they are appropriate. At this moment, no packages have been chosen but these will be selected as we go through methodologies and implementation. ‘Do not re-invent the wheel’ is applicable here, especially as it is only a proof of concept the packages do not need to be perfect but they can be changed before a live release.

## External Interface Requirements

### User Interfaces

As this is a proof of concept there is no set user interface design yet. As a rule of thumb, ideas and designs should be used of other social networks to keep the usability high. A requirement is that it should be web-based, as all users of a computer will be comfortable using a web browser to interact with the network. There will have to be a program installed locally on the users’ computer for the interaction with the peer-to-peer network. General user ability standards should be followed and general accessibility guidelines should be also followed.

## System Features

Below are the system features; these are the functional requirements of the project. They have been outlined with MoSCoW method as it is a useful methodology to reach a understanding of how important a feature is. It would be used so the stakeholders can also fully understand the importance.

* Must – The feature must be included for the software to be released.
* Should – It is high-priority but other possible solutions are available so it does not delay the release.
* Could – It is a desirable feature but not required in the release.
* Won’t – Stakeholders have agreed that in this release, the feature will not be included.

### Peer-to-peer DHT

**Description and Priority**

This is HIGH priority as without this, nothing else can be worked on and this a key part of the requirements. This is the backend storage of the system where the data is spread between the nodes connected. This will work as a frontend interface for whatever DHT we use in the background.

**Stimulus/Response Sequences**

Any frontend user interaction from the site will include getting or setting data from the DHT. This part of the project will be hidden from the users view but without it, the project would not work. A test client will be created to be able to test this item.

**Functional Requirements**

* REQ-1-1: Backend storage mechanism of the key/values
* REQ-1-2: Start joining the peer-to-peer network
* REQ-1-3: Put mechanism to save to the backend storage but also propagating the information on the P2P network.
* REQ-1-4: Get mechanism to get the information. If it is not found in the backend storage, needs to searched for on the P2P network.

### Social Network

**Description and Priority**

This is HIGH priority as without this the project would be a failure based on not creating a proof of concept social network using a P2P network as the backend mechanism. This will be a user interface that will communicate with the DHT to create a very basic social network with user profiles and the ability to post on your own profile.

**Stimulus/Response Sequences**

This will be a frontend mechanism of the social network. It will have a method of showing User Profiles and the users can post onto their own User Profiles.

**Functional Requirements**

* REQ-2-1: Backend storage mechanism of user profile
* REQ-2-2: Backend storage of the user profile data. For example, a text status.
* REQ-2-3: Creation of users profile
* REQ-2-4: View users profile by their ID
* REQ-2-5: View users status on user profile
* REQ-2-6: Post to own users profile

### Security

**Description and Priority**

This is LOW priority as to create the proof of concept this is not needed. For the social network to be used everyday it would need this functionality.

**Stimulus/Response Sequences**

This means that the data will need to be encrypted on the network. It will require a ‘friends’ functionality and then the encrypted data will be allowed to be unencrypted to the ‘friend’ but the users could select the data be free to view by everyone.

**Functional Requirements**

* REQ-3-1: User profiles to include encryption keys to be created on account creation
* REQ-3-2: Re-creation of keys to be allowed
* REQ-3-3: Adding/removing of friends and the transfer of keys between.
* REQ-3-4: Posting of a status has a selection of who to show the data to.
* REQ-3-5: Show the data or unencrypted data first if friends.

The Test Cases to cover the above features are located in the [Appendix](#_Appendix_–_Test) section at the end of the report. These software features will be tested in a variety of ways, which will be shown in the testing section of the report.

## Other Non-functional Requirements

### Performance Requirements

The backend DHT system needs to be able to perform with two seconds to keep the users attention. Two seconds is suitable for the proof of concept application but in the future this requirement will be lowered to one second. As even at one second it can still be a noticeable wait. The error rate should be below 0.01% of requests for the DHT system. As the DHT can sometimes take longer parallel calls will be a must but for the proof of concept this can be omitted. As the calls could take a couple of seconds, a loading icon should be shown. As soon as any amount of data is returned, it should be shown to the user. The proof of concept application should be able to scale to around 500 users without issues. Whilst building the network, consideration should be thought of that in the future it could be used by millions of users.

The frontend should be instantaneous on all calls. There should be no waiting within the local application.

### User Interface Requirements

Using Jesse James Garret research on elements of User Experience. This will outline the user interface designs also needed. Garret created this layered system because the web is not just about looking great; it’s also about usability. Usability is a key part to a social network and as the audience is wide it is part of the success.

The first two layers (shown in Figure 7) from the bottom have been well covered in the functional requirements which are Strategy and Scope.

The other’s are:

**Structure**

Interaction design covers how the user moves through the application and makes sense on how to do a task. So this will include how the user understands to go to another profile or move around the site. As this is the most basic item on the social network, this should be as easy of a click away if they are friend or a search box away from finding another user.

Information Architecture covers how the users make sense of the information, when the user moves through the site. The user profile is a key part of the site and the information should be easily read and that the newer status should be at the top of the page so keeping up to date is easy. The structure of the site is simple as there are only a few key pages but the data inside these pages changes.

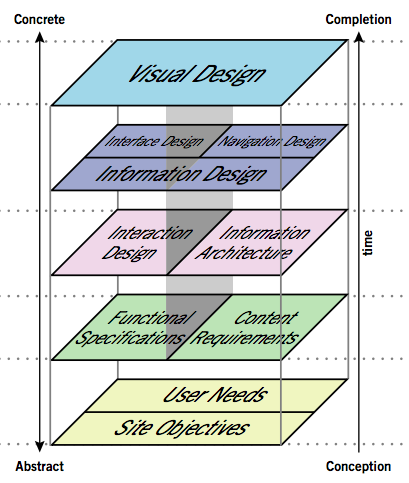
**Skeleton**

Information Design is how the information can be easy comprehended by the layout of the page. On an eBay auction page the first items to be seen are the image, title and price. For example, the profile pages will start with the users picture, name and then follow on to their time line. If it is the users own profile it will have a form so they can submit their own status.

Navigation Design covers the movement through the pages and movement through the information. The social site will have:

* The list of friends on the right so they are easily accessible.
* A navigation bar on top, which will include a search form to search for other users and a button for settings.

**Surface** is the last layer. It covers the actual Visual Design of the site. As this will act as a proof concept we have decided on using a template framework, which will allow quick prototyping of the pages.

Figure 7 – The elements of User Experience 

(Garrett, 2002)

Looking at the Mac OSX design basics and principles had given ideas on how the site should be kept as simple as possible and that the ‘mental model’ is a key part of the software principles (Apple, Not known). This mental model is good when the first ‘skill-based behaviour can not be used as it is new. So instead ‘rule-based behaviour’ can be used. These are key to Jens Rasmussen paper on ‘Skills, Rules, and Knowledge; Signals, Signs and Symbols, and Other Distinctions in Human Performance Models’ (Rasmussen, 1983). A good example of how a trash bin is used as rubbish bin to indict that it is a throwaway place. So in our application a paper plane could be used to indicate, send this off to everyone.

### Safety Requirements

This application will not be a critical application but action should be taken to make sure that data could not be lost or damaged.

### Security Requirements

Security is a system feature, which has a low priority.

### Software Quality Attributes

A tight testing feedback loop should guide the development methodologies used. The phrase ‘test early, test often’ is key. All the features added to the system will be tested either by user testing, unit testing or end-2-end testing. Traceability should be easily done, to aid this each requirement has been given an ID so it can be followed through to the testing.

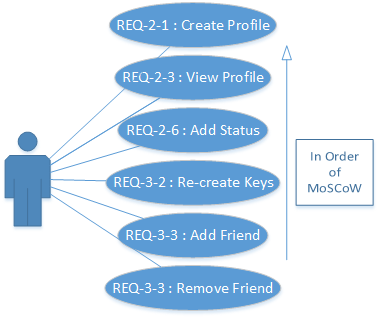
Test driven development will also be key for consistent code testing. On committing of software the tests must be ran by the committer. Code tests should be used to test code and that the code coverage should meet at 75% of the code being tested.

The DHT part of the project should be written in a separate package so it can be fully tested externally to the system. This should increase the reliability, reusability and robustness. This is advised because of how it could be used for other projects in future but also for testing purposes.

## Other Diagrams

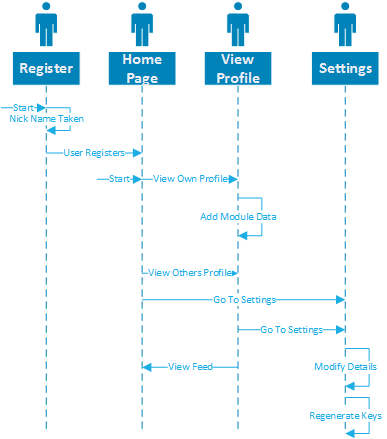
### Use Case Diagram

Figure 8 - Showing the use cases in the system prioritised by MoSCoW



### Sequence Diagram

Figure 9 - Sequence Diagram showing the movement between actors and page interaction.



This shows the interaction/movement between the pages and the interaction that could be made on each page.

### Communication Diagram

Figure 10 – Communication Diagram between the different parts

The above shows the communication between the levels of systems. The first being the remote nodes all communicating with each other to have the data spread around the world. The second shows the local application, which has the local DHT workings and the service to communicate with the browser.

# Methodology

An overview of why the following languages, tools and libraries were chosen. This has been located before the design section because, with knowing the languages or the libraries known it would make the design more specific on how it was to be built otherwise the overall design architecture would be very broad so it could be built in many languages.

## Mono C#

Mono C# was chosen because of its easy cross platform tools which are still a full object oriented language. It immediately gives cross platform-ability across Mac, Linux and Windows with the option to extend to Mobiles easily.

Mono C# was chosen over other languages because:

* Cross platform including Mobiles for the future.
* Language features that are not in Java - closures, operator overloading, partial classes, runtime generics, indexers, delegates and LINQ/Entity Framework.
* The massive amount of packages available via NuGet
* C# from Microsoft now being open source.
* The editors such as Visual Studio / Xamarin which are easy to use and highly configurable.
* The testing tools

Other languages that could have been chosen are:

* C and C++ because unparalleled performance and also the ubiquitous cross platform use. The issue is that separate builds would have to be created for each platform, which can be time consuming, and as a proof of concept this would take a lot of time.
* Java was considered because it is cross platform and could also work on Android. The issue was with the asynchronous nature of the DHT and the features mentioned in the requirements specification necessitated use of a performance critical, which the Java Virtual Machine (JVM) may not be able to give.
* Perl is also quite hard to install on Windows, which could scare people away.
* Microsoft’s .Net was not chosen because of the main limiting factor or it being only Windows, which is a major system requirement.

## Client/server side communication

Web sockets were used instead of AJAX and other similar tools because of the many of the calls will be asynchronous but will want to send parts back from the server at any time depending on how long the DHT takes to reply. It will want to send back the data to screen as soon as possible. The issue with going with web sockets it is a new technique and those old browsers will not be able to use this but as a proof of concept it will work well. If this were going to be released as a production system it would need to be able to fall back to AJAX. There are many WebSocket packages which work with Mono C# such as Fleck.

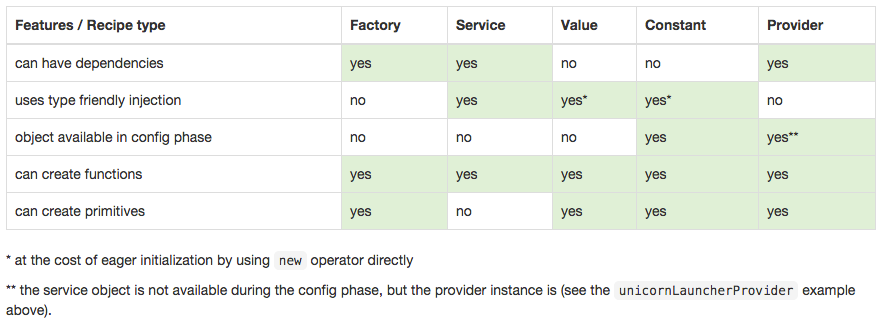
## Client Side JavaScript

Angular.JS is chosen because of how it works as a full framework for creating web applications in the browser with ease. It can easily work with WebSocket and update the data on screen with ease. It allows the MVC pattern in the browser with ease. The idea is to write less code but more declarative with the code in use. This allows us to bring the proof of concept forward easily and quickly. It also allows for unit testing and has with been built with this in mind.

Angular.JS has many types of items that can be used within it, these will be explained in the following paragraph and the design stage it will be chosen what will be suitable for each component.

* A **Controller** is a component, which can control the view to the user but also send commands to the other below components. It should not contain business logic as the below are where that work should be done. “translates from user-interface actions to application-level actions.” (Fowler, 2006)
* **Factories** are components that create new objects to be returned. It returns functions, which can contain many different objects.
* **Directives** are functions, which can be called from views, to for example change a JavaScript Date into a more readable date.
* Put simply a **Service** is an injectable constructor.
* **Filters**
* **Value** and **Constants** are two other places to store variables.
* **Modules** can include the all of the above to create modules, which can be shared such as any libraries that would we would require creating the application. These can also be used when the application becomes large so that the application can split into easier manageable chunks.

Figure 11 – From the Angular.JS Documentation, showing features available to each type of item.



(AngularJS, Google, 2015)

Dependency Injection can be used on the following: modules, factories, directives, services and filters. Templates can be included in the way of HTML files by using simple tags in the HTML which Angular will read and automatically fill in. This is similar to the way it automatically files in variables which must be located in the current scope by using {{var}} which will be replaced by whatever is in the $scope.var in the current scope such as the current controller. A root scope is provided at $rootScope which allows for objects to be accessed across controllers and the above items.

## Local Database

SQLite is chosen as the background research showed some of the local programs using it such as OpenBazaar. It has good integration with Entity Framework also which means easy use of the database through an ORM framework. At the implementation if there is time, the repository/unit of work pattern will be used as using a commit to the local database and DHT will be useful.

## Testing

The only testing platform that works with Mono C# is NUnit. This will be setup to run automatically with the build servers on commit for consistent testing across platforms.

Angular.JS is easily testable with Jasmine but as the frontend will be very simple to start with this would be a concern for the future. An End-2-End test would be useful for the future so the whole system would be tested but again this is a future concern.

## Peer 2 Peer Libraries

After the amount of time spent trying many different libraries shown and documented in Appendix Section 9.2 and unfortunately not getting anywhere showed that it would be quicker to make a basic proof of concept DHT library. It will be very basic and not include the Kademlia algorithm but will be designed with the creating a full DHT in future.

## Chat

Whilst looking into P2P libraries BitMessage[[4]](#footnote-4) was found which is a reliable chat platform, which works between users instead of a central server. As many social networks contain chat systems this could be additional use case for the future.

## Other Methodologies

After looking into KadOH, it was thought that it could work fully in the browser as browser extensions. Implementing the latest technologies such as:

* WebWorkers allow for full threaded application such as the Routing Table and to handle the multi-concurrent connections from multiple nodes.
* WebRTC which allows real time communication through the JavaScript API. So far implemented in Chrome, it allows UDP and TCP sockets. Demonstrations have included full video/audio chat which does not include a central server architecture.

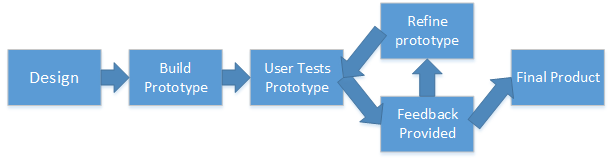
As the browser is the most common application on the computer, it would have that always on experience, even with Chrome now supporting extensions with running threads in the background when the browser is not even open.

## Development Methodology

There are many methodologies available to us, all with their own advantages and disadvantages.

Waterfall is great if the requirements are set out perfectly and will work great when the projects runs perfectly but unfortunately in the real world nothing will run like that. Agile is great for short iterations, cost and stakeholder information but also when the features are not clear yet. The V-Model has a very simple process but the previous phase must be completed before moving on, leaving no over lap between processes which is not great for prototyping.

Evolutionary Prototyping was chosen because it works great with unclear requirements, unfamiliar technology and is easy to manage. Its like a middle ground between Agile and Waterfall and allows rapid prototyping and rapid feedback in a small timeframe which is key to bringing a proof of concept product to get feedback from the stakeholders but also users. This with elements of the Draper Cycle to bring in more Human Computer interaction which is a key part of the user requirements as the user classes have a full coverage of the population.

Figure 12 – Showing how the Prototyping feedback model works 

Test-driven development is a process that about writing the unit test first and letting it fail before developing the actual code used. This requires small cycles of testing and the idea is to write the minimum amount of code to pass the test.

# Design

This chapter covers the design of the software, which is lead by the requirements and the methodologies.

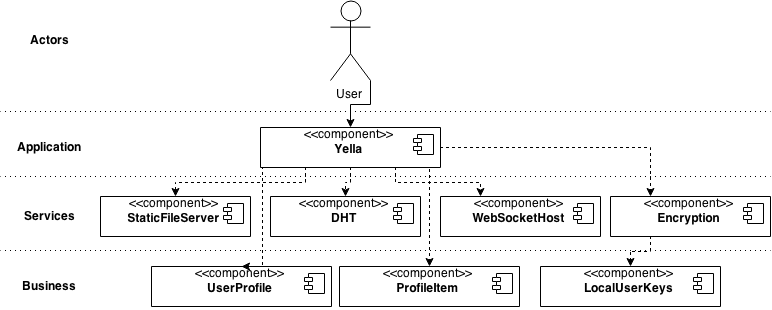
## Company Brand

The name BitBook was used at the beginning to name the project, in a similar fashion as BitCoin and BitTorrent.

Jeff Graham, during the ‘Project in Progress’ day introduced the name ‘Yella’. Linked to them was the idea that bringing the hand up to the mouth to indicate that your speaking to someone far away. This could also be a small icon to indicate that you are talking to everyone with the hand showing. The other icon would not include the hand to show privacy. This leads on from ‘Jens Rasmussen’ ideas of rule based behaviour.

## Component Design of the Backend

Figure 13 – Component Design of the Modules needed for the backend system to work.



Above is a top down component diagram used within the system. Each of the application or services components would be located within their own project in the C# Solution. The business types would be in a separate project. Even though the ‘StaticFileSever’ and the ‘WebsocketHost’ are very small components they have been separated out so they can be swapped out in the future. They will have interfaces exactly same as they are main classes. Also, they will implement the disposable pattern, as they both will use system resources which will be needed to be released on closure of the program.

## Component Design of the Angular.JS

Below in Figure 17 is the component design of the browser side JavaScript. It is more of ‘Model-View ViewModel’ than the standard Model View Controller (MVC) architecture. In the Methodologies section, the names of the Angular.JS functions were explained; these on the diagrams are shown by <<…>>.

The core differences between the MVC architecture and the MVVM pattern is:

* The controller is replaced by the View-Model, it controls the data binding and the actions between the data service layer called Model and the View.
* View-Model is more of a proxy between the View and the Model.
* View-Model does not do any computations except for controlling UI elements.

From the top, the views are split into two sections. The register view has its own design. The Yella layout will be the main design, once the user is logged in. It can inherit the partial views shown in the diagram. These are replaced easily using Ui-Router.

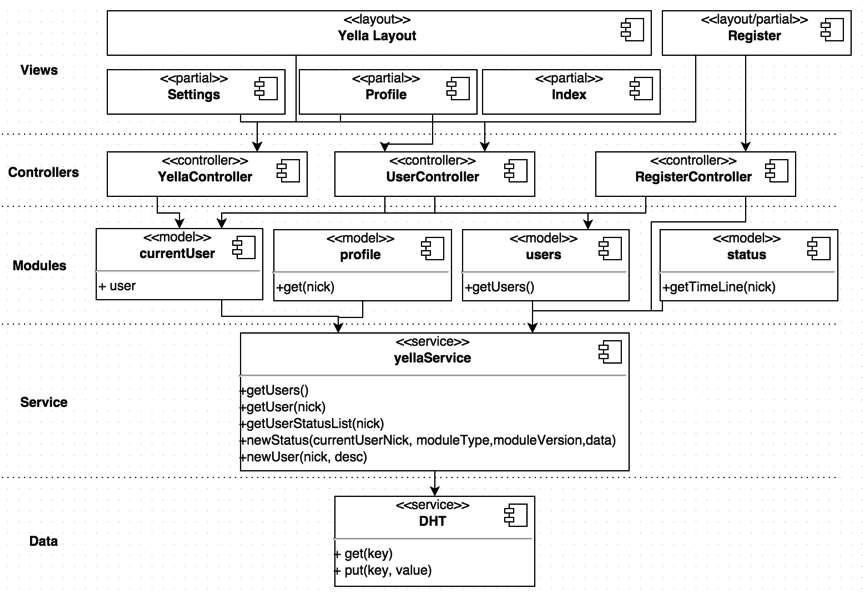
The relevant controllers include the code, which controls the interface. The Yella Controller has more control and it has control over the Yella Layout but can interact with any partial view below that.

Modules are the method of interacting between the Yella Service and the controllers. They will get the data from the service and apply any changes required to the data. An example is the ‘getTimeline’, which will require multiple calls to get each user and then each users status. The model will control the calls and once they have been made, they will pass the data back.

The service interacts with the DHT data layer. This layer handles any processing needed with the data received from the backend DHT.

The DHT will then communicate with the C# through the WebSocket connection.

Figure 14 - Component Diagram, which covers the Yella MVVM architecture of Angular.JS



Excluded from this diagram is the code which creates the modularity of the system. The module will communicate through another service which then communicate with the Yella service. This is for security purposes. The controllers, the module requires will be loaded via HTML as Angular.JS allows ‘ng-controller=”ExampleController”’ syntax which will be then loaded dynamically. This means each module will have two partial views; one for submission of the status and one for showing that status on a users profile or timeline.

## Class Design

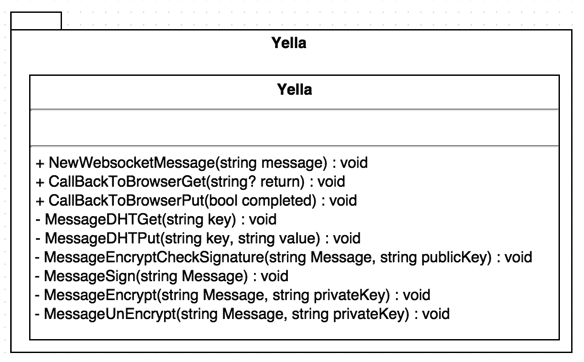
These are UML class diagrams for the modules in Figure 13. All properties start with lowercase and the methods are all ‘CamelCase’ with the first letter capitalised.

### Yella

Yella is the main program, which loads all the components needed. On start-up, for the proof of concept it will allow us to enter a port to start on. Whilst testing we can run multiple versions of the software on the same machine. It will also ask for the details of the bootstrapping node.

Once these details are known the software will automatically, start the other components fully, by calling the Start methods on the relevant components. This will start the DHT, StaticFileHost and the WebSocket components. The website will be accessible to the user from the browser. Inside of here it will control the messages between the browser and the DHT. This will be controlling the messages sent between them both and sending them between the components via the WebSocket. A lot of the messages used will contain no synchronous return as all the DHT work is done through Deferred Events and the encryption message call-backs will directly interface with the WebSocketHost.

Figure 15 – UML2 Class Diagram of the Yella Module



### DHT

The Dynamic Hash Table is a key part of the whole integration. A Persistent Hash Table will be used as storage. This will be separate so that in future it can be swapped out easily. For example, a different backend data store. For the proof of concept it will first use a dictionary. In the future, it would need to be persistent which could be written to disk so when the session restarts it would allow quicker start up times. This would also be best to keep the nodes known and save them to disk to simplify the bootstrapping process when it has been ran once.

When bootstrapping the node, the Start Method on the DHT class allows one ‘IPEndPoint’ to be given. That first endpoint will then contacted which should then return k Nodes. ‘IPEndPoint’ is a standard C# class[[5]](#footnote-5) which contains all the information required to contact a node.

The Routing Table is used to keep the nodes known and up to date within the correct Buckets. It contains the list of Buckets and the functionality to add and get Buckets.

Each Bucket contains a list of Nodes. One of these is the start and end points of the area of Node’s it covers. When the capacity has reached it’s maximum, the split command is called to split the bucket into two.

Nodes are kept in the Buckets. New nodes are added automatically on the first new message to the Node.

The transport class is used for communication; it contains a listener for messages on the port specified when being constructed. The listener has a public event handler which the main class which will be subscribed too. The messages will be passed to MessageRecieved method. This method will then decrypt the message and pass it onto the relevant method which handles the message. The handle methods start with text handle and end with the message type. On the Transport class there is a method to send messages.

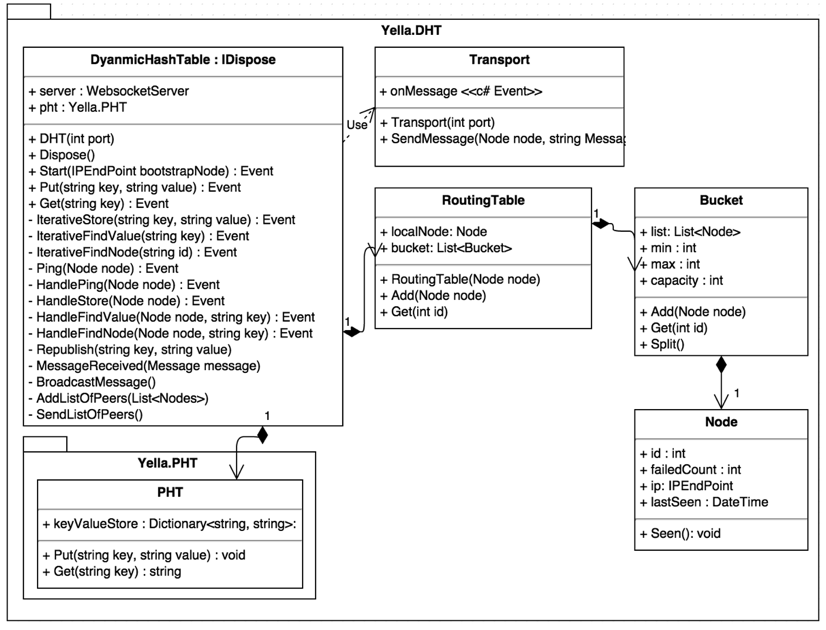
The DHT class has the methods Get and Put which will be required by the Yella main class as they will be the methods called over the connection from the browser. The rest are internal and are used for running the DHT.

An example of the DHT class being used is located in the [Sequence Diagram](#_Sequence_Diagram) located at Section 9.5. This shows a put request being placed on the DHT, which then communicates away to other DHT’s to store the information.

The republish method will not be implemented in this proof of concept version but will be implemented as the Kademlia paper recommends.

Locally, the Get command will first search in the local PHT for the key and then place the responsibility to the DHT to run an ‘iterativeFind’ for the key. It has been called this, because it will call multiple nodes to get the key as fast as it can. This will create an event which will be returned so once one has been found it will be returned and the rest of the connections dropped as they are not needed.

Figure 16 – UML2 Class Diagram for the DHT part of the program.



### StaticFileServer

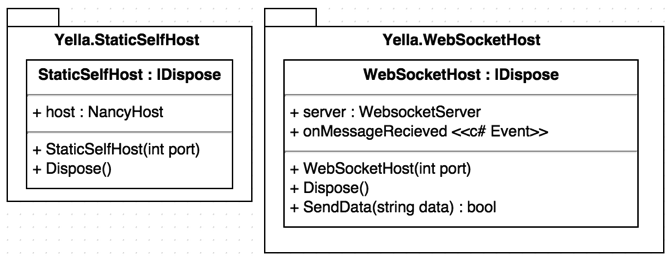
As shown below in Figure 20 this is a relatively simple module which contains the package Nancy. That does the static file hosting for the files required for the site.

### WebSocketHost

The ‘WebSocketHost’ is very similar to the ‘StaticFileHost’ but with an event handler so the Yella module can accept messages and a ‘SendData’ method to send messages.

Both of these use the disposable pattern. As they takes on system ports which require system handles to host/talk to the website. It will then correctly handle the clean up of these system handles and prevent resource leaks which can be common within languages that do automatic garbage collection.

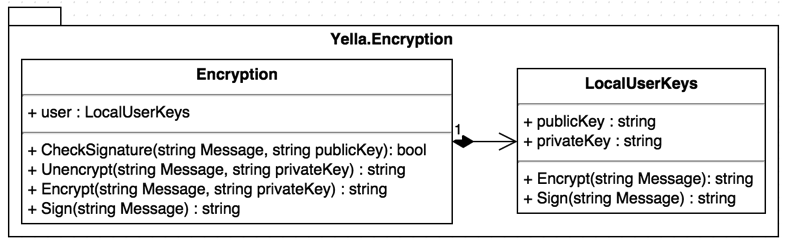
Figure 17 - UML2 Class Diagram for Static Self Host module and the Web Socket Host Module



### Encryption

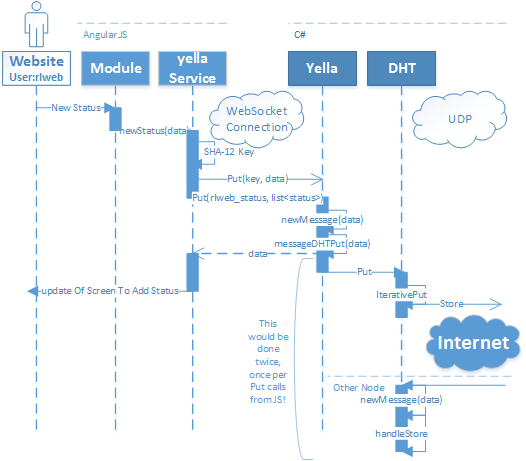
The encryption module has two classes as shown in the UML diagram below in Figure 21. One, which will initiate a ‘LocalUserKeys’, which is a business type class with keys that have been saved to disk previously or it create new ones. The main class will include the basic calls, which can be used to encrypt, unencrypt, sign and check the signature of data. The Encryption module will wrap a PGP library so it can be swapped out in the future.

Figure 18 - UML2 Class Diagram for the Yella Encryption Module



## Sequence Diagram

Figure 19 - Sequence Diagram showing a New Status Task



The above diagram shows the code sequences taken across the multiple languages and the transport used. So from the actor, where a new status has been submitted, the module code calls the main Yella Service. This is within the Angular JavaScript code. The status Data will be stringified and turned into JSON and a hash created from that. The YellaService will then create two put calls, one to update the ‘rlweb\_status’ list with the new hash of the data. The second call will be the put of the data against the hash created. The WebSocket connection sends the JSON request to the C# where it will be received by the ’NewMessage’ Event handler. This handler will then recognise the put request and send it to the DHT to start a iterative put otherwise called a store command.

The other nodes which receive the command will handle the store by putting the data received into their persistent hash table.

## User interface Design

These designs were created with the Jessie James/Apple UI guidelines that were mentioned in the requirements section. Such as the simplicity but also with icons describing their use. The page before they have an account will be a very simple landing page with just the two textboxes, which will be the nickname and user description.

Figure 20 - The home page design

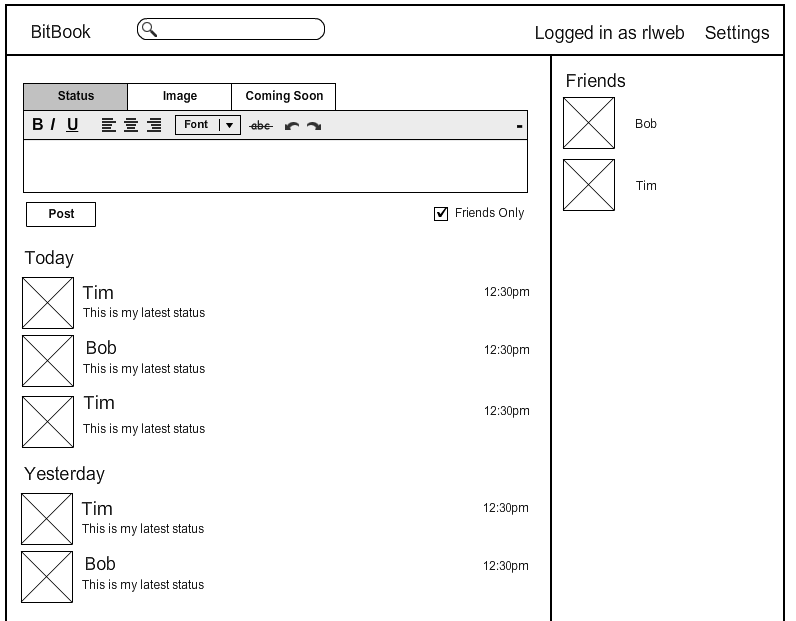
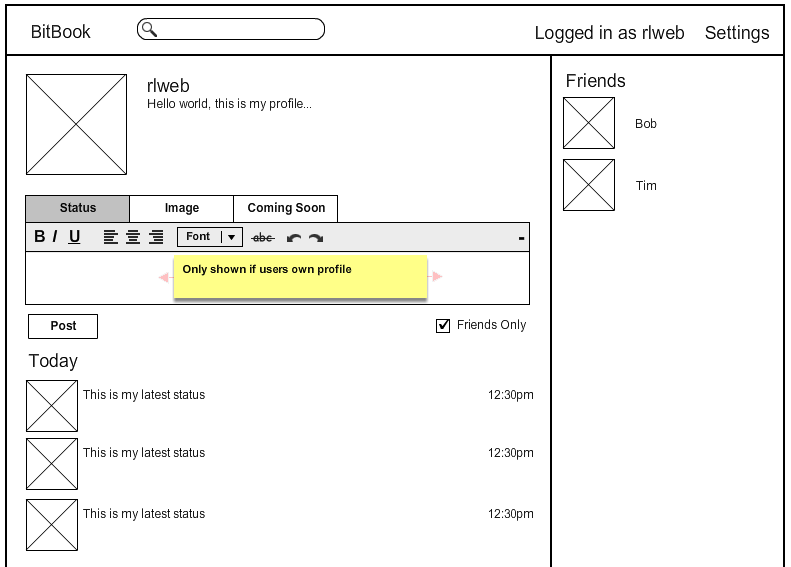


Figure 21 - The user profile design

****

From the wireframes above feedback was received from a few people as the requirements required constant reflection from the user classes:

* “I can see the simplicity and the similar elements from Facebook which is good and easy to use” said a student.
* “It’s very intuitive which is very good as LinkedIn is stupidly complex” said a parent.

MockFlow was used to create the two above figures.

## Protocol Messages

The protocol messages are the same as the Kademlia RPC’s for simplicity and they are shown in [Section 6.4.4](#_Kademlia). These messages will be encoded as JSON as they are the smallest payloads.

### Find Node

This takes a parameter of Node Id and what will be returned are the k number of nodes closet to that Id.

Find Node Request:

{"id" : "<request node id>", "target" : "<response node id>"}

Find Node Result:

{"id" : "<response node id>", "nodes" : "<list of nodes>"}

<list of nodes> example:

"nodes" = [["<IP>", <port>], ["<IP >", <port>], ...]

### Find Value

A find value message will be used when the key is in the local PHT. The Kademlia algorithm will be used to choose the node to send the message too. If the key/value were located at that node it would be returned or it will send back a list of nodes likely to have that key.

Find Value Request:

{"id" : "<sender nodes id>", "key" : "<value key>"}

Find Value Repsonse:

{"id" : "<response node id>", "value" : "<key value if found>", "nodes" : "<list of nodes\*>"}

\* Example of list of nodes located at 9.7.1.

### Store

A put message would be sent to the nodes. If no response were received, it would send it to the next closet node. Error will generally return nil or true if an error occurred on the remote node.

Store Request:

{"id" : "<request node id>", "key" : "<store key>", "value" : "<store value>"}

Store Response:

{"id" : "<response node id>", "err" : "nil"}

### Ping

Ping will be used as a ‘keep alive’ message to check that the node is still awake.

Ping Request:

{"id" : "<request node id>"}

Pong Response:

{"id" : "<response node id>"}

### Node/Key ID’s

In this proof of concept we will use the 160-bit Node ID’s as specified in the paper. The key ID’s will be a hash of the file if possible. If the file was to change in future, we will use the hash of a piece of data. For example their username.

## Stored Data Types

As we are using a key/value storage, the data will be stringified using JSON. On taking this data into the language we would serialize it into the relevant classes if to be used in C# or will be turned into a object in JavaScript by using the JSON.Deserialize().

Attached to the end would be the digital signature created by the PGP library in use. The digital signature can then be checked when it is being loaded against the public key which would be part of the publically available user profile.

### Example User Profile

{

"createdOn": "2014-11-23T18:25:43.511Z",

"nick": "milkybar",

"desc" : "The milkybar kid is …"

}

Below in Figure 25 shows the UML diagram of the user profile and the status and the inheritance of module data. Pictures in the future could be added by either linking the picture or directly embedding the image data into the JSON.

### Example Status

{

"nick": "milkybar",

"createdOn": "2014-11-23T18:25:43.511Z",

"type": “Status",

"version": "v0.1",

"data": {

"status":"Hello World!"

}

}

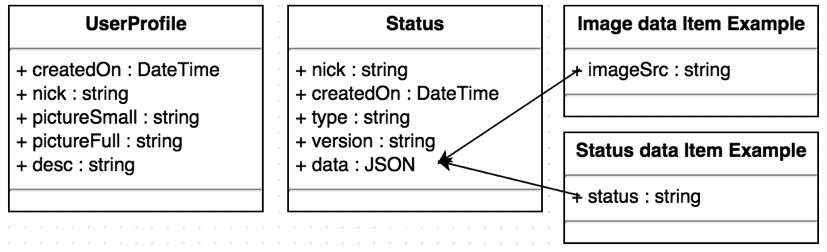
The data part of the status would be JSON which would be passed straight onto the browser without being de-serialized. The data would be include whatever is needed for that module to run.

### Example Linking Data Structure

This structure is a simple list which using the users nick with ’\_status’ located at the end will retrieve a list of status identifiers. The identifiers if used will retrieve the JSON Status shown above at Section 9.8.2.

"milkybar\_status": [  
 "mfqrf8prtm",  
 "tvqtyc9qr9"  
],

Figure 22 – UML Diagram of Business Data Types, which will be stored in the DHT.



# Implementation

This chapter is in order of work done as such as a log book.

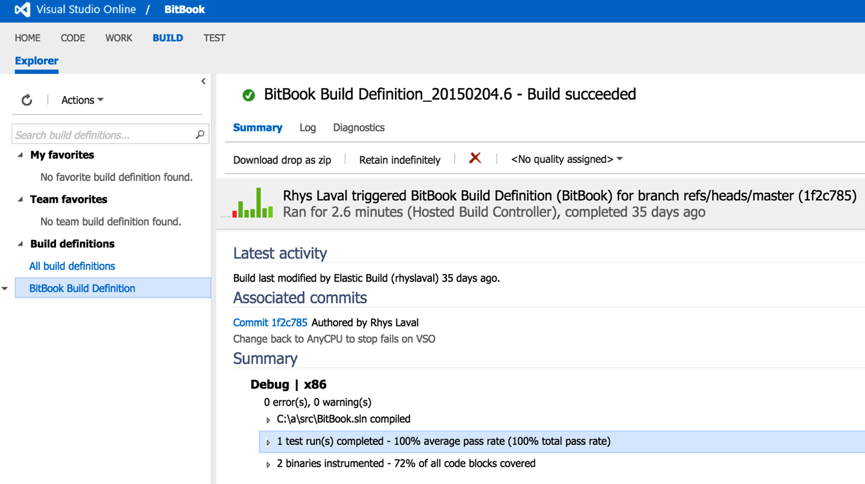
## Version Control Software & Build Server

GitHub was used as version control software which extends to be able to be used as a bug tracker in future but also integrates with many open source tools. SourceTree is a GUI used to interact with the Git server hosted by GitHub. The code is available here[[6]](#footnote-6).

Using Travis to create builds and to check that the tests were running without failure enabled us to test our code on a Linux platform, which they will be expanding to Mac soon. Travis is free for open source projects and you can see the last Travis build here[[7]](#footnote-7).

Pushing the code to another remote branch, which was easy to setup. The other remote branch was Microsoft Visual Studio to run the build and then afterwards get the tests to run on a Windows machine. This ensures cross platform compatibility at every step of the implementation. Visual Studio was also setup to run coverage on the DLL’s created, as can be see below it was at 72% at this commit.

Figure 23 – Visual Studio Online showing a build, the test run and the test coverage.



## Self-hosting Server

After looking at writing my own simple server to just server the static files located in a directory. On the Internet there were many packages that could do this, so without trying to reinvent the wheel. Nancy is a simple server which is very powerful and in the future could be used to extend the site easily. The two packages used are Nancy and Nancy.Host.Self which creates a server without any external software needed. Nancy also had a package to help with the testing of the server, which I used to create a few tests to test the static server. As per commit[[8]](#footnote-8).

## Server-side Web sockets connection

After spending a large amount of time looking into having Nancy passes on the connection to a WebSocket package. Unable to find one that worked with Mono C#. It is possible with OWIN and Nancy but looks like it only works with the latest C# as part of the implementation used System.Web.Websockets which is part of the .net 4.5 libraries which are now open source so with time they would be released to Mono C# also which would simplify the code in the future.

Fleck was used but needed one change of the code for it to work fully with Mono. This was because of a change of an enumeration, which did not match in Mono Dll’s compared to the latest Microsoft Dll’s. For this I forked the Fleck project and added a complier flag so the one line would only be ran the Microsoft Dll’s not the Mono Dll’s depending on how it is compiled.

Code Snippet 1 – Showing how the issue faced with Fleck and Mono C# was fixed

#if \_\_MonoCS\_\_  
// none  
#else

socket.SetSocketOption(SocketOptionLevel.IPv6, SocketOptionName.IPv6Only, false);  
#endif

With this I also added the compiler flag to a test that tested the IPv6 compatibility. The GitHub commit for these changes are located here[[9]](#footnote-9). Then ran the unit tests on a Windows computer with the Microsoft Dll’s to check that it was cross compatible. The project owner of Fleck pulled my changes in to his repository for other users[[10]](#footnote-10). I did try using git submodules to pull in the module from the fork but this broke the build on Visual Studio Online as they want you to really use NuGet so once the change was on and pushed to NuGet. I changed the dependency back to NuGet.

A sample page served by Nancy was created to test the WebSocket connection with the browser, this worked well and messages were sent and received by both sides.

## Website

For the frontend, another package manager was to be used called Bower. Bower works as to allow easy installation of packages for the frontend work so such as frameworks, libraries, assets and utilities. The packages added were:

* JQuery, ~2.1.3 – Basic JavaScript library which includes DOM traversal & manipulation and also AJAX.
* Bootstrap, ~3.3.2 – Frontend framework for quickly building HTML pages.
* Angular-latest, ~1.3.10 – JS framework built for making interactive web applications.
* Ui-Router, ~0.2.13 – A routing framework for angular which allows the interface to be turned into a state machine, which you can control the layout whilst being on one page.
* Font-awesome, ~4.3.0 – A font framework which includes many icons which can be used.

Grunt was also added, so the packages installed with bower could be minified and put into one file, which could be easily included in the html file. So for example all the CSS from the above packages were put into one file.

All these files are so far static and there are a lot of extra files for packages such as documentation and un-compiled files. In the build options for the BitBook project, only files needed have been selected so that the build is still quick and fast. Once this was done, I added tests to check that the required files would be put into the built project.

Once the above was correctly set-up, the website was created. The following screenshots show the website before any dynamic functionality has been added. Comparing these to the designs in chapter 4, I have kept the design the same as possible. In line with the Mac OSX design guidelines, which were mentioned in the Requirements section of the UI design, I have included simple images to describe of the buttons to create a mental model of what it does.

## Complete Website Designs

The landing page is the first page, the user is shown as they create an account. It only asks for their nickname and a tag line for their profile. From here this creates the account and sends the details to the DHT to be saved. Once this is saved, they are shown the homepage.

|  |  |  |
| --- | --- | --- |
| Figure 24 – Landing Page (Mobile) | Figure 25 - Landing page (Tablet size) |  |

Figure 26 – Landing page, desktop size

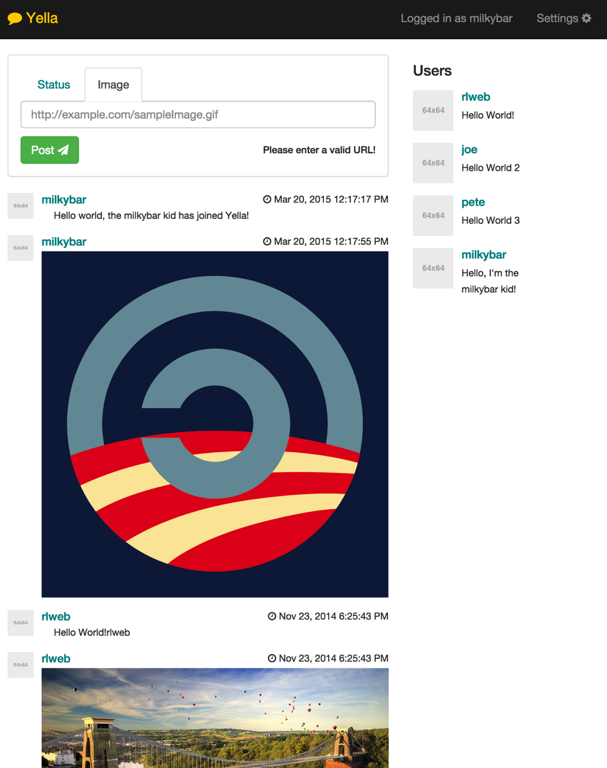


These screenshots show that in the future, the website works well on all sizes of screens. That only the backend DHT would have to be created to work on mobiles and tablets.

This is the homepage once a user has logged in. From here, from the top, you can post a status and below it shows the latest statuses. On the side it shows all users on the network. In the future, this would be changed to show only friends.

|  |  |
| --- | --- |
| Figure 27 – Homepage design (Mobile) | Figure 28 – Homepage design (Tablet) |

Figure 29 - The front page website design in HTML/CSS (Desktop)

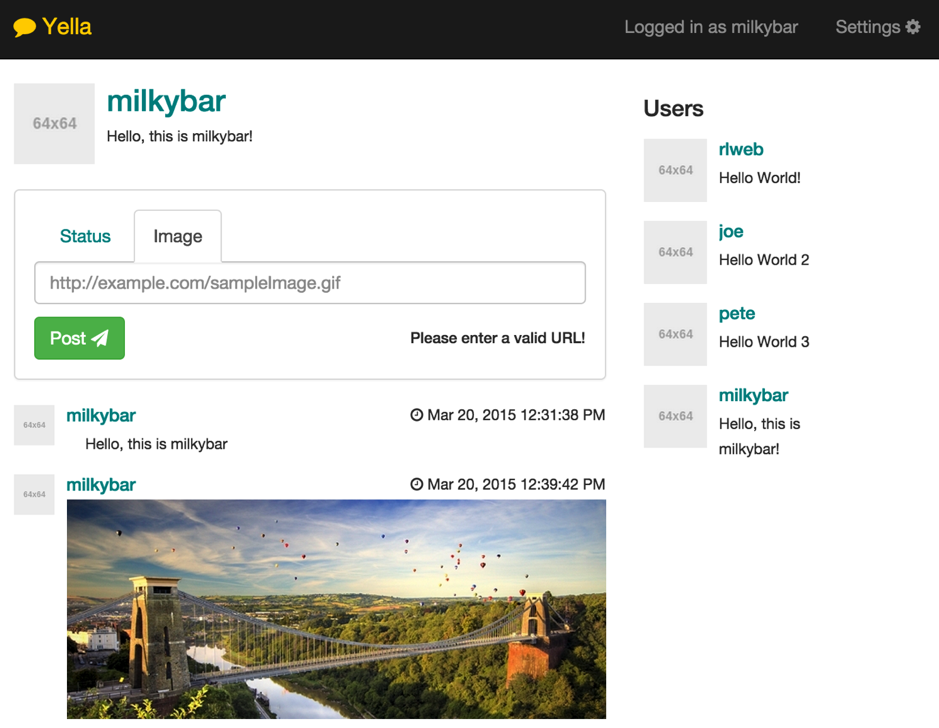


The above screen shot shows the front page of website logged in as the user ‘milkybar’.

The User profile pages show the users details at the top, if it was the same users logged in it allows them to submit a status and then below is the statuses for those users.

|  |  |
| --- | --- |
| Figure 30 - User profile page for a user (Mobile) | Figure 31 - User profile page for a user (Tablet) |

Figure 32 - User profile page for a logged in user (Desktop)



The above screenshots were generated using Chrome’s developer tools, allowing a developer to choose a device and it was recreate the size and settings of that device.

Feedback was received after allowing a few people from the user classes use the site. The following are some:

* It is very professional! I like the simplicity of it like Facebook used to be! – Frank
* Need’s to fill the page, to much white space! Mike
* Bootstrap, Looks cool though, maybe have something separating each post? Like a super thin line or something?

This feedback was gathered to bring the constant reflection of the designs as mentioned in the requirements chapter of this report. From this feedback a line was added to separate the posts to make it clear that they were separate. This can be seen at commit c90a109[[11]](#footnote-11).

## Browser side - Angular.JS

From the start we created the Angular.JS from the bottom upwards of the controller diagram. First creating the DHT, which was faked with a Key Dictionary store and some faked JSON data so we could see elements of the website working without having to implement the whole DHT in one go across languages and communication with them. This code is within the YellaService.js file.

Once all the code was written for the Modules and services, the registration functionality was written, at this time Local Storage to keep the user details, this is outlined in chapter 10.7 below. The code for the modules is within the Yella.Js file.

Afterwards, it was possible to recognise if the user was logged in or not, the rest of the view’s were implemented within the main layout view. At this Ui-Router was used to split the screen’s used inside of the Yella Layout, this is explained more in chapter 10.7 below.

Once all the screens we’re implemented into the site and were all now dynamic, the modularity was worked on, the end implementation is written in 10.9

### Ui Router

The main three states as they are called within Ui-router or parts of the Yella Layout are:

* Index (/) – contains the main user newsfeed
* Settings (/settings) – contains a page with settings such as changing the user nick name, description and security details for future encryption.
* Profile (/profile/:nick) – contains a users profile with the relevant profiles status’s.

The bracketed text is added to the end of the URL to send the user there but also so the browsers back and forward buttons can be used. How the states are implemented within the Javascript can be seen at the bottom of the Yella.Js file.

These can be linked to within HTML shown in Code Snippet 2 or via Javascript shown in Code Snippet 3.

Code Snippet 2 – Showing a Ui Router link in HTML

<a class="pull-left" ui-sref="profile({nick:status.nick})">

The above example show’s giving a nickname from the current angular.js scope called status.nick.

Code Snippet 3 – Showing linking with Ui Router in JavaScript to index

$state.go("index");

### LocalStorage

Persistent storage in the browser has been something missing for a long time with many technologies come and gone such as Flash Cookies, Google Gears and IndexedDB. Cookies for a long time have been used for this and they have worked very well as they can also be access by the server during a request. LocalStorage is now in the HTML5 specification and has been adopted by all broswers.

LocalStorage will be used to keep which user is currently logged in on the computer.

This is the currentUser controller, which has one property, which is user. The user is found by Code Snippet 4.

Code Snippet 4 – How localSorage is used to get the current user

**if** (localStorage.getItem("user")) {  
 **return** yellaService.getUser(localStorage.getItem("user"));  
}  
**return false**;

In the registration controller, if the user has been successfully created we use the following code to save the nick into the localStorage.

Code Snippet 5 – Local Storage being used to set the user.

localStorage.setItem("user",$scope.nick);

From the above code, it shows how easy it is to get and set items in the store, which is very similar to how a DHT works, in fact it could be called a PHT.

### Module System

Using Angular.JS allowed the Module system to work well without complexities. An array of modules is kept in the ‘YellaController’ but in future this could be dynamically given from the C# depending on the Modules located in the Module directory.

Inside each Module Folder is three files:

* PartialSubmit.html – Contains the html to be added to the submit form.
* PartialTimeLine.html – Contains the code for displaying a item in the news feed which could be located on the front page or on a users profile.
* Script.js – Contains the code to power the two above files.

One of the issues with the Modularity was that the modules had to be loaded before the Controller.Js file as they were required before any status’s were loaded into the page otherwise they would not be loaded in time. This means the Script.Js files are currently hard coded in the html files to be imported but with some more searching for a solution this could be found as Angular.Js has been built with dependency injection which is rather complex.

Currently they can submit statuses through ‘YellaService’ but this should be changed in the future to a separate service just for module interaction and that modules should only be able to access data through that one module service for security reasons.

Code Snippet 6 – Showing the modularity and how it works.

$scope.Modules = ["Status", "Image"];  
$scope.submit = [];  
$scope.submit.Type = $scope.Modules[0];  
$scope.getModuleUrl = **function** (name) {  
 **return** "Modules/" + name + "/PartialTimeLine.html";  
};  
$scope.getModuleSubmitUrl = **function** () {  
 **return** "Modules/" + $scope.submit.Type + "/PartialSubmit.html";  
};

Code Snippet 6 loads modules, which are relevant to where they are being called from.

Code Snippet 7 – An example html of the module submit file.

<div class="panel-body">  
 <ul class="nav nav-tabs">  
 <li ng-repeat="module in Modules" ng-class="{active:submit.Type==module}">

<a ng-click="submit.Type=module">{{module}}</a>

</li>  
 </ul>  
 <div ng-include src="getModuleSubmitUrl(submit.Type)">

</div>  
</div>

Code snippet 7 is for the submit status form. The li tag is repeated for the list of modules and when on clicking of one updates the submit.Type which then will change the ng-include to the relevant module.

### Angular.JS Testing

Using Karma, Mocha and Jasmine to test angular was very simple as Angular.JS was built with testing in mind and because of this the dependency injection (DI) allows easy swap in/out of anything to be mocked.

The packages used for this are:

* Karma[[12]](#footnote-12) as the testing environment which allows for cross-browser testing and also E2E testing. By Browser testing this is done via headless browsers but it can also run the major browsers and test with them for more E2E testing.
* Jasime[[13]](#footnote-13) and Mocha[[14]](#footnote-14) are frameworks which adding functionality for checking assertation and creating easy to understand reports because you can group and describe tests.
* ngMock which is part of Angular to inject mocked controllers/services/factories easily without hassle to the DI.

All of the tests were created around the Controllers and yellaService as the methods were quite complex. Twelve tests were created and ran with the build on Travis.

Allow the JavaScript testing files are located within the test folder within the static web root folder.

Code snippet 8 is one of the tests which is testing the RegisterController within the Controllers.js file.

Code Snippet 8 – Shows an example test simplified to show the DI of angular testing.

// The green shows the name of the group of tests

describe('RegisterController Tests', function () {  
 beforeEach(module('yella'));  
 var $scope, $controller, mock;

// above just setting up, loading the yella angular module

// below is ran before each test  
 beforeEach(inject(function (yellaService) {  
 $scope = $rootScope.$new();  
 mock = {  
 '$scope': $scope,  
 'yellaService': yellaService   
 };

// load the controller and pass the mocked items  
 $controller('RegisterController', mock);  
 }));

// the green shows the test name  
 it('new user submit when name taken', function () {  
 *// mock yellaService newUser to return false* spyOn(mock.yellaService,"newUser").and.returnValue(false);  
 *// call the form code in the controller* $scope.submit();  
 *// check correct changes were made* expect($scope.nameTaken).toBe(true);  
 });  
});

## DHT

Before the DHT was worked on a basic PHT was made using a Dictionary, this is fully tested project with code coverage at 100%.

The DHT is not finished at this point but it is to a point where the node completes the bootstrapping process. Where nodes spread information so the nodes can learn of other nodes. The next steps are to implement the Get and Put functionality within the lines of the Kadmelia protocol.

The DHT has been created with two testing projects, one for unit tests and one for an integration test so it can be tested without the full program needed to be ran.

# Testing

From the implementation, testing was done via White Box Testing. Unit-testing was done inline with Test Driven Development (TDD), which was mentioned in the requirements. Unit testing provides quick testing of separate parts of code, which also allows for mocking and is sandboxed. Integration tests will be used to test the complex parts of the system such as the DHT. The acceptance testing proves that the requirements and design have been followed. The unit tests were ran on both languages through Travis. Code coverage was monitored but is not at the required 70%.

Some basic usability testing was done at each stage; to get feedback on the designs and to see if they would be able to navigate the site.

Documentation of the tests will be in the form of tables and screenshots of the examples of the methods listed above. Below, in figure 34/35 it shows the test runners passing the tests. The test results are located in [Appendix Section 11.4](#_Testing_Results).

Figure 33 – Xamarmin Studio Test Runner

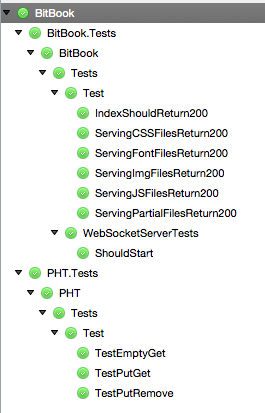


Figure 34 – Karma test running in terminal showing the Angular.JS tests passing in2 browsers.

Milkybar:static rlweb$ karma start

INFO [karma]: Karma v0.12.31 server started at http://localhost:9876/

INFO [launcher]: Starting browser Chrome

INFO [launcher]: Starting browser Safari

INFO [Safari 8.0.4 (Mac OS X 10.10.2)]: Connected

INFO [Chrome 41.0.2272 (Mac OS X 10.10.2)]: Connected

Chrome 41.0.2272 (Mac OS X 10.10.2): Executed 12 of 12 SUCCESS (0.08 secs / 0.075 secs)

Safari 8.0.4 (Mac OS X 10.10.2): Executed 12 of 12 SUCCESS (0.039 secs / 0.037 secs)

TOTAL: 24 SUCCESS

# Conclusion

The findings of the report have found that the technology for the solution could work. An issue may be scaling the product up to the number of users which some of the current sites work on. BitTorrent DHT is the largest example of this but works well with very small amounts of data.

Many developers would get involved in this interesting new approach of using the technology. This would then cause it to be used in other area’s such as decentralised websites, where anyone could publish a website without inherent cost! The control would be kept in the owners hands and outside that of large corporations. From the evolution of this project and seeing projects like MaidSafe and BitCoin advancing quickly. Peer-to-peer technologies will be come more apparent in the future.

The product could be brought into the world but there are still some issues to be worked on which are outside of the technological side. One of the questions from the introduction was “Could moderation be spread across users?” This aspect would need to be improved.

The social side of social networks needs to be researched on a larger scale. Beyond the current research which focuses on addiction. To see more of the long term psychological effects especially on younger users. The issues found need to be researched to minimise these effects for future generations as social media is now something users have been found to, not too be able to live without.

Lastly, the project goals were:

* **Free** - it has been shown that the project could be free without any cost.
* **Open source -** is why it would be free and still advance by using people power!
* **Decentralised** - allowing no central server architecture making it harder for governments to read the data.
* The project is **social** because it is created by its users and could even be moderated by its users.
* It is **modular** as anyone could create a newsfeed status plugin.
* PGP has shown that **security** could be implemented into the software, which is unbreakable and has lasted many years.

## Future Work

To bring the product up to a standard that could be released, the following improvements would be required:

* The proof concept needs much work to produce a full working piece of software which needs to the full Kademlia algorithm implemented. Also to bring various improvements outlined in research papers to extend the Kademlia algorithm to help it work with a large number of nodes.
* To bring a login system so users could login wherever they are.
* The testing of a large number of users would be needed. This could be totally automated such as how the KadOH testing example is shown in [Section 6.8.9](#_Peer_2_Peer). This type of testing is called Concurrent testing.
* The security/privacy part of the project would also need to be completed and an external security analysis would be required to check that it is a solution which is safe to be advertised as secure.

Following through with the Open Source method and having one or two contributors, this project could be brought to a fully working version within months! This does not solve the social aspects of the social network though. The designs should be looked into more, to see if it would be possible to reduce stress/envy, which was show to be a major issue from the primary research.

## Evaluation

From reflection on the methodologies chosen, it was found that maybe JavaScript would have been a better language to be used. The KadOH and BitCoin’s showed that it could be used across tablets and mobiles without the need for the full DHT to placed on phones decreasing the bandwidth needed. The recent changes in how JavaScript can be used as browser extension make it a great choice and it allows the new API’s to be used, which have been outlined at the end of the [Methodologies section](#_Other_Methodologies). On the other hand, it would not have to be in the browser but could also work as a Node.JS background application.

The development worked well because of the fast iterations and constant feedback. One of the issues was that the DHT had to be written instead of using a library. If the DHT was selected to be coded first, this would lead to issues to be brought up sooner, so maybe a change in the coding language could have been made earlier on rather that later on in the process. When a lot of the code for the communication between the browser was already coded. This could of be averted by specifying that the DHT was to be fully coded and tested first before moving on to the other features of the project. Using MoSCoW did not lead to this but maybe this could be brought up that testing the DHT first would have been critical to ensuring that proceeding functionality would have required the DHT.

## Reflection

This project from start to finish was interesting with many different aspects and views that could be explored. It made awareness of the ethical side of these social media sites, which show to be needed to be reformed. In retrospect it would have been more interesting to probe the social side more and to see how the design of social media sites could be changed to reduce the angst, which was found to be secondary result of using social networks. It had many difficulties, which had to be overcome.

## Closing Statement

As people are becoming more aware of how the world is becoming a surveillance state with governments creating a panopticon state. This awareness will lead people to be more cautious with their interaction with large corporations who are getting closer to the government. As this is being written, Facebook is been investigated by the European Union in terms of tracking logged out users by using analytical cookies (The Guardian, 2015). As Russell Brand talks about a revolution is near and the Internet will be the leader of it (BBC Newsnight, 2013). Hopefully Tim-Burners Lee will see the Internet change into what he had hoped his concept would be.

“Social media has infected the world with a sickening virus called vanity.” (Elmore, 2013)

# Appendix

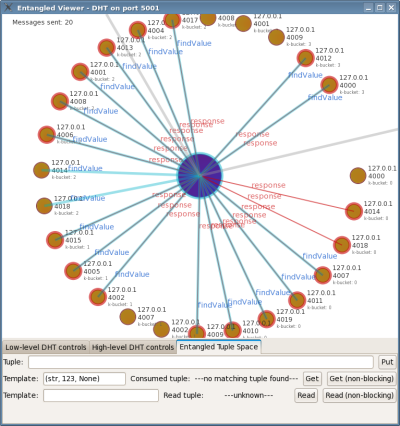
## Peer to Peer Libraries

Rhino DHT[[15]](#footnote-15) is part of Rhino’s suite of C# libraries which include Service Bus, ETL and Queues. It looked to be a complete solution except that it was not mentioned if it would work under Mono C#. After trying to use it, problems bubbled up and it was found it was not a finished solution. After spending many hours trying to finish the solution, it was found not to be useful as it was very complex.

NChord is an old C# implementation of the Chord algorithm. Firstly, the code when downloaded would not run as a command line program. Secondly, it seems not to run correctly in Mono C#.

Entangled[[16]](#footnote-16) is a python implementation of a DHT based on the Kademlia algorithm. The examples they supplied worked well and it handled even the JSON data that was tried. A few python integration libraries were tried to use Entangled in C# but keeping the process running whilst asking commands seem to be an issue. It would only access new running processes instead of the pre-existing process. In addition it includes delete functionality and keyword search space functionality.

Figure 35 – An example of the Entangled Viewer



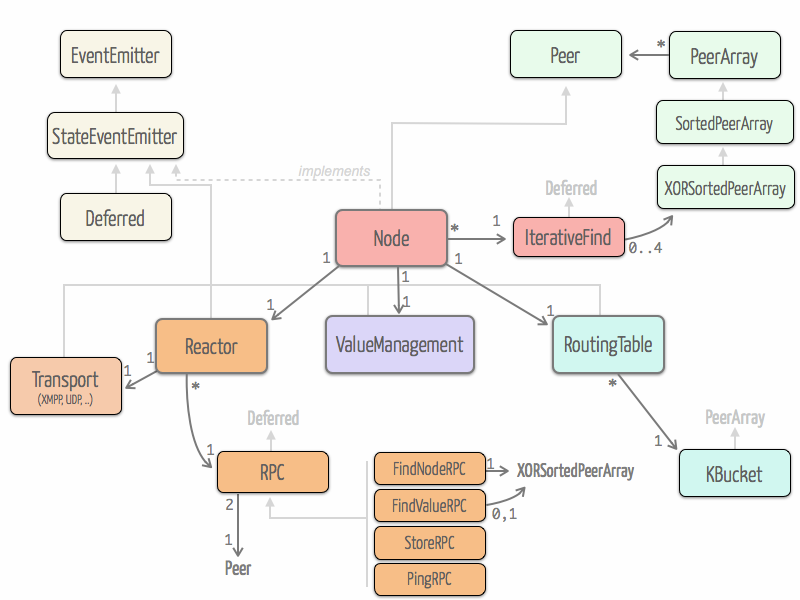
TeleHash[[17]](#footnote-17) is another library, which could be used for the DHT, but it is over complicated for our integration. The C# library is still in very early days compared to their other language integrations such as Node.JS and C. They interestingly use open BitTorrent trackers for bootstrapping nodes. This idea could be used for automatic bootstrapping of the application instead of asking for one node so the program knows where to start connecting to the network.

LibTorrent[[18]](#footnote-18) is used for Twister’s implementation. Fundamentally, it is based on the torrent protocol and uses torrents as a user and each torrent data part is post. Looking at the amount of work Twister did, it seems unfeasible to be used in this project.

MaidSafe was going to be peer-to-peer library to be used but after some testing with the library it was over complicated and as it is only in beta, it was found to buggy to be used. MaidSafe are currently in very early versions of creating a global peer-to-peer network to be used for decentralized applications with the idea of payment for helping the network.

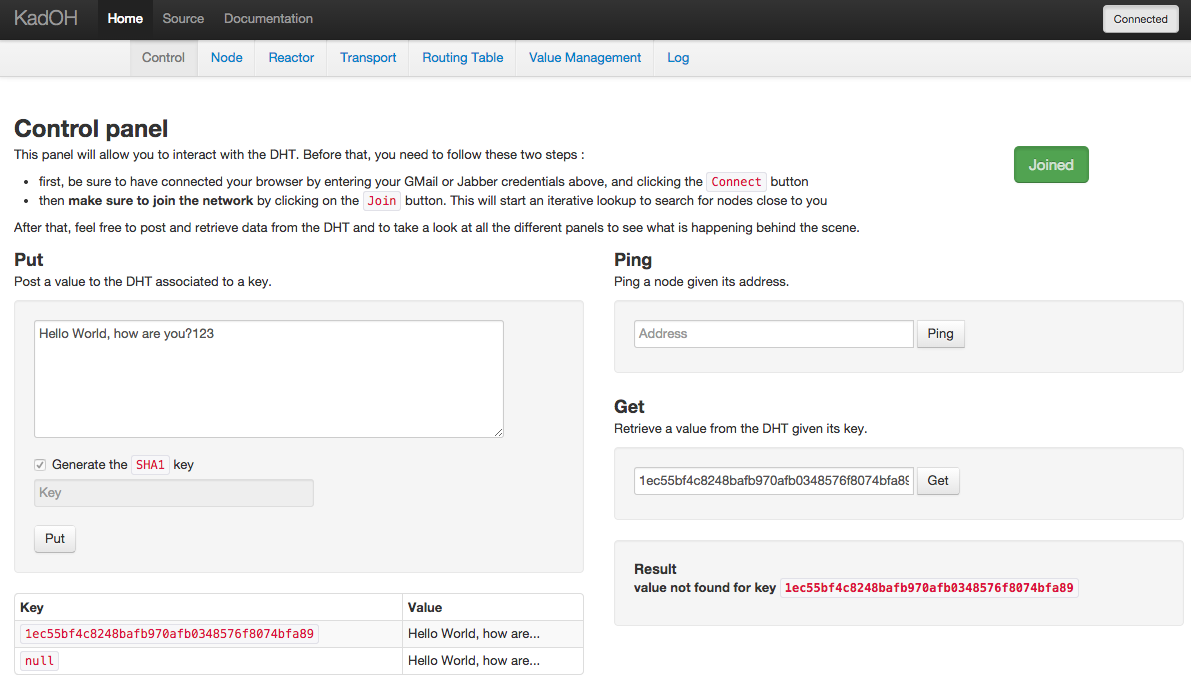
KadOH[[19]](#footnote-19) is a implementation of a Kadelia over HTTP. It was created in node to be a framework that brings DHT to mobile applications. This is an interesting framework written in Node.JS. As it works from the side of a server could help in getting working decentralized site to mobiles via using a client located on a computer. Written by Alexandre Lachèze and Pierre Guilleminot for which they used idea’s from Tangled to create it as part of a semester project at University of Maryland. The report, which is located at the Github repo also. Their architecture is shown below and works using the Reactor pattern.

Figure 36 – KadOH Architecture Diagram specific to the DHT implementation.



(Lachèze & Guilleminot, 2012)

Figure 37 – KadOH even comes with a full testing suite!



(Lachèze & Guilleminot, 2012)

Other libraries looked into:

* A Twister DHT implementation [[20]](#footnote-20)
* A C++ implementation written by the MIT team[[21]](#footnote-21)

## Project Files/Folder Structure

Static Folder Structure

|-Modules

|---Image

|---Status

|-css

|-fonts

|-js

|-partials

|-test

Backend Folder Structure

|-BitBook

|---Properties

|---static – The website which the folder structure is shown above.

|-BitBook.StaticSelfHost

|---Properties

|-BitBook.Tests

|-BitBook.WebSocketHost

|-DHT

|---Messages

|---Nodes

|---Properties

|-DHT.ConsoleTest – contains a method of testing the DHT through a console.

|-ExternalItems

|-PHT – The PHT component

|---Interfaces

|-PHT.Tests – contains Unit Tests relating to the PHT

|-packages - contains all the NuGet Packages

## Test Cases

**Test Name: Testing simple storage engine**

Requirements Tested: REQ-1-1: Backend storage mechanism of the key/values

Outline: Ensure that basic backend storage is working correctly before testing peer-to-peer network

Pre-requisites: Create an executable test suit or unit tests

Method:

|  |  |  |
| --- | --- | --- |
| Step | Action | Expected Observation |
| 1 | Test Get(sample key) on Empty Storage | Unable to find key |
| 2 | Test Put(sample key, sample value) | Should return success |
| 3 | Test Get(sample key) | Should return sample value |

**Test Name: Testing peer-to-peer network**

Requirements Tested: REQ-1-2: Start joining the peer-to-peer network

Outline: Ensure computers can talk to each other and keep track of users on the network.

Pre-requisites: Creation of an executable test program, which can be, ran on multiple ports on the same machine.

Method:

|  |  |  |
| --- | --- | --- |
| Step | Action | Expected Observation |
| 1 | Program 1 start on port 8000 | Program starts up and awaits for connections |
| 2 | Program 2 start on port 8001 | Program starts up and awaits for connections |
| 3 | Inform Program 2 of Node at local IP with port 8000 | Program 1 returns ping and list of nodes, which will be none. Program 2 returns connection details to screen |
| 4 | Program 3 start on port 8001 | Program starts up and awaits for connections |
| 5 | Inform Program 3 of Node at local IP with port 8002 | Program 1 returns ping and list of nodes, which will be one triplet of node ID, IP, port. Program 3 returns connection details to screen. It will initiate connection to local IP/8001 and Program 2 returns connection details to screen. All three nodes should be connected to each other informing each other of each node. |

**Test Name: Testing PUT of the DHT**

Requirements Tested: REQ-1-3: Put Mechanism to save to backend storage but also propagating the information on the P2P network. ; REQ-1-4: Get Mechanism to get the information which if not in the backend storage needs to get searched on the P2P network.

Outline:

Pre-requisites: Creation of a test tool to be able to run the DHT backend and to make calls of PUT/GET through it with basic strings.

Method:

|  |  |  |
| --- | --- | --- |
| Step | Action | Expected Observation |
| 1 | Test Get(sample key) on Empty Storage | Unable to find key |
| 2 | Test Put(sample key, sample value) | Should return success |
| 3 | Test Get(sample key) | Should return sample value |
| 4 | Start up another Test tool, connect to existing opened program. | Both programs should return that they are connected. |
| 5 | On Program 2 - Test Get(sample key) | Should return sample value |
| 6 | On Program 2 - Test Put(sample key2, sample value2) | Should return success |
| 7 | On Program 1 - Test Get(sample key2) | Should return sample value2 |

**Test Name: Testing storing of User Profile**

Requirements Tested: REQ-2-1: Backend storage mechanism of user profile

Outline: Ensuring data is saved/serialized correctly from Classes to the DHT and back out into the classes.

Pre-requisites: DHT loaded correctly and integrated with main solution.

Method:

|  |  |  |
| --- | --- | --- |
| Step | Action | Expected Observation |
|  | Data to be used:  Username: rlweb  Bio: Why, Hello world? |  |
| 1 | Create Example User Object and print to JSON format and pushed to DHT corrected. | All fields shown same in Class/JSON and it is statically correct JSON in the DHT and whilst being passed around. |
| 2 | Whilst getting data from DHT it taken out and can be de-serialized correctly into the class. | All Data above can be observed correctly in the class. |

**Test Name: Testing storage of User Status**

Requirements Tested: REQ-2-2: Backend storage of the user profile data for example a text status but for future this should allow other data types.

Outline: Ensure data can be serialized and de-serialized without any issues

Pre-requisites: All classes created corrected and a test program which can write to console.

Method:

|  |  |  |
| --- | --- | --- |
| Step | Action | Expected Observation |
|  | Data to be used:  User ID:  Status: Test Status |  |
| 1 | Create Example User Object and print to JSON format and pushed to DHT corrected. | All fields shown same in Class/JSON and it is statically correct JSON in the DHT and whilst being passed around. |
| 2 | Whilst getting data from DHT it taken out and can be de-serialized correctly into the class. | All Data above can be observed correctly in the class. |

**Test Name: Creation of a user’s profile**

Requirements Tested: REQ-2-3: Creation of a user’s profile

Outline: Ensure that users profile can be created and view their own user profile.

Pre-requisites: The program is running with no local user created on computer

Method:

|  |  |  |
| --- | --- | --- |
| Step | Action | Expected Observation |
| 1 | Load up the program and make sure it’s connected to the network. | Program loaded correctly and that we can load the homepage. |
| 2 | Create User Profile | Creation should show it has been saved successfully |
| 3 | View of own user profile | Able to view own users profile, which contains all correct details for user. |

**Test Name: Viewing a users profile**

Requirements Tested: REQ-2-4: View users profile by their ID

Outline: Ensure that users profile can be viewed with the correct details.

Pre-requisites: The network is running with two users with created profiles

Method:

|  |  |  |
| --- | --- | --- |
| Step | Action | Expected Observation |
| 1 | Load up the program and make sure it’s connected to the network. | Program loaded correctly and that we can load the homepage. |
| 2 | Load up own users profile | Able to view own users profile, which contains all correct details for user. |
| 3 | Load up other users profile | Able to view other users profile which contains all correct details for user. |

**Test Name: Testing viewing of a User Status**

Requirements Tested: REQ-2-5: View user’s status on user profile

Outline: Ensure that user’s status can be viewed on a users profile with the correct details.

Pre-requisites: Creation of a two user profiles with status on each

Method:

|  |  |  |
| --- | --- | --- |
| Step | Action | Expected Observation |
| 1 | Load up the program and make sure it’s connected to the network. | Program loaded correctly and that we can load the homepage. |
| 2 | Load up own users profile | Able to view own users status on their own profiles, which contains all correct details for the status. |
| 3 | Load up other users profile | Able to view other user’s status on their profile, which contains all correct details for the status. |

**Test Name: Post Status**

Requirements Tested: REQ-2-6: Post to own users profile

Outline: Ensure users can post to their own profile and that it can be viewed.

Pre-requisites: Program loaded and connected to network. Already created user profile

Method:

|  |  |  |
| --- | --- | --- |
| Step | Action | Expected Observation |
| 1 | Post Status to Page | Creation should show it has been saved successfully |
| 2 | View Status on profile | Viewing the status, with the correct text which was entered on the last page |

**Test Name: Creation of User Keys**

Requirements Tested: REQ-3-1: User profiles to include encryption keys to be created on account creation

Outline: Ensure security keys are created on user profile creation on the network.

Pre-requisites: Program loaded correctly with no local user profile

Method:

|  |  |  |
| --- | --- | --- |
| Step | Action | Expected Observation |
| 1 | Creation of user profile | Success message |
| 2 | Look at locally private/public made keys and import them into a PGP program | Keys should be accepted into PGP program |

**Test Name: Re-creation of User Keys**

Requirements Tested: REQ-3-2: Re-creation of keys to be allowed

Outline: Ensure if user his security is comprised, to allow recreation of new keys.

Pre-requisites: Program loaded correctly with already created user profile.

Method:

|  |  |  |
| --- | --- | --- |
| Step | Action | Expected Observation |
| 1 | Action the re-creation keys | Success message |
| 2 | Look at locally private/public made keys and import them into a PGP program | Keys should be accepted into PGP program |
| 3 | Data should be rehashed with newly created keys | Data should be still able to be viewed. But the background data should have changed. |

**Test Name: Adding of friend**

Requirements Tested: REQ-3-3: Adding/removing of friends and the transfer of keys between

Outline: Ensure that the keys are passed over to the friend so he can unencrypt the data.

Pre-requisites: Program loaded correctly with an already created user profile and status which are set to ‘Friends only’.

Method:

|  |  |  |
| --- | --- | --- |
| Step | Action | Expected Observation |
| 1 | Go to a Users Profile | Can view users profile |
| 2 | Click the Add Friend button | Should show success message |
| 3 | Load the program as friend | Should be able to see ‘Friends only’ and ‘everyone’ status |

**Test Name: Removal of friend**

Requirements Tested: REQ-3-4: Adding/removing of friends and the transfer of keys between, REQ-3-5: Show the data or unencrypted data first if friends.

Outline:

Pre-requisites: Program loaded correctly with an already created user profile and status, which are set to ‘Friends only’ and a status of ‘Everyone’.

Method:

|  |  |  |
| --- | --- | --- |
| Step | Action | Expected Observation |
| 1 | Go to a Users Profile | Can view users profile |
| 2 | Click the Remove Friend button | Should show success message |
| 3 | Load the program as friend | Should be NOT able to see ‘Friends only’ status but be able to see ‘Everyone’ status |

## Testing Results

**Test Name: Testing simple storage engine**

Requirements Tested: REQ-1-1: Backend storage mechanism of the key/values

Outline: Ensure that basic backend storage is working correctly before testing peer-to-peer network

Pre-requisites: Create an executable test suit or unit tests

Method:

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Action | Expected Observation | Observation found |
| 1 | Test Get(sample key) on Empty Storage | Unable to find key | Unit Test Pass |
| 2 | Test Put(sample key, sample value) | Should return success | Unit Test Pass |
| 3 | Test Get(sample key) | Should return sample value | Unit Test Pass |

**Test Name: Testing peer-to-peer network**

Requirements Tested: REQ-1-2: Start joining the peer-to-peer network

Outline: Ensure computers can talk to each other and keep track of users on the network.

Pre-requisites: Creation of an executable test program, which can be, ran on multiple ports on the same machine.

Method:

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Action | Expected Observation |  |
| 1 | Program 1 start on port 8000 | Program starts up and awaits for connections | Console test ran and worked |
| 2 | Program 2 start on port 8001 | Program starts up and awaits for connections | Console test ran and worked |
| 3 | Inform Program 2 of Node at local IP with port 8000 | Program 1 returns ping and list of nodes, which will be none. Program 2 returns connection details to screen | Console test ran and worked |
| 4 | Program 3 start on port 8001 | Program starts up and awaits for connections | Console test ran and worked |
| 5 | Inform Program 3 of Node at local IP with port 8002 | Program 1 returns ping and list of nodes, which will be one triplet of node ID, IP, port. Program 3 returns connection details to screen. It will initiate connection to local IP/8001 and Program 2 returns connection details to screen. All three nodes should be connected to each other informing each other of each node. | Console test ran and worked |

**Test Name: Testing PUT of the DHT**

Requirements Tested: REQ-1-3: Put Mechanism to save to backend storage but also propagating the information on the P2P network. ; REQ-1-4: Get Mechanism to get the information which if not in the backend storage needs to get searched on the P2P network.

Test NOT COMPLETE. Pre-requisites missing which was a test tool to show the DHT

**Test Name: Testing storing of User Profile**

Requirements Tested: REQ-2-1: Backend storage mechanism of user profile

Test NOT COMPLETE. Pre-requisites missing but worked with the faked JavaScript DHT, C# needs completing.

**Test Name: Testing storage of User Status**

Requirements Tested: REQ-2-2: Backend storage of the user profile data for example a text status but for future this should allow other data types.

Test NOT COMPLETE. Pre-requisites missing but worked with the faked JavaScript DHT, C# needs completing.

**Test Name: Creation of a user’s profile**

Requirements Tested: REQ-2-3: Creation of a user’s profile

Outline: Ensure that users profile can be created and view their own user profile.

Test NOT COMPLETE. Pre-requisites missing but worked with the faked JavaScript DHT, C# needs completing.

**Test Name: Viewing a users profile**

Requirements Tested: REQ-2-4: View users profile by their ID

Test NOT COMPLETE. Pre-requisites missing but worked with the faked JavaScript DHT, C# needs completing.

**Test Name: Testing viewing of a User Status**

Requirements Tested: REQ-2-5: View user’s status on user profile

Test NOT COMPLETE. Pre-requisites missing but worked with the faked JavaScript DHT, C# needs completing.

**Test Name: Post Status**

Requirements Tested: REQ-2-6: Post to own users profile

Test NOT COMPLETE. Pre-requisites missing but worked with the faked JavaScript DHT, C# needs completing.

**Test Name: Creation of User Keys**

Requirements Tested: REQ-3-1: User profiles to include encryption keys to be created on account creation

Test NOT COMPLETE.

**Test Name: Re-creation of User Keys**

Requirements Tested: REQ-3-2: Re-creation of keys to be allowed

Test NOT COMPLETE.

**Test Name: Adding of friend**

Requirements Tested: REQ-3-3: Adding/removing of friends and the transfer of keys between

Test NOT COMPLETE.

**Test Name: Removal of friend**

Requirements Tested: REQ-3-4: Adding/removing of friends and the transfer of keys between, REQ-3-5: Show the data or unencrypted data first if friends.

Outline:

Test NOT COMPLETE.

## List of Code Snippets

Code Snippet 1 – Showing how the issue faced with Fleck and Mono C# was fixed 78

Code Snippet 2 – Showing a Ui Router link in HTML 88

Code Snippet 3 – Showing linking with Ui Router in JavaScript to index 88

Code Snippet 4 – How localSorage is used to get the current user 88

Code Snippet 5 – Local Storage being used to set the user. 89

Code Snippet 6 – Showing the modularity and how it works. 90

Code Snippet 7 – An example html of the module submit file. 90

Code Snippet 8 – Shows an example test simplified to show the DI of angular testing. 92

## List of Figures

Figure 1 - Percentage of P2P file sharing of Internet traffic in North America 23

Figure 2 – The questionnaire software through a browser. 25

Figure 3 – The full questionnaire 26

Figure 4 - 16-bit Chord Network with the node having ‘fingers’ with 1,2,4,8,16. 32

Figure 5 – The routing table with example buckets. 33

Figure 6 - From the first Kademlia paper outlining the tree and locating a node 33

Figure 7 – The elements of User Experience 53

Figure 8 - Showing the use cases in the system prioritised by MoSCoW 55

Figure 9 - Sequence Diagram showing the movement between actors and page interaction. 55

Figure 10 – Communication Diagram between the different parts 56

Figure 11 – From the Angular.JS Documentation, showing features available to each type of item. 59

Figure 15 – Showing how the Prototyping feedback model works 62

Figure 16 – Component Design of the Modules needed for the backend system to work. 63

Figure 17 - Component Diagram, which covers the Yella MVVM architecture of Angular.JS 65

Figure 18 – UML2 Class Diagram of the Yella Module 66

Figure 19 – UML2 Class Diagram for the DHT part of the program. 68

Figure 20 - UML2 Class Diagram for Static Self Host module and the Web Socket Host Module 69

Figure 21 - UML2 Class Diagram for the Yella Encryption Module 70

Figure 22 - Sequence Diagram showing a New Status Task 70

Figure 23 - The home page design 72

Figure 24 - The user profile design 72

Figure 25 – UML Diagram of Business Data Types, which will be stored in the DHT. 75

Figure 26 – Visual Studio Online showing a build, the test run and the test coverage. 77

Figure 27 – Landing Page (Mobile) 80

Figure 28 - Landing page (Tablet size) 80

Figure 29 – Landing page, desktop size 81

Figure 30 – Homepage design (Mobile) 82

Figure 31 – Homepage design (Tablet) 82

Figure 32 - The front page website design in HTML/CSS (Desktop) 83

Figure 33 - User profile page for a user (Mobile) 84

Figure 34 - User profile page for a user (Tablet) 84

Figure 35 - User profile page for a logged in user (Desktop) 85

Figure 36 – Xamarmin Studio Test Runner 95

Figure 37 – Karma test runner showing the Angular.JS tests passing in2 browsers. 95

Figure 35 – An example of the Entangled Viewer 100

Figure 36 – KadOH Architecture Diagram specific to the DHT implementation. 102

Figure 37 – KadOH even comes with a full testing suite! 102

## List of Tables

Table 1 - A brief comparison of the social networks reviewed. 15

Table 2 - Results for Question 1 27

Table 3 - Results for Question 2 27

Table 4 - Results for Question 3 27

Table 5 - Results for Question 4 27

Table 6 - Results for Question 5 28

Table 7 - Results for Question 6 28

Table 8 - Results for Question 7 28

Table 9 - Results for Question 8 29

Table 10– Shows amount of government requests towards Facebook 42

# Bibliography

Action Fraud. (2011, October 11). *Identity fraud continues to rise with 4 million victims in UK alone - Yet consumers continue to be complacent with their ID.* Retrieved March 3, 2015 from Action Fraud: http://www.actionfraud.police.uk/Identity\_fraud\_continues\_to\_rise\_with\_4\_million\_victims\_in\_UK\_alone

Anderson, D. (2012, Novemember 30). Splinternet Behind the Great Firewall of China. *ACM* *, 10* (11).

Andrea Broughton, B. F. (2013). *The use of social media in the recruitment process.* ACAS.

Andreassen, C. S. (2012). Development of a Facebook Addiction scale. *Psychological Reports* *, 110* (2), 501-517.

Andreassen, D. C. (2012, 05 06). *New Research about Facebook Addictio*. Retrieved 03 03, 2014 from University of Bergen | News: http://www.uib.no/en/news/36380/new-research-about-facebook-addiction

AngularJS, Google. (2015, Unknown). *AngularJS API Docs.* Retrieved April 9, 2015 from AngularJS API Docs: https://docs.angularjs.org/api

Anstiss, D. T. (2012, April 26). *How social media is supporting people with depression*. Retrieved Febuary 18, 2015 from Guardian: http://www.theguardian.com/social-care-network/2012/apr/26/social-media-depression-support

Apple. (Not known). *OSX Human Interface Guidelines*. Retrieved March 3, 2015 from Apple Developer: https://developer.apple.com/library/mac/documentation/UserExperience/Conceptual/OSXHIGuidelines/DesignPrinciples.html#//apple\_ref/doc/uid/20000957-CH18-SW1

Armitage, R. (2002). *To CCTV or not to CCTV?* London: Nacro.

Bangkok Post. (2014, Feb 9). *Youngest warned over selfie addiction.* Retrieved Mar 02, 2015 from Bangkok Post: http://www.bangkokpost.com/news/local/394102/youngsters-warned-over-elfie-addiction

BBC Newsnight. (2013, October 23). *Paxman vs Brand.* Retrieved April 8, 2015 from Youtube: https://www.youtube.com/watch?v=3YR4CseY9pk

BBC. (2014, 10 4). *Stolen celebrity images prompt policy change at 4Chan.* Retrieved 2 17, 2015 from BBC News: http://www.bbc.co.uk/news/technology-29061358

Berners-lee, T., & Fischetti, M. (1999). *Weaving the Web.* London: Orion Business.

BitTorrent. (2012, 1 9). *BitTorrent and µTorrent Software Surpass 150 Million User Milestone; Announce New Consumer Electronics Partnerships.* Retrieved 2 18, 2015 from BitTorrent - Press Releases: http://www.bittorrent.com/company/about/ces\_2012\_150m\_users

Danezis, G. (2014). *SybilInfer: Detecting Sybil Nodes using Social Networks.* Cambridge: Microsoft Research.

Daria J. Kuss, A. J. (2013, September). Internet addiction in adolescents: Prevalence and risk factors. *Computers in Human Behavior* , pp. 1987-1996.

DeCandia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., et al. (2007, October). Dynamo: Amazon's Highly Available Key-Value Store. *ACM* , 14-17.

*Dictionary.com Unabridged.* (n.d.). Retrieved March 23, 2015 from Dictionary.com Unabridged: http://dictionary.reference.com/browse/diaspora

*Dictionary.com Unabridged.* (n.d.). Retrieved March 23, 2015 from privacy: http://dictionary.reference.com/browse/privacy

*Dictionary.com Unabridged.* (n.d.). Retrieved March 23, 2015 from http://dictionary.reference.com/browse/open-source

Druschel, P., Kaashoek, F., & Rowstron, A. (2002). *Peer-to-Peer Systems: First International Workshop,* (Vol. 1). IPTPS.

Einziger, G., Friedman, R., & Kibbar, E. (2009). Kaleidoscope: Adding Colors to Kademlia. *13-th IEEE International Conference on Peer-to-Peer Computing.* *13.* Isreal: IEEE.

Ello. (2014, 8 1). *Homepage*. Retrieved 11 28, 2014 from Ello: https://ello.co

Elmore, K. (2013). *Jagged Little Pieces .* Winter Goose Publishing.

European Union. (2014). *Factsheet on the “Right to be Forgotten” Ruling (C-131/12).* European Commission.

Evans, S. (2015, 2 10). Mark Zuckerberg's fight for bedroom privacy: Inside Facebook billionaire's battle with backdoor neighbor who sold him his California property 'at a discount rate in exchange for entree into Silicon Valley elite. *Daily Mail* . London, UK.

Facebook. (2014, August 20). *Government Requests Report*. Retrieved March 20, 2015 from Facebook: https://govtrequests.facebook.com/

Financial Times. (2013, Jan 3). *Dopamine jolt behind internet addiction*. Retrieved Feburary 18, 2015 from Financial Times: http://www.ft.com/cms/s/0/27514afc-5444-11e2-9d25-00144feab49a.html#axzz3TWcm1EYz

Fowler, M. (2006, June 19). *Supervising Presenter.* Retrieved March 26, 2014 from Martin Fowler: http://www.martinfowler.com/eaaDev/SupervisingPresenter.html

Freitas, M. (2013). *twister - a P2P microblogging platform.* Center for Research in Inspection Tecnology.

Garrett, J. J. (2002). *The elements of User Experience.* New York: AIGA.

Glassman, T. (2012). Implications for College Students Posting Pictures of Themselves Drinking Alcohol on Facebook. *Journal of Alcohol & Drug Education* *, 56* (1).

Gothenburg Research Institute. (2012). *Sweden´s largest Facebook study.* University of Gothenburg.

Gowan, M. (2002, 5 22). *Requiem for Napster*. Retrieved 1 8, 2015 from PC World: http://www.pcworld.idg.com.au/article/22380/requiem\_napster/

Grove, J. V. (2010, 5 28). *Mark Zuckerberg Donated to Facebook Alternative, Diaspora*. Retrieved 11 15, 2014 from Mashable: http://mashable.com/2010/05/28/zuckerberg-diaspora/]

Guardchild. (2010, Janurary 05). *Identity Theft Statistics.* Retrieved March 3, 2015 from Guardchild: http://www.guardchild.com/identity-theft-statistics/

Gwenn Schurgin O'Keeffe, K. C.-P. (2011, March 28). Clinical Report—The Impact of Social Media on Children, Adolescents, and Families. *The American Academy Of Pediatrics* .

Hoback, C. (Director). (2013). *Terms & Conditions May Apply* [Motion Picture].

House of Lords. (2009). *Surveillance: Citizens and the State.* London: House of Lords.

Kelner, J., & Maymounkov, P. (2009, Sep 15). Electric routing and concurrent flow cutting. *Proceedings of The 20th International Symposium on Algorithms and Computation* .

Kreitz, G., & Niemelia, F. (2010). Spotify – Large Scale, Low Latency, P2P Music-on-Demand Streaming. *Peer-to-Peer Computing (P2P), 2010 IEEE Tenth International Conference on* (pp. 1-10). Delft: IEEE.

Kross E, V. P. (2013). Facebook Use Predicts Declines in Subjective Well-Being in Young Adults. *PLoS ONE* .

Lachèze, A., & Guilleminot, P. (2012). *Kademlia over HTTP, a Javascript framework bringing DHT to mobile applications.* Maryland: University of Maryland.

Lambert, N., & Bollen, B. (2014). *The SAFE Network a New, Decentralised Internet.* MaidSafe.

Liberty. (2009, July 30). *Another Home Secretary; another ID card launch - as public support for the scheme sinks*. Retrieved March 13, 2015 from Liberty: https://www.liberty-human-rights.org.uk/news/press-releases/another-home-secretary-another-id-card-launch-public-support-scheme-sinks

Maier, C. (2012). WHEN SOCIAL NETWORKING TURNS TO SOCIAL OVERLOAD: EXPLAINING THE STRESS, EMOTIONAL EXHAUSTION, AND QUITTING BEHAVIOR FROM SOCIAL NETWORK SITES' USERS. European Conference on Information Systems.

Mason, M. (2008). *The Pirate's Dilemma: How Hackers, Punk Capitalists, Graffiti Millionaires and Other Youth Movements are Remixing Our Culture and Changing Our World.* London: Penguin.

Maymounkov, P. (2013, Mar 17). *Kademlia—my biggest contribution by far*. Retrieved 1 29, 2015 from Petar Maymounkov Blog: http://www.maymounkov.org/kademlia

McCullagh, D. (n.d.). *How SOPA would affect you FAq?* From Cnet: http://www.cnet.com/news/how-sopa-would-affect-you-faq/

Mendelson, J., & Fanning, S. (1999).

Meyer, R. (2014, 10 6). *What Firechat's Success in Hong Kong Means for a Global Internet*. Retrieved 2015 from The Atlantic: http://www.theatlantic.com/technology/archive/2014/10/firechat-the-hong-kong-protest-tool-aims-to-connect-the-next-billion/381113/

Miller, R. (2007, 6 5). *P2P Networks Hijacked for DDoS Attacks*. Retrieved 2 10, 2015 from Netcraft: http://news.netcraft.com/archives/2007/05/23/p2p\_networks\_hijacked\_for\_ddos\_attacks.html

Munmun De Choudhury, M. G. (2013). *Predicting Depression via Social Media.* Redmond: Microsoft Research.

Music On the Internet: Is there an upside to downloading?, J–106–96 (United States Senate July 11, 2000).

Nakamoto, S. (2004). *Bitcoin: A Peer-to-Peer Electronic Cash System.* Bitcoin.

Newman, J. (2011, 11 9). *6 things you'd never guess about googles energy use*. Retrieved 2015 from Time: http://techland.time.com/2011/09/09/6-things-youd-never-guess-about-googles-energy-use/

Nieva, R. (2013, 09 05). *Ashes to ashes, peer to peer: An oral history of Napster*. From Fortune: http://fortune.com/2013/09/05/ashes-to-ashes-peer-to-peer-an-oral-history-of-napster/

Ramos, J. (2012, 2 18). *Occupy wall street and the peer to peer revolution a discussion with michel bauwens part i*. Retrieved 2014 from Action foresight: http://actionforesight.net/media/2012/02/21/occupy-wall-street-and-the-peer-to-peer-revolution-a-discussion-with-michel-bauwens-part-ii/

Rasmussen, J. (1983, May). Skills, Rules, and Knowledge; Signals, Signs, and Symbols, and Other Distinctions in Human Performance Models. *IEEE Transactions on Systems, Man and Cybernetics* *, SMC-13*, pp. 257-265.

Rhea, S., Godfrey, B., Karp, B., Kubiatowicz, J., Ratnasamy, S., Shenker, S., et al. *OpenDHT: A Public DHT Service and Its Uses.* UC Berkeley and Intel Research.

Robinson, J. (2014). *the federation - a statistics hub*. Retrieved 11 10, 2014 from http://pods.jasonrobinson.me/

Sandvine. (2014). *Global Internet Phenomena Report 2H 2014.* America: Statista.

Schlosser, M., Sintek, M., Decker, S., & Nejdl, W. (2003). HyperCuP — Hypercubes, Ontologies, and Efficient Search on Peer-to-Peer Networks. *First International Workshop, AP2PC* (pp. 112-124). Belonga: Springer Berlin Heidelberg.

Schmid, G. (2001). *On the existence of a global system for the interception of private and commercial communications (ECHELON interception system).* European Parliament.

Shaer, M. (2013, 5 26). *Life, death, and free culture in the Mission*. Retrieved 11 26, 2014 from Fortune: http://fortune.com/2013/03/26/life-death-and-free-culture-in-the-mission/

Smith, J. (2014, 9 24). *Ello’s Traffic Deluge Almost Caused a Total New User Freeze-Out, Crisis Averted*. Retrieved 11 2014 from BetaBeat: http://betabeat.com/2014/09/ellos-traffic-deluge-almost-caused-a-total-new-user-freeze-out-crisis-averted/

Stoica, I., Morris, R., Karger, D., Kaashoek, M. F., & Balakrishnan, H. (2001, Aug 27-31). Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. *SIGCOMM* .

The Economist. (2014, 4 1). *Peer-to-peer lending Banking without banks* . From Economist: http://www.economist.com/news/finance-and-economics/21597932-offering-both-borrowers-and-lenders-better-deal-websites-put-two

The Guardian. (2015, March 31). Facebook 'tracks all visitors, breaching EU law'. London, UK.

V, S. (2015, 1 19). *In simple terms, how does a BitTorrent client initially discover peers using DHT?* Retrieved 2015 from Superusr: http://superuser.com/questions/592238/in-simple-terms-how-does-a-bittorrent-client-initially-discover-peers-using-dht

Wheeler, D. (2014, 02 16). *Make Your Open Source Software GPL-Compatible. Or Else.* Retrieved 03 02, 2015 from David A. Wheeler’s Personal Home Page: http://www.dwheeler.com/essays/gpl-compatible.html

Whitworth, B. (2006). *Socio-technical Systems.* Chicago: Encyclopedia of human computer interaction.

Wikipedia. (2010, April 21). *Chord network.* Retrieved March 22, 2014 from Wikipedia: http://en.wikipedia.org/wiki/File:Chord\_network.png

Wilkinson, S., & Lowry, J. (2014). *A Blockchain-Based Decentralized File Storage Application.* MetaDisk.

Winter, A. (Director). (2013). *Downloaded* [Motion Picture].

Wright, R. (2000). *Nonzero: The Logic of Human Destiny.* New York: Vintage Books.

Yishanl, Wong, & al., e. (2010). *Patent No. 20090198487.* US.

Zindros, D. (2014). *A pseudonymous trust system for a decentralized anonymous marketplace.* OpenBazaar.

1. bittorrent.org/beps/bep\_0005.html [↑](#footnote-ref-1)
2. https://github.com/sit/dht [↑](#footnote-ref-2)
3. uni-obuda.hu/users/boraros-bakucz.andras/2013/srs\_template.doc [↑](#footnote-ref-3)
4. bitmessage.org [↑](#footnote-ref-4)
5. msdn.microsoft.com/en-us/library/system.net.ipendpoint%28v=vs.110%29.aspx [↑](#footnote-ref-5)
6. github.com/bitbook/bitbook [↑](#footnote-ref-6)
7. travis-ci.org/bitbook/bitbook [↑](#footnote-ref-7)
8. github.com/bitbook/bitbook/commit/64352a42d5e1e75456a1842c30711d6da30cbfff [↑](#footnote-ref-8)
9. github.com/bitbook/Fleck/commit/97169adbf19fb7bb48ff1be5dbfe1f79328bf20a [↑](#footnote-ref-9)
10. github.com/statianzo/Fleck/pull/121 [↑](#footnote-ref-10)
11. github.com/bitbook/bitbook/commit/c90a109711a103e2ba3e6db7c2fd38f0fa209865 [↑](#footnote-ref-11)
12. karma-runner.github.io/0.12/index.html [↑](#footnote-ref-12)
13. jasmine.github.io/ [↑](#footnote-ref-13)
14. mochajs.org [↑](#footnote-ref-14)
15. github.com/ayende/rhino-dht [↑](#footnote-ref-15)
16. entangled.sourceforge.net [↑](#footnote-ref-16)
17. telehash.org [↑](#footnote-ref-17)
18. libtorrent.org [↑](#footnote-ref-18)
19. jinroh.github.io/kadoh [↑](#footnote-ref-19)
20. github.com/bmuller/kademlia [↑](#footnote-ref-20)
21. github.com/sit/dht/wiki [↑](#footnote-ref-21)