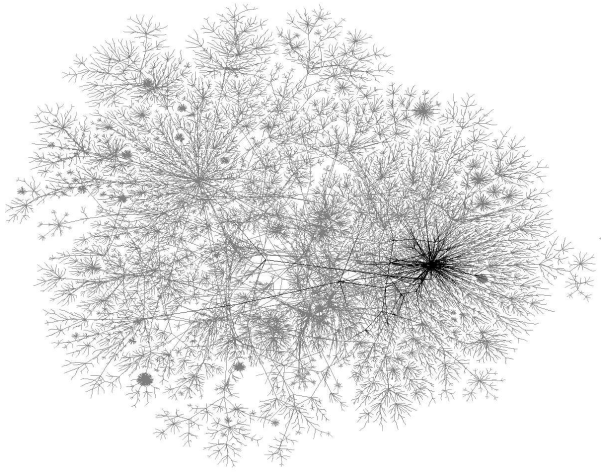
Distributed Web of Trust: Structure and Implementation of Mediated Social Networks

Thomas Barker (0103782)

Supervisor: Jane Sinclair

30th April 2004



"*Every kind of peaceful cooperation among men is primarily based on mutual trust and only secondarily on institutions such as courts of justice and police.*" - Albert Einstein (1875-1955)

Keywords : peer-to-peer, social trust, web-of-trust, instant messaging, networking

Word Count : 16,345

Foreматter

Abstract

Mediated trust is not a new phenomenon; it exists in every large-scale society. Computer technology can transform the way we think about and act on trust.

The current vogue for social networking software is being matched by an increase in academic interest. The dissertation surveys this area, seeking to demonstrate that such systems are workable and can be implemented without raising unresolvable privacy concerns.

To this end a small peer-to-peer application, entitled Hearsay, is developed as a proof of concept. The aim is to provide a guide to the creation of a number of interoperable implementations on the XMPP transport layer.

Preface

Many dissertations focus on technical or mathematical issues; others display a rich body of personal research and analysis. I am attempting to achieve a measure of both, in the hope that placing theory alongside implementation will interest the reader as much as it interested the author.

The first segment of this dissertation is simply background context. The author feels that the most important aspect is not the functioning of such systems, but the ways in which they may be employed.

Aims

- Provide an overview of current progress in social networking systems.
- Prove feasible the concept of a generalised web-of-trust system built around instant messaging technology.
- Defined and justify a workable protocol for such a system.
- Explain how work can be done to improve the technology.

Structure of Work

(1) Background Research

Research of the subject area is necessary, in creating an original piece of work, which accomplishes a worthwhile technical goal.

1. Outline history of how technology has affected social relations.
2. Enumerate the strong motivations for the creation of new social technologies.
3. Learn from existing system, both those on the market and academic research.
4. Draw conclusions which can be used in the creation of Hearsay.

(2) Prototype

By making use of existing technology the project can be broken down into simple do-able steps.

1. Plan simple prototype based on previous research.
2. Analyse behaviour. Ensure that queries always terminate, etc.
3. Define XML assertions and queries.
4. Create and test XMPP Data Transfer Objects.
5. Run basic querying tests over XMPP.

(3) Hearsay system design

Creating a reasoned, justified design, based on effective problem analysis. This design will achieve the goals laid out by the original research.

1. Examine results of the prototype.
2. Analyse the research of existing systems for design principles.
3. Define an outline of the system.
4. Write a functional specification that defines system behaviour.
5. Provide clear guidance on that specifications implementation.

Acknowledgements

Jane Sinclair

For her encouragement, advice and support as a supervisor.

Colm O'Connor, James Forrester and James Reeves

For their advice and encouragement in writing this report.

Andy Chow

For his insightful advice on software and methodology. I am sure that his future research will be of great value.

Jabber Developers and Jive Software

For having produced such an easy and versatile system.

Author's Assessment of the Project

What is the technical contribution of this project? To my knowledge this is the first attempt at implementing an instant messaging based web-of-trust application without a central server. It gives a definite protocol to achieve this, justifing the choice of this solution. The manner of implementation allows the protocol to complement with other systems allowing a large number of uses to be accommodated. This contrasts sharply with existing systems which are difficult to reuse or re-implement.

Why should this contribution be considered either relevant or important to computer science? Trust networks are a huge growth area with enormous implications. They provide the only sensible means to resolve online identities. Identity management is the most pressing enabling technology for online transactions. Pervasive computing technology creates vast amounts of data [location, biometrics, sales receipts] that can only be exploited if authentication can be conducted between people who only indirectly know one another.

Leading public-stock institutions are investing significant resources in this field. Friendster and Huminity became successful start-ups, despite the current recession, by meeting needs in this area. There is a strong business case for web-of-trust in industry, driving demand for sophisticate solutions.

How can others make use of the work in this project? The project provides a protocol definition for the popular XMPP messaging protocol stack. Anyone with a knowledge of XMPP should be able to re-implement the system described here. This protocol will soon be submitted to the Jabber Enhancement Process (JEP), where it could be approved as a Internet Engineering Task Force (IETF) standard.

The source code included in the proof-of-concept would provide a Java programmer with the data structures and communication code required for such a task.

Why should this project be considered an achievement? This project is a genuine achievement because of how easy it tries to make the problem. There is very little to the protocol described, and the source code of Hearsay is reasonably short. The dissertation demonstrates that distributed web-of-trust applications are workable now. It advocates web-of-trust systems as a practical solution to a range of problems, citing prior work and publications to prove its case.

What are the weaknesses of this project? The survey of research in this report is does not cover all the issues in depth. I have tried to fit as much as possible in, but space does not allow a full discussion of all the issues.

Hearsay's implementation is solely a proof-of-concept, it could not be deployed. For such a security critical system, it would be desirable to apply formal methods, or at the very least full test coverage.

Contents

Part I.

Background and Motivation

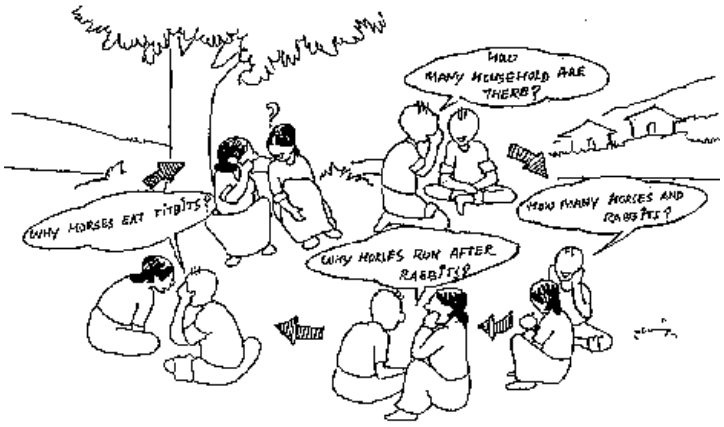
1. Trust in Society

Mediated trust is not a new phenomenon in human society. From Greek olive dealers, to modern investment banks; societies have relied on certificates, marks, signatures and letters of introduction. Successful careers can be been built solely on contacts: The Go-To guy, the fixer, the-man-to-talk-to. Functioning societies bind their members together through a rich tapestry of associations, agreements and loyalty.

Interpersonal dealings have no authentication problems, people can identify each other face to face. You know someone or they know someone you know. Unfortunately face to face meeting do not propagate trust widely or quickly. Large scale societies establish formal social positions, underpinning wider systems that allow strangers to interact without fear of betrayal.

Trust is ultimately an individual trait: our only absolutes in the world are our own dealings. Everything else is hearsay: it lies outside our experience, we have to take a risk

on another's judgement. Through our institutions, we unthinkingly place our property and safety in the hands of people we have never met, solely by relying on hearsay.



The interplay of individuals and groups is dependent on signalling and shared contexts. As the technology changes, society is transformed to avail itself of new opportunities to communicate. The means by which we form our social systems, restricts the forms that can be sustained and made effective.

The constituents of these varied systems though, have stayed fairly constant through history. Family, friends and neighbours: materially overlapping clusters of people are the building blocks for every society or culture. Outside of our own acquaintances we are dealing with hearsay: imagined relationships between ourselves and others whom we do not actually know.

Nationalism was created by the printing press; modern football is made by television. The introduction of the penny-post in nineteenth-century Britain provided labour movements the means to co-ordinate on a national scale, leading directly to the chartists and the Great Reform Act. Each and every large shift in signalling, has been significant because of and through its effects on broader social structures.

In many ways this is already happening: the VISA credit card system is not monolithic corporation, its actually a vast network of franchises co-operating to run a clearing system [Ref. Dee Hock]; email is routine and the cost of airline tickets have plummeted due to better scheduling.

Even money itself is a form of collateralised trust. Only government guarantees make fiat money real. If the government fails or issues too much currency, its value will fall in **exactly** the same as mortgages devalue when the owner becomes unemployed. Large corporations often hold contracts rather than cash as assets, since banks cannot deal with the large amounts of money involved.

Just as the printing press and television shifted our perception of one another in the past, electronic networks must be expected to do the same today.

2. Motivations

Already we broker trust through mechanistic methods. Bureaucratic procedures, emblems, uniforms, etc. The argument for doing so electronically are similar to the arguments for other information processing tasks: speed and convenience.

Control is the reason for doing this within a peer-to-peer system. Personal contact information is sensitive and private. Organisations and individuals guard this information jealously for good reason: it gives them options to act. Market inefficiencies will be created without these options as central repositories will charge fees to facilitate transactions; exploiting the buyer and seller's ignorance of one another.

2.0.1. Social Networking

Sociologists have found that the most economically most important ties are weak ones. Jobs and contracts primarily follow loose social ties: acquaintances, ex co-workers, people who have meet at conferences, etc. Lose ties are more important because they are far more numerous than close ties. Many of these ties do not led to deals being made, but are used indirectly to find other people. Good business sense recognised this long ago; modern firms regard networking an important aspect of professional work.

Most large firms maintain elaborate internal databases of everything from professional associations to old school ties. There is even shrink-wrap software available for this[spoke, six-degrees]. Groups of associated independent business people will posses contact information of similar value. Utilising these contacts digitally without being within the same organization, means having a means to exchange that information in a controlled manner. They are unlikely to unanimously trust any single body to hold all of their information, but they are likely to entrust one another with different pieces of that information. Partial direct disclosures between individuals allows trust groups to overlap one another, naturally diffusing information to those who ought to know it.

Online Personals

It is no longer possible to write anything about social networks without mentioning online dating sites. At one time derided as the realm of hopeless cases, dating sites have begun to flourish, but mainly in metropolitan areas. This is sure to be a large area of early-adoption for any attempt to create new social networking technologies. Personals are particularly interesting because of their reliance on costly signalling.

> 'Recently craigs list made the national news after a girl posted a note about a 'gorgeous guy' she kept seeing at the bus stop on her way to work. The result was that each morning more and more people turned up at the bus stop to see the 'gorgeous guy'. The boy, who was increasingly aware of something strange going on, only became aware of the extent of his local celebrity when a friend of his worked out what was going on and told him.'[3]

Personals sites have a combined user base of over 24 million people. This alone makes them more significant than any of the social networking websites that will later be described.[4] Several social networking sites are trying to enter the personals market, by using the "comfort factor" which indirect ties provide.[11, 12]

2.0.2. Whitelisting and Blacklisting

Spam is a large problem, constituting 40% of all email traffic[5]. Unsolicited messages are sent from almost every point on the internet, often from a 'spam zombie' owned by an unsuspecting broadband user. Traditional blacklisting based on IP numbers is becoming less and less effective. Newer techniques that discard messages using Bayesian filters[6], are working for now, but spammers are finding ways around them.

Instant messaging spam is set to become huge problem.[7] The numbers are rising rapidly and IM networks are struggling to find new ways to deal with it. Email techniques for handling spam, based on Bayesian message analysis, might have problems with the very small messages sent over IM.

Chatbots[1] that try to lure users to certain types of website are active on the ICQ instant messaging network. These chatbot pretend to be real people, on public ICQ channels this is often quite convincing.[2]

[1] Chatbots, or chatter bots, are programs that talk to users on an instant messaging system. They usually perform simple functions, like recording conversations or retrieving quotes.

[2] Personal conversation with Colm O'Connor.

Conversely, it can be a problem to reasonably know when you may acceptably broadcast a message to other users. Elaborate netiquette has developed around cross-posting on the Usenet, but sometimes its very unclear. In addition public forums are increasingly invaded by people hired to talk up products or companies. Better social context will help it clearer when this is appropriate (if ever).

A sufficiently large web-of-trust cache, say 4000 people, can allow individual users to create their own whitelists. In the rare case that a message not on the whitelist is encountered, the system can fall back on existing techniques. It may well be that most people chose only 2000 or so people to be their entire IM network, and this is fine.

Since it is possible to have a relationship with another nym simply by accusing them of spamming, the reverse process could be used to build shared lists of persistent spammers. News of a new spam flood could then ripple through the network automatically, saving most from having to read the spam itself.

The Eternal September

In the eighties being on the internet meant that you were an academic or a student. You would be at a major research campus, probably in America. The main means of communicating was via the Usenet newsgroups, which are still around today. Most people on the internet either knew each other, or had fairly immediate ties through mutual friends.

As a result of this close-knit world, the Usenet was able to act as a medium for small scale bartering, auctions and political activities.

The Usenet worked extremely well for a while, but as the number of people on the internet increased, the system began to fray. Many would pin this point to the September of 1993, sometimes referred to as the 'Eternal September', when America Online began offering access to its members. In these later years, mutual trust on the Usenet broke down.

The amount of spam increase geometrically, and has yet to be brought under control.[8]

Today most of what might have been done on the Usenet is done through website who can control membership, or impose some moderation or rating system.

2.0.3. Filtering News

The Internet has changed the news media dramatically. News is dispensed around the clock from a multitude of different sources. Whilst this has benefits, these benefits are

accompanied by two problems.

The ease of publication has led to a large amount of unchecked and dubious news being published. Determining the accuracy of a source is not a simple task. Since the Internet is a completely open, and often faceless medium, a lie can be told with ease and bias is not always evident. Selecting relevant information quickly has become difficult. While Google has done a very good job of indexing the Internet, its main site only updates its index every four weeks.

If it where possible to establish a system of on-line referrals, then these problems could be mitigated. A item news could be linked back to its source. The source could then be linked back to the user, if only by association with a wire agency. This would allow trustworthy individuals to publish their own news independently of the mainstream media. Such news would travel outwards from that person through their web-of-trust. This process would be a sort of digital rumour, without the loss of verification or the inaccuracies of the old-fashioned rumour.

2.0.4. Transactions

Financial deals could be conducted between individual parties who only indirectly trust one another. These deals could initially paid for by digital 'I-Owe-You' s. If the parties are trusting, such contracts would not necessarily need to be registered for legal purposes.

Over time, the precipitants might accumulate (possibly via deals with 3rd parties) offsetting positions where transactions cancelled each other out. The final settlement of balances could then take place; whereby the net balance is settled in hard currency, services or goods. Settlements could take any mutually convenient form, taking into account jurisdictional tax efficiencies and financial regulation.

2.0.5. Online Auctions

Auction sites, like eBay, are able to charge commission because of their store of past transactions. These histories place eBay in a privileged position over its customers. To a large extent an auction site's profits are rents[3] on its customers' own histories, not payments for services as an auction house. With their own history, a user might chose to contact and pay a seller directly, using a mutually trusted party as the underwriter. Since each additional piece of price information is less and less important than the last in finding a good price, a smaller number of prices on offer would not significantly

[3]Rents are revenues earnt from possession of a piece of property without additional labour.

decrease the usefulness of the system. Providing there are a hundred or so prices, market conditions will still prevail.

2.1. Summary

It can been seen these cases that the motivations fall into three broad categories:

Removing an unneeded central bottleneck. The aim of peer-to-peer systems is to replace single expensive shared server with co-operation between many small clients.

Maintain a single identity across multiple services. Allowing trust built in one field to be used in another. Regardless of where the user last encounter someone, they ought be be able to leverage that history in the future.

Automatically search for people you can trust who have something you need. Many opportunities for useful interactions pass us by because the cost of discovering them was too great. Web of trust systems can help lower the cost of communication, letting the user rely on their own independent resources, rather than depending on external referees or bodies.

Part II.

Research

3. Privacy and Security

This section outlines basic security techniques and terminology which the author researched in [2, 22, 23, 30, 43]. It is intended to give the reader knowledge of terminology used later in the text.

3.0.1. Attacks

Traffic Analysis

The eavesdropper monitors the pattern of message exchange between users. Even if the traffic is encrypted, a lot of information that can be inferred. The fact that two individuals are communicating at all is often confidential. The classic example is the how the press inferred the start of original Gulf War's by the by looking at the rate of pizza delivered to the Pentagon.[22]

Spoofing and Replays

An outside attacker injects messages into the communication system to simulate the existence of a legitimate party. Often prior communication between legitimate users are simply replayed. Even if the messages are encrypted, this can succeed if duplicate communications are allowed and the attacker can deduce the effect of such a duplication.

Malicious Nodes

The attacker might create their own node, which looks like a genuine node to the rest of the network. This node might then be used to troll for particular people or information, and presumably also would be modified to disgorge anything interesting to its creator. Such bad nodes would be particularly dangerous in the case of distributed trust metrics, since a whole web of fake nodes could be created to manipulate the scores.

Root Compromise

Means that a system has been put under the control of an unauthorised remote user. Such a compromise is total. The attacker can perform any function available to the user. Attackers usually install a 'root kit', which can be used to secretly control the compromised machine, such a machine is often referred to as a zombie.

'Rubber-hose Cryptanalysis'

If a user's real-life identity is know, then physical threats can be used against them. This is often euphemistically referred to as 'rubber-hose cryptanalysis'.

3.0.2. Privacy

Confidentiality

The property of a channel that only the intended recipient is able to read a message that is sent to them. An attacker can only intercept the cyphertext, which will be meaningless to them; or they are unable to intercept anything from the channel.

Authenticity

A property of a received message such that the recipient can identify the sender with confidence.

Repudiation and Non- Repudiablity

The denial by one of the users involved in a communication of having participated in all or part of that communication. Non- repudiatable communications, such as those bearing a digital signature, deny the user the ability to do this.

Zero-knowledge

The property of an interaction such that when it is terminated, a piece of information passed from one party to another, without the recipient being able to prove receipt of that information to any 3rd party. These interactions allow non-repudiatable communication.

Perfect Forward Privacy

Of a key-establishment protocol, the condition in which the compromise of a session key or long-term private key after a given session does not cause the compromise of any earlier session.

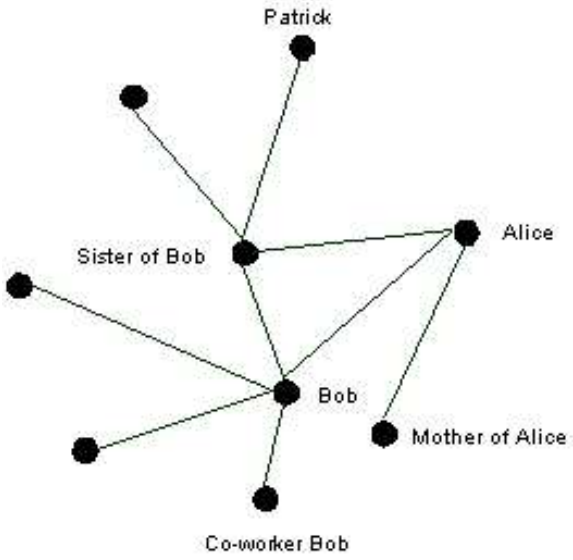
If anyone intercepts your communications traffic, they'd better get to you before you destroy the private keys. Otherwise there's nothing they can do to decode it.

Nym

A nym (pronounced NIHM and a shortened form of "pseudonym,") is a name invented by or provided for an Internet user in order to conceal the user's real identity and, in some cases, to expressly create a new and separate Internet identity.

Web-of-trust

A basic technique for resolving trust. Each user creates an accessible list of people they trust. Other users then use these lists to form a graph between users.



By finding paths between users, the relative trust between them can be calculated in accordance with some trust metric.

3.1. Cryptography

This is simply a description of basic methods and their applications. The reader will have to assume that they work as described. A fantastic introduction to the inner-workings of cryptography is given by Applied Cryptography[2].

Symmetric Encryption

The most common form of encryption. Given a key, the algorithm takes plaintext and converts it to ciphertext. Given the same key, it also converts ciphertext back into plaintext.

$C = f(K, P)$

$P = f(K, C)$

where C is the ciphertext, P is the plaintext, K is the key

The problem is distributing the keys, after all, the channel must be insecure, or encryption would not be needed. So users must first communicate out-of-band to exchange keys. This is generally infeasible, or just too dangerous.

Public-key Encryption

In a public key system, each user has two keys: a public key and a private key, which must be generated together. It is computationally infeasible to deduce the private key from the public key. The public key really is public, should be distributed widely. The private key, on the other hand, is never shared, not even with someone the user wishes to communicate with.

Anyone with a copy of your public key can then encrypt information that only you can read. Even people you have never met. Anyone who has a public key can encrypt information but cannot decrypt it. Only the person who has the corresponding private key can decrypt the information. The need to share secret keys via some secure channel is eliminated; all communications involve only public keys, and no private key is ever transmitted

Cryptographic Hashes

These hash functions compute a short digest of an unlimited-length original message. These functions have the unusual property that changing any single bit of the original message changes, on average, half of the bits of the digest. It is computationally easy to verify that a particular message does, in fact, hash to a particular value, even though it is infeasible to find a message which produces some particular hash.

Digital Signatures

Since cryptographic hashes are compact yet give an unambiguous indication of whether the original message has been altered, they are often used to implement digital signatures. A hash is created and then encrypted with the sender's private key. The receiver can then decrypt this encrypted hash using the senders public key. If the signature is valid, the decrypted hash will match the hash calculated from the message plaintext.

4. Commercial Services

These services have only existed 'in the wild' for a few years, so no independent studies are available in the academic literature. The author therefore derived much of this material from directly observing and using these services.

Despite the inevitably subjective nature of this research, it was extremely valuable in designing Hearsay. Hopefully, there will be some academic investigation of these services in the future.

Huminity

Huminity is social networking software that allows users to graphically navigate animated maps of connections and view the links between friends and friends-of-friends. It also builds in instant messaging capabilities for chatting with other Huminity users. All the data is stored on a central server.[9]

The system requires five email addresses from every user who joins, which has helped it to grow fast. Their software is very slick and easy to use, although this does mean that users have to download a client.[9]

Huminity have a frequently updated corporate weblog, which reports on developments in the industry. The most recently added features are a search engine of their members, which publishes interactive graphs of them using Java applets. The engine also brings up various 'spam member', ex. "Hi! I'm Heidi! I'm 18 years old and just bought my new webcam."

Microsoft: The Not-So Sleeping Giant

So far Microsoft have not carried out any major product rollouts. However, the MSN Messenger team has described social network as an natural extension of the MSN buddy list. In the meantime, Microsoft have produced a new invite only service named 'Wallop'.

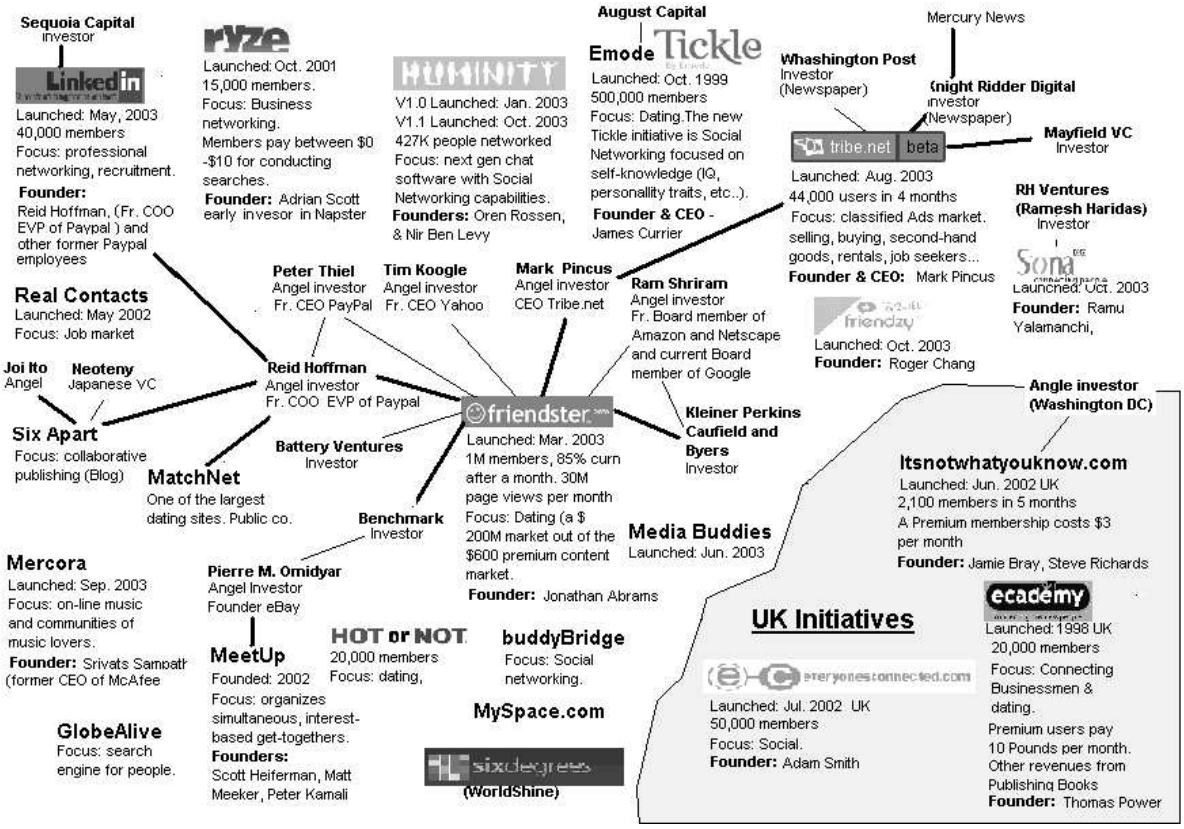


Figure 4.1.: Significant Social Networking Sites [Ref. Huminity blog]

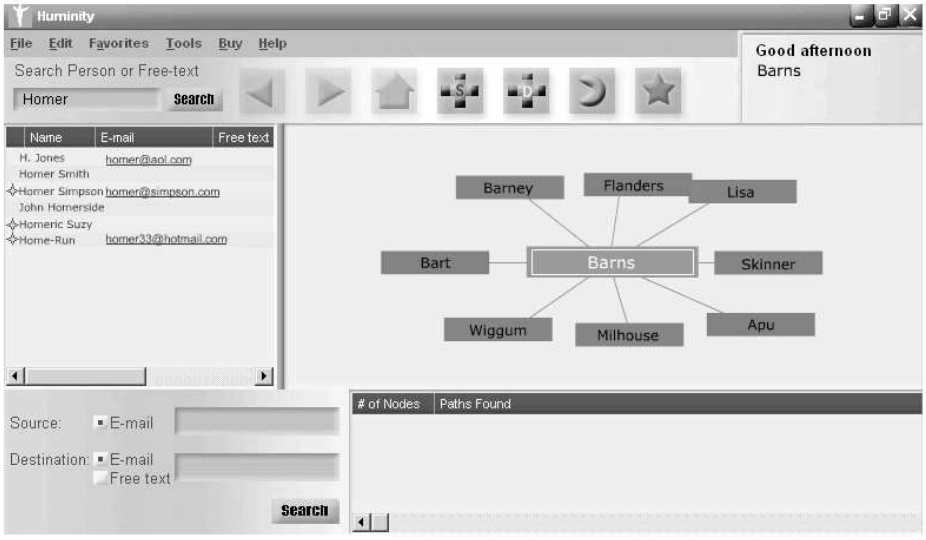


Figure 4.2.: Huminity Screenshot

Figure 4.3.: Wallop Screenshot

Currently restricted to 150 users, eventually this service will be rolled out to all of Microsoft's MSN customers. Details about this service are very scarce, so it is unclear exactly what functionality it will offer.[10]

Commercially Run Web-based Systems

There are many websites offering to hook people up to each other. They usually focus on a theme, or a particular geographic area. This seems to be an attempt to avoid fragmentation. Since the number of eyeballs [active users] is very large, competition is fierce and specialisation seems to be the natural response.[28] Unfortunately this means that these sites rarely seem bring different sorts of people together. They largely act as lumps of like minded individuals.

Arguably this is a feature rather than a flaw for sites that revolve around informal socialisation. Nevertheless, it precludes the overlaps that occur between difference social groups to be given their full significance. These connecting people are vital to our real-world lives, it seems a pity not to use them our systems.

Most Famous: Friendster

This site has had several New York Times articles and comedy video made about it.It has 2.8 million users and an estimated value in 2003 of $54million.[11] Despite this it

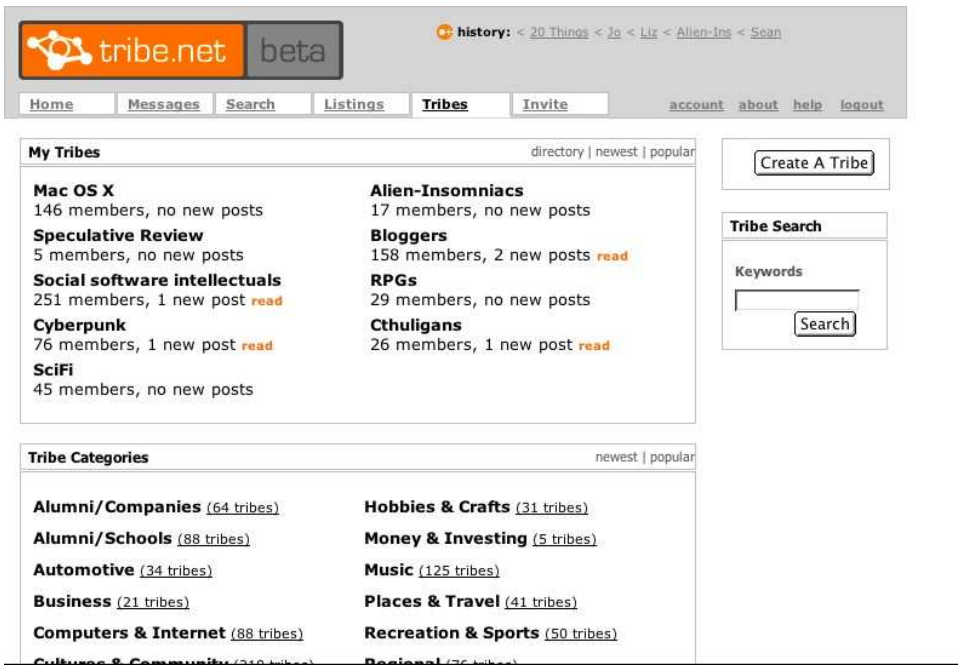


Figure 4.4.: Tribe.net Homepage

does seem to be having problems recently. The site is starting to stall under the load, frustrating users.

Friendster's friendship system is loose and undifferentiated, the site is mainly aimed at dating. As a result, most people have a large number of "friends" who they scarcely know.[12]

To be fair to Friendster, they are very popular and have a very simple, intuitive interface. The really site revolves around personals, since people are more comfortable meeting not-quite strangers. This aspect is very well supported.

It remains to be seen if Friendster's current popularity will hold when the service begins charging users this year.

Cutest: Tribe.net

Tribe.net takes a more indirect approach to encouraging links. Rather than having everyone linked in a one-to-one mesh, tribe.net is organised around tribes. There are several thousand tribes, covering every possible topic. Some tribes are open to the public, others require an invite.[13] People can link directly to one another, but the existence of the tribes seems to limit this to sensible boundaries [6-12 friends].

Tribe.net offers area-based small ads and personals[13], although it interprets UK

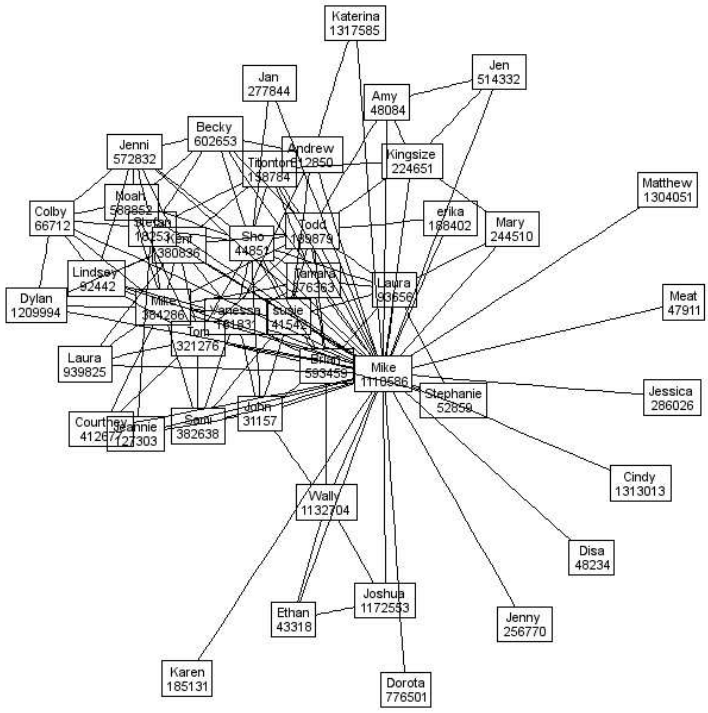


Figure 4.5.: Livejournal Friends Cluster

postcodes as being near San Jose. There's seems to bit bits of spam creeping in, and long abandoned tribes litter the listings, although this isn't a real distraction.

Best kept secret: Livejournal.com

With 2.3 million users, 1.6 million of whom are regularly active, Livejournal.com is one of the webs most active sites. Livejournal was never promoted as a social networking system. It started life 5 years ago as a weblog[1], over time simple interest groups and personal friend systems were added.[14]

Livejournal works well because people use their weblogs to stay in touch with their friends. This leads their friends to join and start their own weblogs. So despite being a weblog site, Livejournal actually has one of the largest social network databases on the internet. Currently, Livejournal is looking to add FOAF (explained below) capability to its weblogs.[14]

Livejournal is mainly run by volunteers, with a small commercial staff selling premium accounts to pay for server costs. The lack of advertising and funding, probably

[1]Simple self published website consisting of a series of text entries, ordered by date.

explains how such a large site could be so often over looked by the media.

5. Semantic Web

> *'The web is more a social creation than a technical one. I designed it for a social effect - to help people work together - and not as a technical toy. The ultimate goal of the Web is to support and improve our weblike existence in the world. We clump into families, associations, and companies. We develop trust across the miles and distrust around the corner.'*
>
> -Tim Berners-Lee, Weaving The Web

There are two current initiatives on the way that use semantic web methods[15] in trust systems. Firstly individual users place their relationship data on a publicly accessible web server. Each person referenced is linked to by pointing at their relationship data, on their website.

When enough people have done this, web spiders can crawl the web to build relationship graphs. Although this is a viable approach, that seems to be gathering momentum, the privacy issues raised would limit its use.

5.1. Friend-Of-A-Friend (FOAF)

A RDF[1]-based semantic web standard.[17] There is an increasing range of tools[29] for working with the format, but there are complaints about how complex it is to set-up.

FOAF is catching on with commercial websites because of it flexibility. Both Livejournal and Tribe.net are planning to roll-out FOAF web interfaces soon.[13, 14] This would bring the number of FOAF users up to 1.5+ million, a very considerable number.

[1] Resource Description Framework provides a lightweight ontology system to support the exchange of data over the web.

```
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:foaf="http://xmlns.com/foaf/0.1/">
     <foaf:Person>
         <foaf:name>Chris Heathcote</foaf:name>
         <foaf:nick>ChrisDodo</foaf:nick>
         <foaf:firstName>Chris</foaf:firstName>
         <foaf:surname>Heathcote</foaf:surname>
         <foaf:weblog rdf:resource="http://www.undergroundlondon.com/antimega/"/>
         <foaf:homepage rdf:resource="http://www.undergroundlondon.com/antimega/"/>
         <foaf:mbox_sha1sum>813acdde23e4d86cb6f40934bdfab8a25032bc97</foaf:mbox_sha1sum>
         <foaf:mbox_sha1sum>7eb4defabb2d4cda4d87c7670fdf9c290e68864f</foaf:mbox_sha1sum>
         <foaf:workplaceHomepage rdf:resource="http://www.orange.com"/>
         <foaf:knows>
             <foaf:Person>
                 <rdfs:seeAlso rdf:resource="http://downlode.org/foaf/foaf.rdf"/>
                 <foaf:surname>Martin</foaf:surname>
                 <foaf:mbox_sha1sum>8699ba79a95abf86e0055c133bf5d87ceab921e9</foaf:mbox_sha1sum>
                 <foaf:firstName>Earle</foaf:firstName>
             </foaf:Person>
         </foaf:knows>
         <foaf:knows>
             <foaf:Person>
                 <rdfs:seeAlso rdf:resource="http://husk.org/misc/blech.xml"/>
                 <foaf:surname>Mison</foaf:surname>
                 <foaf:mbox_sha1sum>af4f9b76ee3dceefd5bcbc0d1485541fd6078005</foaf:mbox_sha1sum>
                 <foaf:nick>blech</foaf:nick>
                 <foaf:firstName>Paul</foaf:firstName>
            </foaf:Person>
        </foaf:knows>
</rdf:RDF>
```

Figure 5.1.: FOAF XML

```
<a href="http://jane-blog.example.org/" rel="sweetheart date met">Jane</a>
<a href="http://dave-blog.example.org/" rel="friend met">Dave</a>
<a href="http://darryl-blog.example.org/" rel="friend met">Darryl</a>
<a href="http://www.metafilter.com/">MetaFilter</a>
<a href="http://james-blog.example.com/" rel="met">James Expert</a>
```

Figure 5.2.: XFN Example

5.2. XFN

Extended friends network (XNF) is a simple extension of the anchor tag: a 'rel' attribute in a web link to a friend's site, with the value of 'rel' is a list of web markers.

XFN seems to be spreading very rapidly through weblog sites, since it is so simple to implement. It does not as yet have the same level of adaptation as FOAF, but this might reverse as these systems make their way out into the general internet population.

XFN does not just list friends, it allows you to specify in detail how and why you are associated with someone. This is an innovation over most other systems, which seem to ignore context altogether.[16]

5.3. Search Engines

Several interesting attempts are being made to build FOAF and XFN search engines. By entering an email address or homepage URL, a whole web of information can be

retrieved.[19, 18] These search hubs could be seen as an attempt to reverse the specialisation that has occured on the web-based systems.

6. X.509 Certificates

Most web users have made use of X.509 certificates, since they are used to verify SSL[1] connections.[20] These certificates are available from a number of vendors, such as Verisign, and cost around £40. A certificate is a public-key signed by a vendor. The vendor acts as a certifying authority [CA].

A certificate authority issues a certificate, binding a public key to a particular name. This name is supposed to be the 'Distinguished Name' defined by X.500. As no real implementation of this standard exists, the binding is usually to an e-mail address or DNS[2]-entry.[20]

Certificate authorities exist in a hierarchical tree. The top node is imaginary, so in practice all certificates derive from the second level root certificates. Each certificate is distributed with a chain of signatures that link it to a root certificate. This means that trust can only flow downwards, the is no scope for conflicts or ambiguity.[22]

Root certificates can be issued to all employees by an organisation so that all employees can use the company X.509 system. Browsers come with root certificates pre-installed, so SSL certificates from larger vendors who have paid for the privilege of being pre-installed will work instantly; in essence the browser's programmers determine which certificate authorities are trusted third parties. X.509 also includes standards for Certificate Revocation List implementations.

X.509 works well as a system. It is widely deployed, and a very good solution to work with if only a signing hierarchy is required.[22]

[1] Secure Socket Layer, used by Secure HTTP.
[2] The Domain Name System translates human-readable URI addresses to IP machine addresses.

7. Pretty Good Privacy (PGP)

PGP has famous history, not only was it one of the first widely distributed public-key encryption packages, its author was at one point prosecuted by the US Government over it.[21] It was the first system to adopt a distributed trust model without a central certifying authority. PGP is primarily used to encrypt and digitally sign email, but is often used to sign files for software distribution.[22, 23]

The PGP design was sufficiently influential that it has been made an IETF Internet standard (OpenPGP). Used properly, PGP is capable of very high security; most informed observers believe that even government agencies such as the NSA are incapable of cryptographically breaking properly produced PGP messages.[22]

PGP uses a certificate 'vetting scheme'. PGP certificates are 'digitally signed' by other PGP users who, by that act, are 'endorsing' the certificates authenticity. PGP uses a 'vote counting' scheme to determine which certificates a user should be trust. The parameters are user adjustable, and can be completely bypassed if a PGP user wishes. Unlike most public key infrastructure designs, the scheme is flexible, leaving trust decisions to the individual user.

Since PGP uses a passive web-of-trust, every entry must be added by hand, its webs rarely grow to a very large size without there being some specific purpose. Because of this public key-servers have been established. They behave similarly to a certificate authority, except that they normally only claim to certify that a user has a certain email address.

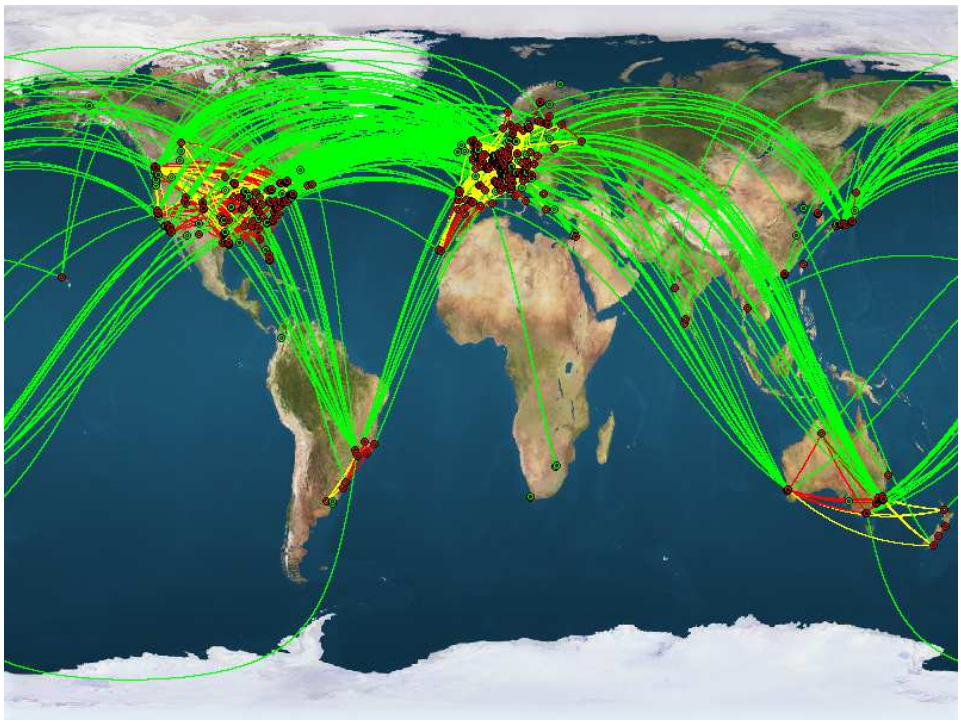


Figure 7.1.: Debian World PGP Keyring

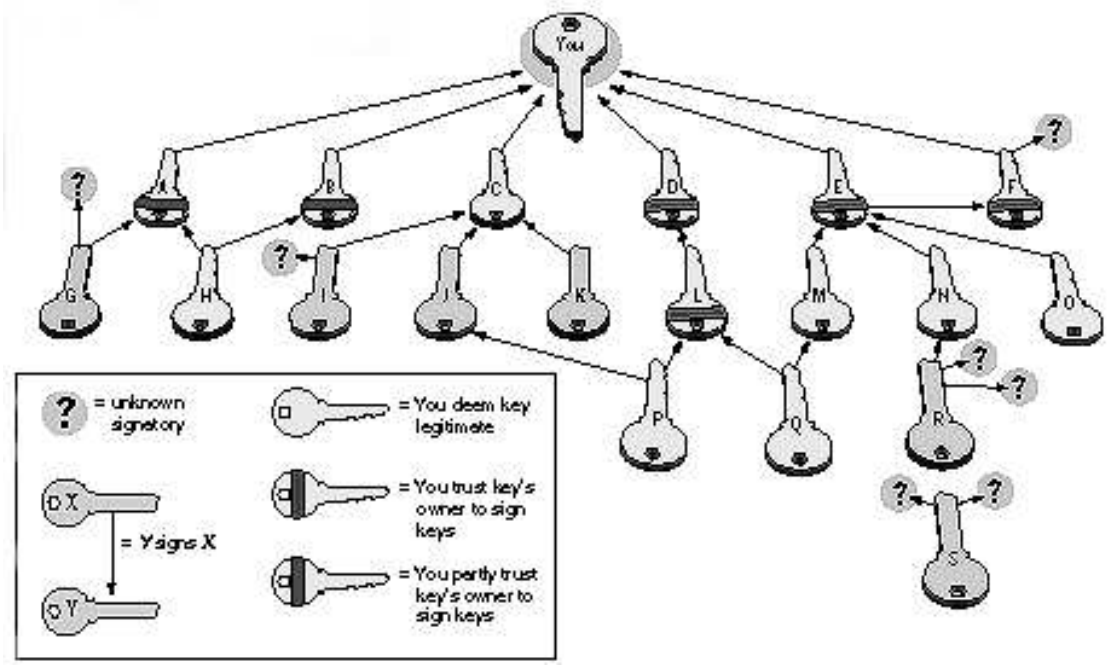


Figure 7.2.: PGP Keyring Scheme [Ref. BYTE Magazine]

8. Academics Systems

8.1. Yenta

The Yenta Multi-Agent Distributed Matchmaking System is the doctorate thesis of Leonard Newton Foner. It is a multi-agent architecture designed to use both strong cryptography and decentralization to enable a broad class of applications which handle personal information securely. Yenta aims to show that complex communication systems can be created without compromising personal privacy.

Yenta uses large numbers of agents who form coalitions sharing common interests, enabling conversations between their users. It does so with a high degree of privacy, security and robustness, without requiring its users to place exclusive trust a single entity. The information for determining each user's interests is taken from keyword analysis of the files on their computer.[24]

All of Yenta's traffic is encrypted and sent over UDP. The system handles discovery at the network level, broadcasting packets and relying on a critical mass of users being installed on the same subnet. When a Yenta agent finds another, they exchange information about other agents. Over time agents are able to seek out others with similar interests and send messages to them. Since agents can relay messages on each others behalf, there is plausible deniability of having taken part in any given cluster.[24]

Yenta was intended to serve as an example for other systems architects who are designing systems that handle personal information. The thesis also deals with gather

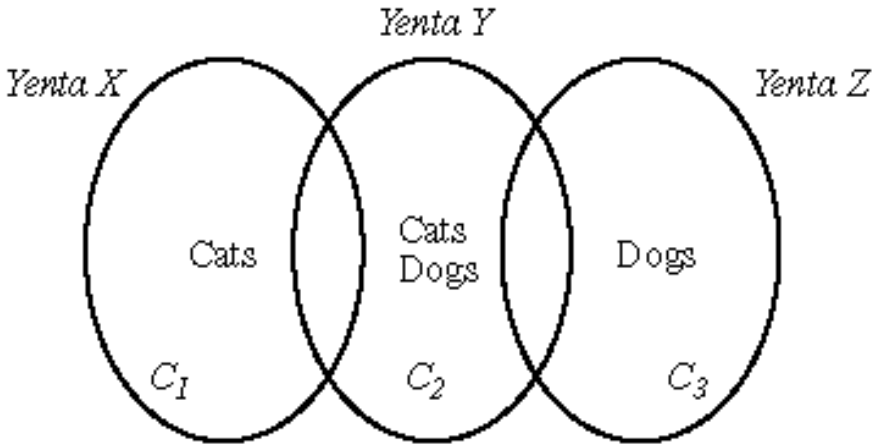


Figure 8.1.: Yenta Discovery

anonymous usage data from such networks, and includes a system for public reviews of software.

8.2. Agent Amplified Communication (AAC)

A scheme for finding experts developed at Bell Laboratories. Each user in the system compiles a list of knowledgeable experts they are willing to reference, with the areas of expertise being represented by keywords.

AAC operates by broadcasting emails. When a user queries the system, specially marked emails are sent out to the closest matches that can be found on the users own referral list. A software agent on the recipient's computer, intercepts those messages and reads them. If the agent finds a good enough reference on the recipient's reference list, it will alert the recipient.[25]

The recipient must then approve the recommendation before it is sent directly back to the original requester. If a good enough match is not found, the software agent silently forwards the request to the closest matches on its own referral list.

In this way, AAC cascades messages from one user to the next until a good referral is found or a preset limit is reached. Research ACC systems has showed that the number of messages needed to find a match increased with the accuracy required and decreased for longer referral lists.[25]

8.3. Avocado

The Avocado website was one of the first online web-of-trust systems. Advogato was built by Raph Levien, at Berkeley as part of his PhD. research into web-of-trust metrics.[27, 26] It operates a sort of high-end Slashdot[1][27], for people involved in open-source projects.

It exists to certify the competency of open-source developers. The site is open to all, but to receive a certification [observer, apprentice, journeyman, master] you must be rated by an existing user. These ratings are then aggregated and transformed into a single public ranking on the system. The system also runs a stories section, a listing of projects and user diaries.[26]

The trust metric used in Advogato is interesting, since it uses correlation between different people's ratings as a guide to how trustworthy they are. This means that a few malicious nodes cannot undermine the system. So far the system is working well, and Advogato rankings are generally held to be accurate.[27]

[1] An very popular internet discussion board frequented by OpenSource enthusiasts. Known for being one of the first to introduce a user moderation system.

9. Research Summary

Commercial systems are overcentralised, placing total trust in a single body. Their applications are limited by their initial design, and it is impossible for third parties to add functionality. Users incur costs (if only in terms of time) when they sign-up for a networking website. Since there are so many sites, but only a finite number of users, it likely that most of these sites will die off leaving a few of the larger ones behind. Otherwise, users would have to sign-up to multiple sites, or have a very small network on the single site they used.

That having been said, commercially operated sites provide a wide range of value added services, have good privacy policies and generally take security reasonably seriously. The presence of an administrator tends to impose some vague standard of behaviour; probably even more important will be the imposition of service fees as a form of costly signalling.

The academic systems also serve their purposes well. Yenta is particularly noteworthy for addressing privacy issues in detail and solving them convincingly. Yenta's use of physical network level discovery is very innovative, and should allow it to be used ad-hoc way over wireless. Advogato has succeeded admirably in creating a generally accepted ranking system for open-source developers.

The new semantic web systems seem workable, and some very nice tools have been produced to spider them.[29] However, since these systems entail placing all the relationship data on the internet, they do not allow for selective disclosure of information. This limits their usefulness to handling non-confidential information.

Popular services all seem to suffer problems with 'friendship inflation'. Costless virtual friendships tend to accumulate in the system, leading to people who have hundreds of friends, most of whom they have never met.[28] Orkut[1] has tried suspending the ac-

[1] Google's social networking website. To join requires an invitation from an existing member. The author is currently insufficiently social to have met any Orkut members.

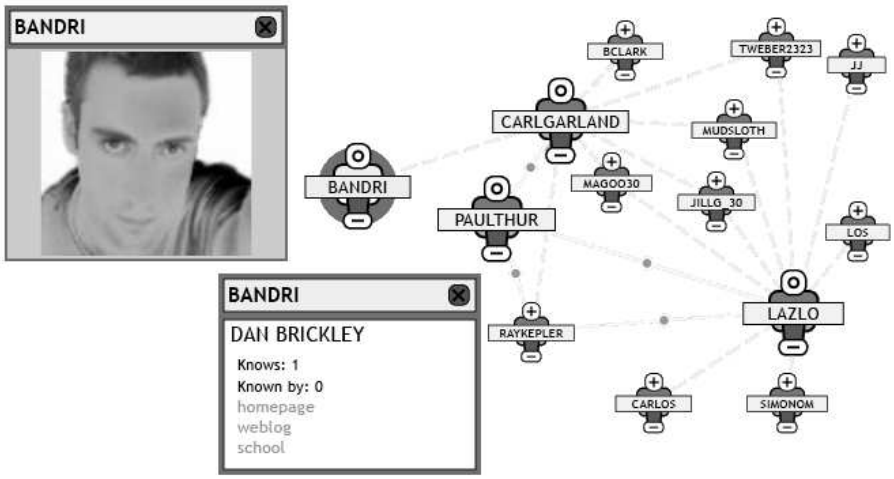


Figure 9.1.: FOAF Navigator SVG Web Application

counts of users who have too many 'friends'. It seems as if friend is a term that can only be used in context, on the internet that context does not carry over the way that it should. Personals sites have sometimes been seen as being popular, simply because they allow people to project their fantasies onto each other. Maybe there is another form of 'Eternal September' where users are too polite, too close, too undiscriminating, an Eternal April.

Spam, the dry rot of the internet, is also making an appearance on popular sites. Hopefully, the trend towards paid accounts will dampen the flood, and the techniques used for email[6] can be applied to these sites as well. Spam is an issue that will affect any service that allows open access.

None of these systems provide a general mechanism or platform. Hearsay should avoid this by using an existing messaging infrastructure. Hearsay will be a protocol, not a specific application. It must be interoperable with any application that the user might wish to use.

Lesson for Hearsay:

Add the social networking tool to the applications.
Not another application to the social networking tool.
Leave all user data with the user.

Part III.

Hearsay Prototype

10. Conceptual

10.1. Approach

The conceptual design of Hearsay works from a firm set of principles, derived from the research, through to a functional specification and a guide to its implementation. Each stage follows from the previous one.

It is a common problem in software design that pieces of structure are described ad-hoc in an attempt to document a previously created system. This conceptual design can be read as a coherent guide to the implemetation of a Hearsay system.

10.2. Principles

These principles will guide the design of both the Hearsay protocol and the proof-of-concept. These principles are similar to those used in many other projects[22, 24, 25], and so could be thought of as emerging best practices.

(1) Trust No-one The only meaningful origin for a trust network is the user's own identity. Only first hand information is inherently trustworthy. There is no value for a nym to make assertions on behalf of another; the only value is in assertions made by that nym about another. It should also be assumed that any external system can fail or make erroneous responses.

(2) They are out to get you

Privacy should be safeguarded at all times. It is too late to implement security after a compromise. Automated attacks on address and customer databases already happen regularly[30], they will inevitability happen to any web-of-trust system.

(3) Make no assumptions

As we have seen, there are an enormous number of ways in which people define trust. They vary hugely between people and cultures, any attempt to set this in stone, or reduce it to a number will just make the system useless for other applications. Nyms themselves can be anything that can be connected to the internet. A organisation could be considered a nym; could a chatbot; could your dog?[1] Who is to decide? The user, no one else.

(4) It's an X problem

The system cannot and should not attempt to solve every problem. The system may require highly specific services to be available to it. This does not mean there is any value in the system implementing its own transport layer, encryption and so forth. Indeed doing so increases the chances of making mistakes which might comprimise the safety of the design.

(5) One user, equals one machine

Security is a function of the entire system, even good system design is worthless if it can be compromised by bribing or threatening someone. A decentralized architecture avoids having a single point of trust, or single point resource that will bottleneck as the system expands. For an individual machine to be compromised is unfortunate, but it should

[1] Friendster **did** decide that the customers dogs' were not legitimate users. The customers were unhappy.[12]

not affect people unrelated to the user. Every user ought to be able to run the system in an environment that they feel is secure.

(6) Minimise collected information

The surest way to protect private data collected from others is never to collect it in the first place. Legal manoeuvres and threats aimed at personal data are very frequent. It is simple unbelievable what some people will send through email, even when they know it is being monitored.

10.3. Scoping

In keeping with principle 4, the following areas will not be addressed by Hearsay:

- Transport Layer Encryption
- Obscufication of Traffic Patterns
- Authentication of Users
- Service Discovery
- Ensuring a trustworthy execution environment
- Protecting information in physical storage

Therefore Hearsay will require an implementation platform which provides these facilities.

Hearsay will attempt to perform the following functions, in accordance with the afore mentioned principles:

- Define a simple structure for social networking data
- Storage of such social data
- Publishing such data to a given transport
- Determining the privileges of an authenticated user
- Simple processing of this data for use

10.4. Structural Overview

"Now we're getting sophisticated enough that we can talk about the engineering of software, as opposed to throwing a bunch of hackers in a room and hoping that on Monday morning something comes out."

- Michael Bennett, University of Western Ontario

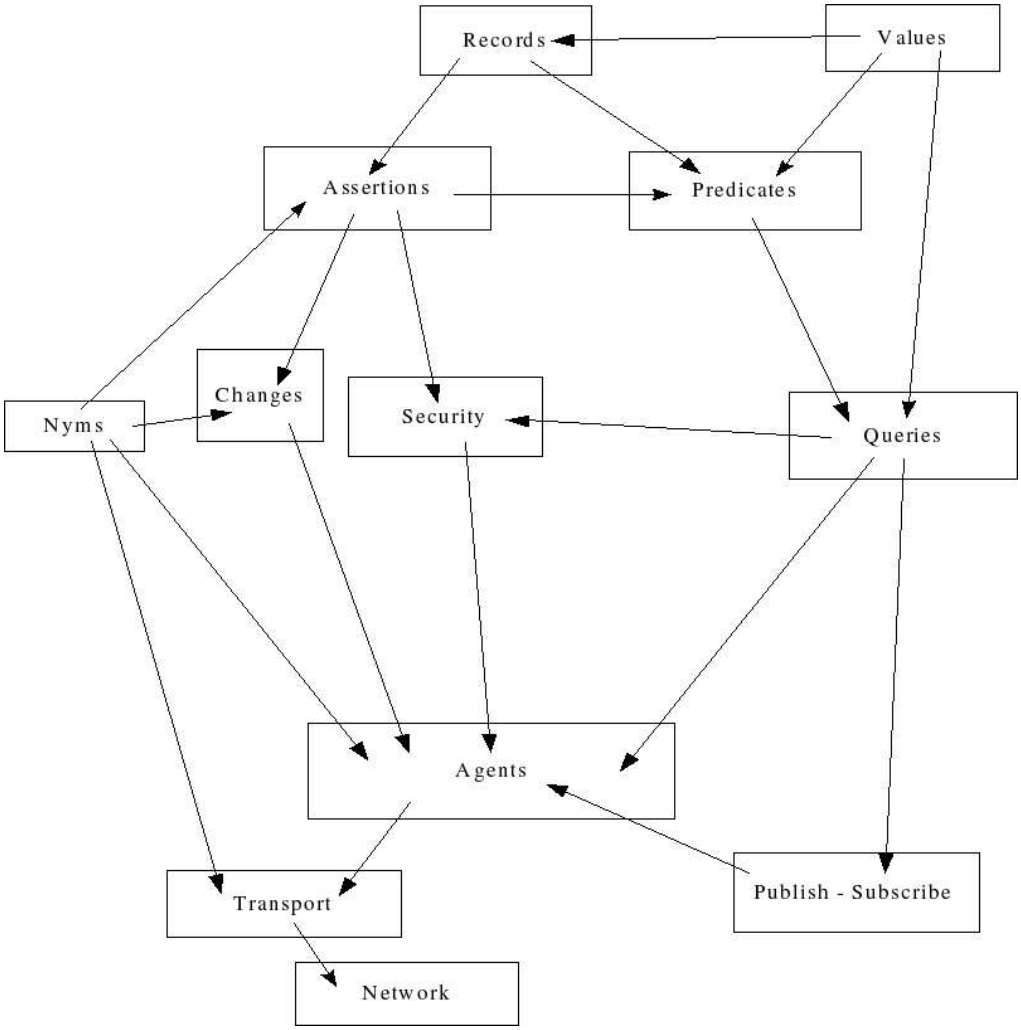


Figure 10.1.: Structural Overview of a Hearsay System

Data Model The ***values*** form ***records*** which are linked with ***nyms*** to form ***assertions***. ***Predicates*** are to select sets of ***assertions***. The user is represented by their own ***nym***, which is an invariant of their ***agent***.

Access ***Queries*** are defined by ***predicates*** and made by ***nyms***, in order to retrieve ***assertion*** sets. ***Changes*** alter the assetion set stored by ***agents***.

Security Policies are set by ***security*** that determine the data that can be retrieved or altered by remote *nyms*.

Transport Layer Communication of one ***agent*** with another, is entirely managed by the ***agents*** themselves. The ***transport*** provides the message delivery system through which they communicate.

10.5. Dependencies and Interoperability

In terms of external dependencies and interoperability we can seperate Hearsay's design into three clear tiers.

Transport Dependent

- Query-Response Message Format
- Publish-Subscribe Message Format

All Hearsay implementations sending and receiving the same the format of the messages over the same transport layer, will be able to share information and interoperate. They will not be operatable by their **owner** nym in the same manner.

Deployment Dependent

- All of the above
- Add assertion signalling
- Remove asserrtion signalling
- Same method of managing permissions

Equivelence at this level will allow Hearsay implementations to be used with the same user training and tools.

Implementation Dependent

- All of the above
- Any persistent data formats
- Setting and installation mechanisms.

Such implementations have identical behaviou and may be freely interchanged.

11. Functional Specification

'The more interesting your types get, the less fun it is to write them down!'

- Pierce

The functional specification is annotated with Standard Modelling Language types and examples. The Standard Modelling Language was developed at Edinbough University for use in a logic analyser for proving and specifying computer programs. Since then it has been developed into a standard functional programming language.[31]

SML was to preferable to other formal notations such as the Z specification language or the B method. Although these notations have well developed bodies of proof-technique, they are unintelligible to the untrained. SML is reasonably clear when presented in context, since it deals with basic types [int, strings, float, etc.] familiar to programmers. UML also has these properties, but it tends to require a great deal of space and to over-specify the structure of programs. Although a fine "blue-print" language, it is a clumsy means for defining program behaviour.

11.1. Relationships

11.1.1. Nyms

To handle social networks, it is necessary to represents them as a convenient abstract data structure. There needs to be a model that accurately mimics real-life, while being convenient for manipulation and processing. A social network defines relationships between given nyms. These nyms are given, it cannot be determined who or what they represent.

```
type nym
```

At its most basic level any relationship is an assertion made by one person about another. For example, 'this person is my friend', or 'I worked with Pete at Waitrose in June'. These relationship are not necessarily reciprocal or transitive.[1] As such they may be represented in the following form.

```
type direction = nym * nym
```

Where the domain of *direction* is the asserter of the relationship, and the *range* is the subject that direction.

11.1.2. Data

Unwarranted assumptions cannot be made as to the nature and content of such relationships. Therefore additional data must be present in a valid assertion.

```
type data = ...
```

The form of this data cannot be predetermined. The schema used must necessarily be flexible, capable if representing a simple general data structure.

It is now necessary to define a membership of *VALUE* which will represent this structures basic elements, such that most reasonable uses of Hearsay are supported. Based on the prior research the following members where chosen.

Basic Types

- Booleans
- Integers
- Floating Point Numbers

Text Strings Can be used as human-readable flags, or to pass unformatted natural language information.

Ex.

- Comment on a skill-set
- Description of an item

[1] Although many existing systems treat them as such.[12, 11, 13, 28]

Time Period The duration between two moments in time, possibly unbounded into the past or future. A single moment can be identified by making both moments identical, but most relationships relate to a period of time. The standard format for defining an interval of time is ISO standard number 8601.[33]

Location Information To enable location sensitive services requires locations data. There are many different forms of geolocation data, GPS data being the defacto standard. GPS is a pair of degrees, designating a particular place. It is also useful to allow distances as values, especially is Hearsay interoperates with location sensing devices.

```
signature Location =
    sig
        type degree

        type position = degree * latitude
        type location = position * string option

        datatype distance = meters of int
        val distanceBetween: position * position -> distance
    end
```

Giving the full value schema.

```
signature Values =
    sig
        include Location
        include TimePeriod

        datatype value = int      |   real   | bool
                       | location | distance | period
    end
```

A mapping from simple text labels to values is appropriate as the enclosing structure.

```
type field  = string * value
type data   = field set
```

11.1.3. Namespaces and Records

It can be the case that instances of *data* while having identical data, do in fact refer to different real-world facts.

For example:

```
val ambigious = { ('place', 'paris'), ('year', 1983) };
```

This data might be interpreted as

- Duration of a flat share
- A period and area of employement
- Holiday plans

Clearly this is non-sensical. These interpretations are entirely unrelated. The solution to this is to qualify our records with idenifiers. By qualifying each record with a unique resource identifier (URI), accidental collisions can be avoided.

```
signature Record =
    sig
        include Values

        type uri
        type field = string * value
        type data = field set
        type record = uri * data
    end
```

Such a unique namespace identifier should be a reference to its definition. This means that third-parties can add their own record definitions to the system without worrying about collisions, in much the same manner as XML.[2]

[2]XML stands for extensible markup language.

```
val UnAmbigious =
    (
        'http://example.org/holiday',
        {
            ('place', 'paris'),
            ('year', 1983)
        }
    );
```

We can now represent a assertion by the following schema.

```
type assertion = direction * record
```

So the social network as a whole will be represented by the schema.

```
signature Relationships =
    sig
        include Record
        type nym

        type direction = nym * nym
        type assertion = direction * record
        type socialnetwork = assertion set
    end
```

11.2. Predicates and Selection

A method required to define desired selections of assertions.

```
val isA: ( predicate * assertion ) -> bool
```

e.g.

```
  equals('firstName', 'Pete')
AND
  (
    less('dateOfBirth', 1981-03-19)
   OR
    equal('dateOfBirth', 1981-03-19)
  )
```

The predicate statements made of values must be matched to fields within a record.

```
type valuePredicate = value -> bool
```

These assertion predicates must be qualified the namespace of the records they are applicable to.

```
val composePredicate: ((string * valuePredicate) * uri) set -> predicate
```

This form is most easily expressed with conjunctive normal form. A statement is in conjunctive normal form if it is a conjunction (sequence of ANDs) consisting of one or more conjuncts, each of which is a disjunction (OR) of one or more literals (i.e., statement letters and negations of statement letters. So a statement in conjunctive normal form would be represented as:

```
type predicate = predicate of (assertion -> bool) set set
```

Any predicate can be expressed in conjunctive normal form.

```
type isA: ( predicate * assertion ) -> bool
```

In this form their truthfulness can be easily checked through predicate resolution.

All those assertion which do not match the predicate specified in the query can then be filtered out. It is assumed that the nym fields of an assertion contain no information, these being arbitary mappings to a transport-end point.

```
signature Logic =
    sig
        include Relationships

        type valuePredicate = value -> bool
        val composePredicate:
                uri * ((string * valuePredicate) set) -> predicate

        type predicate = (assertion -> bool) set set

        val isA: ( predicate * assertion ) -> bool
        val negation: bool -> bool
    end
```

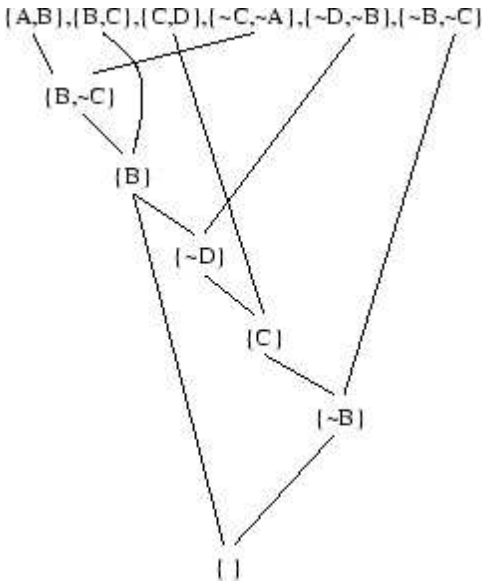


Figure 11.1.: Predicate Resolution

These structures will be of the same form as:

```
{
    {
        ('http://rats.ac', {('colour', equals('red'))}),
        ('http://hamsters.ac',{('size', greaterThan(9))})
    },
    {
       ('http://gerbil.ac', {'hair', equals('long')})
    },
}
```

11.3. Querying

In the conceptual model it is only valid for a nym to publish assertion made by itself. It cannot represent anyone else. The query interface to an individual nym is therefore a predicate which selects from the set of available assertions published by that nym.

```
datatype query = get of predicate
```

The result of such a query will then be a set of assertions, or an instance of error.

```
datatype result = Success of assertion set | Failure of error
```

The form of these errors is specified in the Transport section of this specification.

11.4. Network Transversal and Simplification

A general model cannot simply assign numerical trust weightings to people. The system will need to differentiate between different types of relationship. Any trust metrics must be calculated as aggregate summaries of primitive relationships.

The social network graph is cyclic[3] and infinite.
Information can only the retrieved by transversal.

On a simple level it is possible to simply translate one relationship into a simpler one.

```
fun scoreFriends (_ * a) =
                              w = ... calculate net worth ...
                              case of w:
                                 high => 256
                                 low  => 3
end
```

Simple mappings from one set of assertions to another are useful, but cannot deal with more complex problems. As the representation is a directed graph, as such the natural way to summarise information is through graph rewriting. Since the graph is effectively infinite, it is necessary to select an appropriate sub-graph.

Normally this would be done by a predicate restriction on the assertions forming the graph. Hearsay can only retrieve information from the system by transversing the graph from one agent to another. It cannot have access to the full graph. This is a similar situation to web spidering.[34]

This 'spidering function' can only be used to select the relevant sub-graph. If such a transversal method were used to assign values to, or transform the assertions, it is possible for the result to be dependent on the order in which the assertions were retrieved.

[3]The graph contains edges A -> B -> ... -> C -> A .

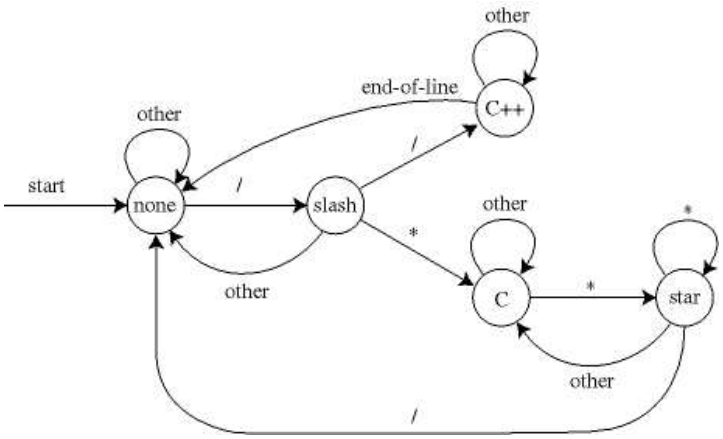


Figure 11.2.: Code Comments Automata

11.4.1. Regular Language for Relationship Selection

Since there is no way to create a graph without walking the network, it is important that we have an initial selection and gathering process. The selected sub-graph can then be processed.

Each time we retrieve an assertion we must determine whether or not it forms a part of the tree we are selecting, then which if any assertions to retrieve from its subject. If the a previously reached assertion is retrieved, then that branch is closed. This is effectively a regular language, which can be evaluated as a deterministic finite state automata.

A regular languages[45] over the alphabet of all possible predicates shall be used to determine the transversal of the social network and the selection of assertions from it.[4]

This regular language shall have the following formal clauses:

Identity <Symbol> Match this symbol, and no other.

Replication <Clause> <natural number range> Match a number of instances of the given clause, were that number of instances is within the given range.

Concatentation <Clause> <Clause> Match the the first of the given two clauses, and then the second of the given two clauses. Can be considered a logical AND, or as

[4] This type of tool could be considered as "awk for people".

sequential composition.

Alternation <Clause> <Clause> Match either of the two given symbols. Can be considered a logical OR, or as parallel composition.

```
signature Spidering =
    sig
        include Logic
        include Transport

        datatype nymGrep = Replication    of predicate * (int * int)
                         | Concatentation of predicate * predicate
                         | Alternation    of predicate * predicate
                         | Grouping       of predicate
                         | Identity predicate

        val spider: network * nymGrep -> socialNetwork
    end
```

Note on Termination *Replication* could be allowed for an unbounded range. Since it is not acceptable to spider the network indefinitely, only finite ranges can be used in Hearsay.

Graph Rewriting Having retrieved the desired sub-graph from the system, the Hearsay can manipulate it in graph form.

11.5. Signals and Transport

In all networkable systems the first aspect which must be defined are those things that may fail. Only after defining failure states, is it meaningful to define the protocol.

1. The network is reliable
2. Latency is zero.
3. Bandwidth is infinite.
4. The network is secure.
5. The network topology doesn't change.
6. There is one administrator.
7. Transport cost is zero.
8. The network is homogeneous.

Figure 11.3.: Eight Fallicies of Distributed Computing[35]

```
signature Faults =
    sig
        type appFaults
        datatype error = TIME_OUT | NETWORK_ERROR
                       | CONNECTION_CLOSED
                       | DENIED of (appFaults option)
                       | appFaults of app
    end
```

Only the signal defined below must to be sent over-the-wire by a Hearsay implementation. Hearsay implementations which share serialisation formats and transport layers will be able to intercommunicate.

```
signature Signals =
    sig
        include Faults
        type guid

        datatype request = query | subscribe | unSubscribe

        datatype result = Success of assertion set
                        | Failure of app error

        datatype signal = request | result
        type message = guid * nym * signal
    end
```

The system must provide access to a transport layer, with a mechanism which returns appropriate errors.

```
signature Transport =
    sig
        include Signals
        include Faults
        type network

        val send: signal * network -> result
    end
```

11.6. Publish - Subscribe

It is not necessary for a node to fetch information from the network every single time it is needed. A different approach, is to have the client node register its interest in certain information with the publishing node. When new information is added to a nym's register of assertions, the node will publish it to the entitled parties.

```
type pubSubRegistery = (nym * predicate) set
```

Then functions to add and remove subscriptions from that registery will then be.

```
val addSubscribe:
        subscribe * nym * pubSubRegistery -> pubSubRegistery

val removeSubscribe:
        unSubscribe * nym * pubSubRegistery -> pubSubRegistery
```

The clients are now free to assume that all assertion have the value last sent to them. It becomes the publisher's responsibility to send new information to the clients. This should have the effect of dramatically dropping the amount of traffic required on the system. Queries would largely be run against information that the node had subscribed to and would not require the sending of messages.

The trade-off is that the a node will have to send messages to all the subscribing nodes when an assertion changes.

This might led to a large number of outward-bound messages being sent at the same time. Care should be taken to avoid filling the outward-bound bandwidth.

11.7. Security and Access Control

The appropriate security model will depend on the implementation and the needs to the user. In all cases, Hearsay will require a trustworthy execution environment. Protection of physical storage depends on the execution environment.

A security policy should at least allow blocking of requests, as well as the filtering of returned results. Rather than simply allow or deny a request, the security manager ought to allow the rejected client to be given a specific reason, or for the request to be silently dropped. This is because the user may wish to prompt the client to undertake some action [for example, providing payment]. It will often be that the existence of something to be queried is a secret in itself.

```
signature Security =
    sig
        include Signals

        type permit = command -> command option
        type censor = command * assertion -> assertion option

        type policy = permit * censor
    end
```

11.7.1. Evaluating Execution Enviroments

Hearsay should not attempt to provide protection for stored data, intrusion detection or other such system level functionality. If the execution enviroment is untrusted, then so should any instance of Hearsay within it. The scarcity of trustworthy environments is a serious problem for any trusted system.

11.8. Agents and Persistence

Each user's nym(s) are represented by

```
signature Agents =
    sig
        include Relationships
        include Security

        type pubSubRegistery = (nym * predicate) set

        type agent =
        {
            self: nym,
            repository: assertion set,
            subsciptions: pubSubRegistery,
            security: policy
        }

        datatype change = add of assertion | retract of assertion
        datatype query = get of predicate
        datatype subscribe =  subscribe of query
        datatype unSubscribe = unSubscribe of query

        val search: assertion set * policy -> assertion set
        val alter: assertion set * agent -> agent

        val subscribe:
                subscribe * nym * pubSubRegistery -> pubSubRegistery
        val unSubscribe:
                unSubscribe * nym * pubSubRegistery -> pubSubRegistery
    end
```

An agent in the Hearsay system will have state representing:

- Its own identity
- The assertion store

- The security policy
- A register of subscriptions

11.9. Virtual Nyms and Assertions

It is not necessary that nyms or assertions, be either real or connected to the Hearsay network. To take live information from other sources, interpret it as Hearsay data and incorporate it is perfectly acceptable. Doing so would allow bridging of legacy systems onto Hearsay.

Verisign certificates (X.509)[5] could be represented as relationships between Verisign [a virtual nym] and the holders [a real nym].

```
(
    (Verisign, AnotherCompany),
    (
        ('http://versign.com/certs/'),
            {
                ('issued', 2001-02-03 12:32),
                ('valid', true)
            }
    )
)
```

Or we could use a virtual assertion to represent a persons presence, as inferred from a Bluetooth PAN.

```
(
    (MySelf, SomeoneElse),
    (
        ('www.example.org/standards/bluetooth/presence'),
            {
                ('distance', 50 meters)
            }
    )
)
```

[5] As described in the research.

Since the origin of the Hearsay trust network is the individual user, it is permissible to include anything the user wishes to trust into their assertions store. This is a distinct practical advantage over a centralised system.

11.9.1. Information Hiding with Virtual Assertions

There will often be assertion that can be made public that are derivable from assertions that cannot. For example, it is reasonable for a grocery store to vouch that a regular customer is not a spammer, however it would be inappropriate to expose the times and amounts of their purchases. The store might create and publish a virtual assertion of not-a-spammer, while keeping the assertion of $30-every-other-Friday restricted to staff accounts.

11.10. Witnessing and Repudiation

Generally it is desirable that communication between nodes should not produce verifiable records. In fact, a node should not really keep logs at all. Verifiable records of relationships can create unwanted embarrassment, and even legal liability. Normally it is highly desirable that communication can be repudiated.

If our transport layer has *perfect-forward-secrecy*[6] then repudiation will be possible, provided that both nodes discard their session keys. It is possible that one or both parties might risk attempting to store both the session traffic and the session key. For most cases this possibility is acceptable since normally it will be equally important for both parties to have secrecy. This could be described as mutually assured disclosure (MAD). In any case, those requiring true anonymity will be using a nym that is not linked to their real-world identity.

Sometimes it may be necessary to establish that a certain set of assertions were valid at a particular time. (For ex. contracts and agreements.) With a repudiatable protocol, any logged communication will be inconclusive. If we wish to create a record a digital signatures must be deliberately attached to an assertions.

[6]See the research for an explaination.

```
signature Witnessing =
    sig
        include Agents

        datatype signed = signed of nym * assertion
        type statement = assertion * signed

        val sign: assertion * agent -> signed
        val verify: statement * nym -> bool
    end
```

These witnessed statements can then be kept as a proof for later reference.

11.11. Wire Protocol Specification

A Hearsay implementation need only serialise the *signal* type. It must reduce the following tree to a sequence of transport primitives.

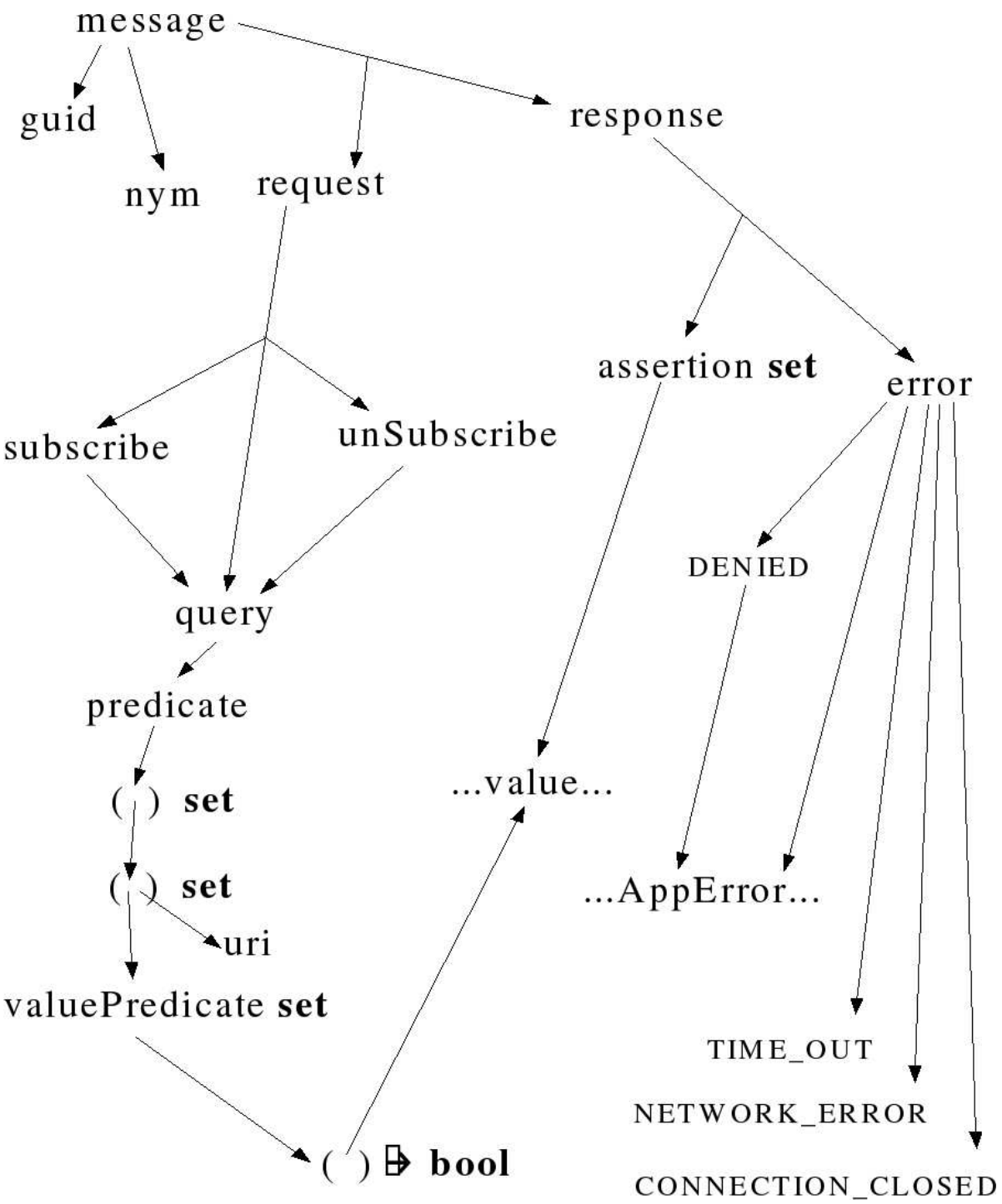


Figure 11.4.: Hearsay External Wire Protocol

This provides a single known **Abstract Syntax Tree** for the Hearsay system's external interface.

12. Scalability

Hearsay implementations cannot pass mobile code, since the termination properties of that code cannot be guaranteed and it raises larger security issues. In a live system it is not meaningful for a node to issue assertion that are not about its own nym. Therefore, there will be as many agents within our network as there are nyms. Each piece of information transferred will involve an interaction between the asserter and the client that wishes to know of it.

Assuming that:

- Interactions consume 1 kilobyte (1 A4 page of XML) of bandwidth at each end.
- Each nym regularly interacts with 2000 others.
- That on average these interactions happen three times an hour.

Then there will be:

$$2000 nyms \times 6 interactions/nym/hour \Rightarrow 3.33 interaction/second$$

$$1 kilobyte/interaction \Rightarrow 3.33 kilobytes/second$$

Which is achievable. Many systems achieve vastly better message processing performance.[49, 34, 39] Clearly messages are not evenly distributed in time. But neither would they be in other systems, so the comparison is still reasonable.

12.0.1. Publish-Subscribe

Since users will want (and be permitted to see) less information on those that are only distantly connected to them, it would be expected that nyms would form clusters. Each nym would have an inner band of nyms with which it communicates frequently, then

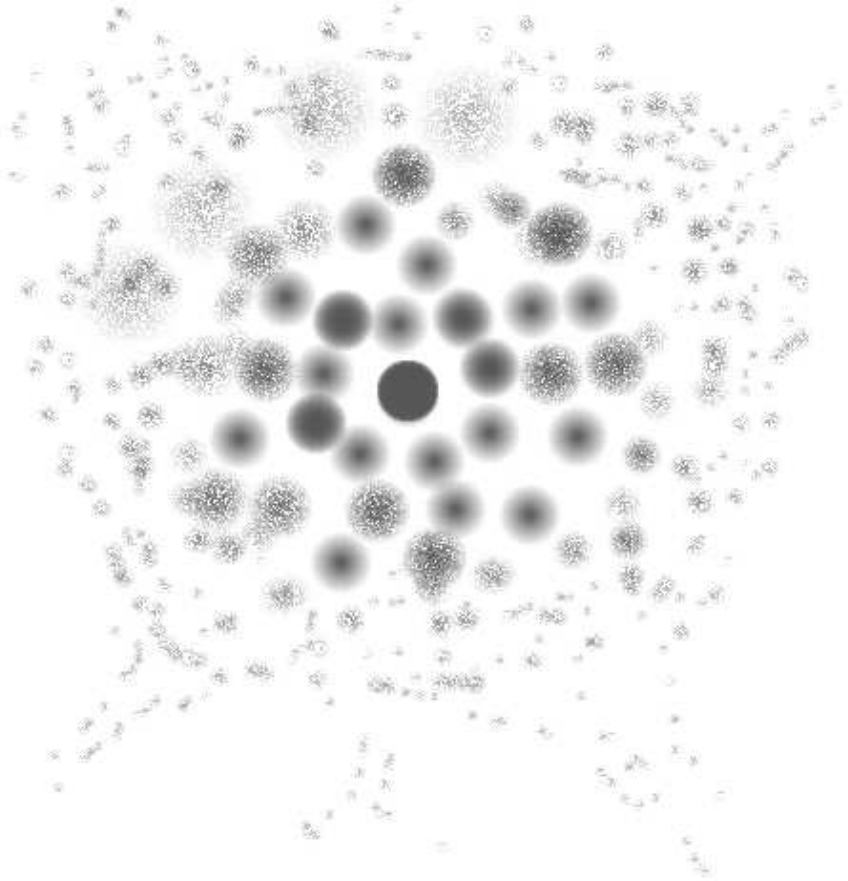


Figure 12.1.: Anticipated Behaviour of Hearsay Clustering

bands of agent at widening distances around it with which it would have increasingly less contact.

Such a subscription mechanism would be restricted primarily by the efficiency of the transport layer in broadcasting messages to large numbers of clients. Ideally this would make use of UDP[1] to cascade messages safely from from the inner nodes to the outer nodes.

[1]The User Datagram Protocol offers a minimal transport service - non-guaranteed datagram delivery. Giving direct access to the datagram service of the IP layer. UDP is used by applications that wish to use communications services (e.g. multicast) not available from TCP.

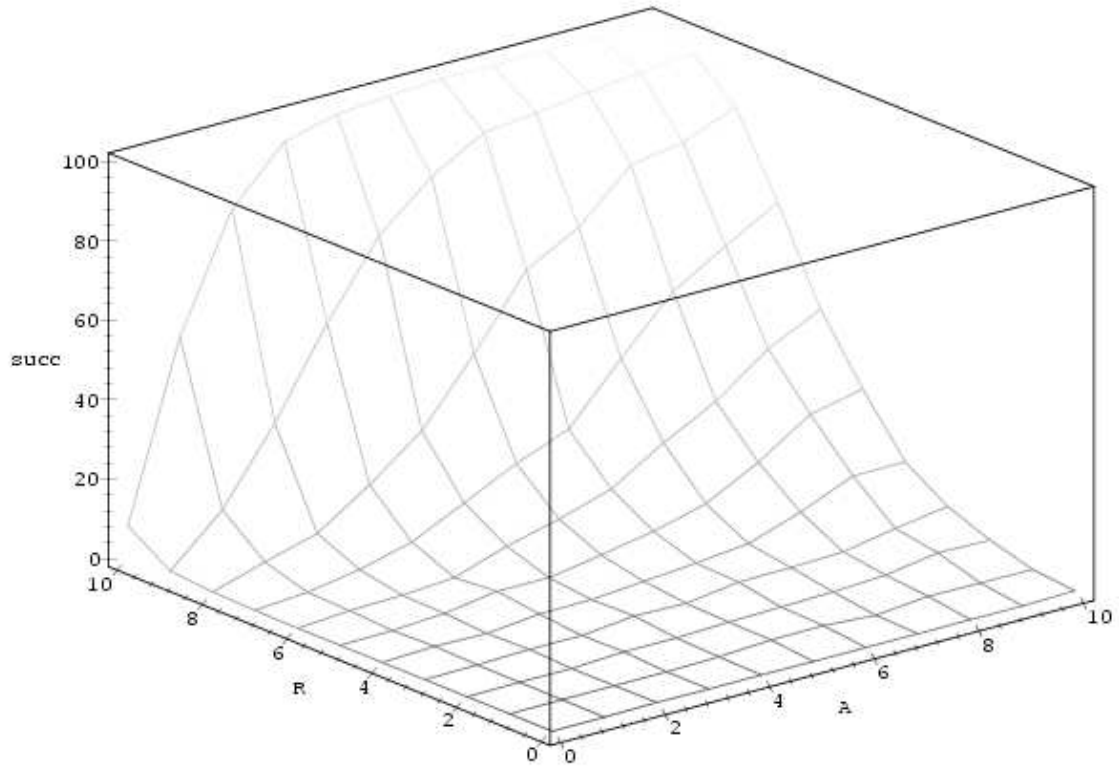


Figure 12.2.: ACC Scalability

12.0.2. Performance of Similar Systems

It would be expected that searching for an assertion on the Hearsay network would be dependent on the rarity of matching assertions, and also on the number of 'free message' coming from a nodes own assertion store (or any communication cache).

Both Yenta[24] and ACC[25] use similar query models to Hearsay. It can been seen that all both graphs are of the same form.

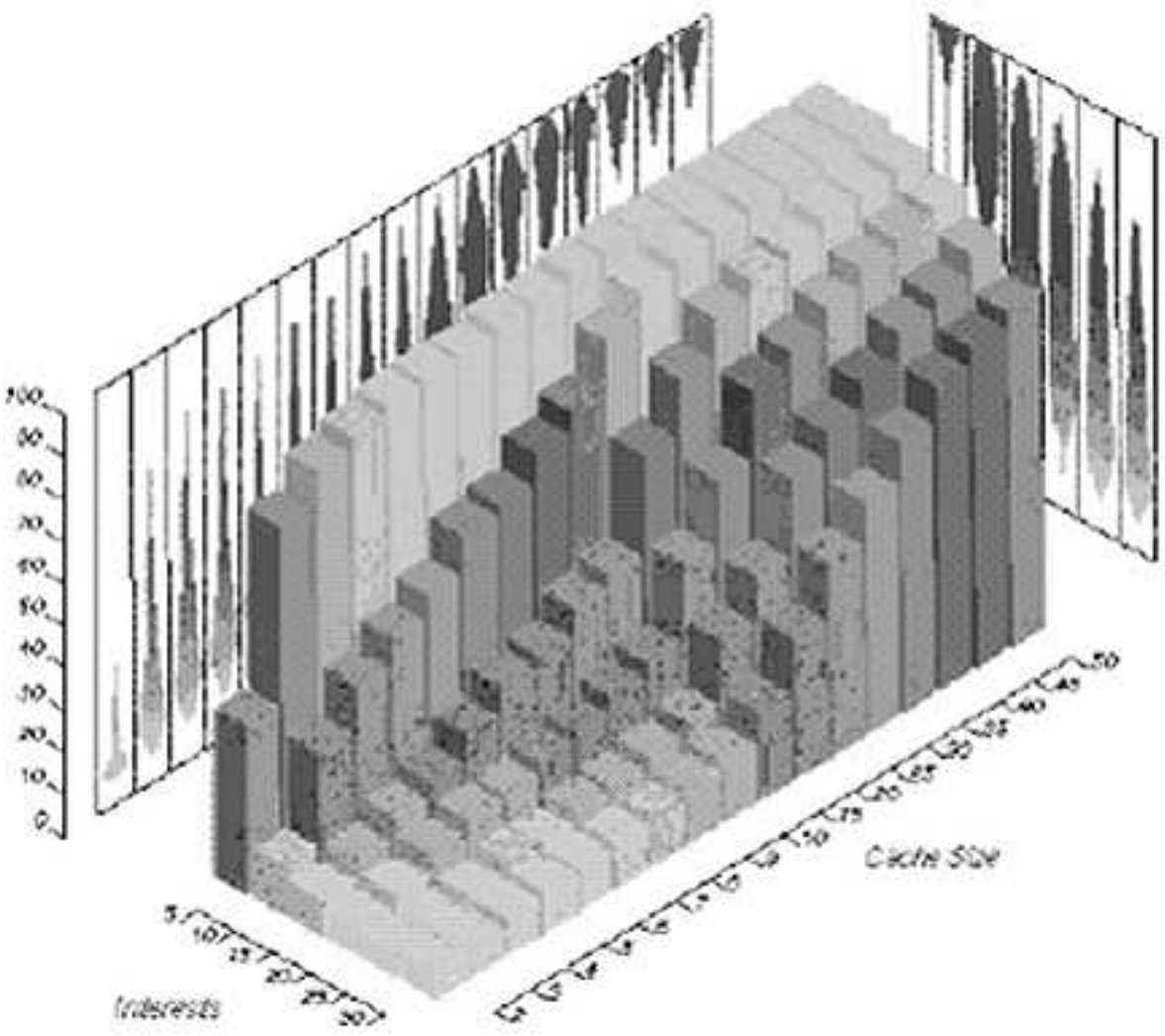


Figure 12.3.: Yenta Performance Profile

13. Practical Design and Implementation

> "*Object-oriented programming is an exceptionally bad idea which could only have originated in California.*" - Edsger Dijkstra

The practical design of Hearsay is relatively simple compared to most peer-to-peer applications. The practical guide presented here, explains how the functional specification can be implemented within a modern object-orientated environment.

From experience of the prototype, it is important to know before starting:

- The data transfer objects available in the chosen platform.
- The failure modes of the transport layer, and the required recovery actions.
- How to initialise the available security systems.
- Standard library types for Dates, etc.

Rewriting Functional Structures

Functional types from the specification can be re-written into objects structures quite easily. Datastructures can be mapped through into objects. For example:

```
type assertion = direction * record
```

and in Java as

```
public final class Assertion
{
    private final Direction direction;
    private final Record record;
    private Assertion(Direction direction, Record record)
    {
        this.direction = direction;
        this.record = record;
    }
    public final Direction getDirection()
    {
        return direction;
    }
    public final Record getRecord()
    {
        return this.record;
    }
    public final boolean equals(Object o)
    {
        ...
    }
    public static Assertion createAssertion
        (String by, String about, String category, Set fields)
    {
        ...
    }
}
```

The two structures carry the same information. Note that the Java object is immutable. Normally Java data objects are mutable, but this is not the easiest case when working from a functional specification.

```
val contains: period * period -> bool
```

Functions can be re-written in two forms, with or without the use of the static context.

```
result = SomePeriod.contains(AnotherPeriod); // dynamic
```

```
result = PeriodOperators.contains(SomePeriod, AnotherPeriod) // static
```

The two are logically equivalent.

Techniques

From experience, the following techniques were very useful in implementing these forms of system.

- Unit-Testing should be done through-out. It is the simplest means finding basic errors.
- A good IDE is important. Without one, it is hard to deal with object structured code.
- Round-trip UML tools are very useful in 'top-down' development.

13.0.3. Design Patterns

In creating netoworked applications there are now common practices and techniques. These are normally represented as patterns: 'cookie-cutter' architecture techniques which are known to be workable.

Patterns useful to the Hearsay design are presented below. They are taken from '*The Patterns of Enterprise Architecture*'[36] by Martin Fowler, which is the standard reference work on enterprise system patterns.

ValueObject A small simple object representing a data value whose equality is not based on identity.

QueryObject An object which represens a query on a data store.

Registry A well know object which other objects use to find information and services.

Ghost An object that does not contain the information you need, but will retrieve it on request.

DataTransferObject An object which carries data between processes in order to reduce the number of method calls.

ServiceLayer Defines an application's boundary with a layer of services that establsihes a set of avaliable operations and coordinates the application's response in each operation.

13.1. Practical Overview

13.1.1. Package Layout

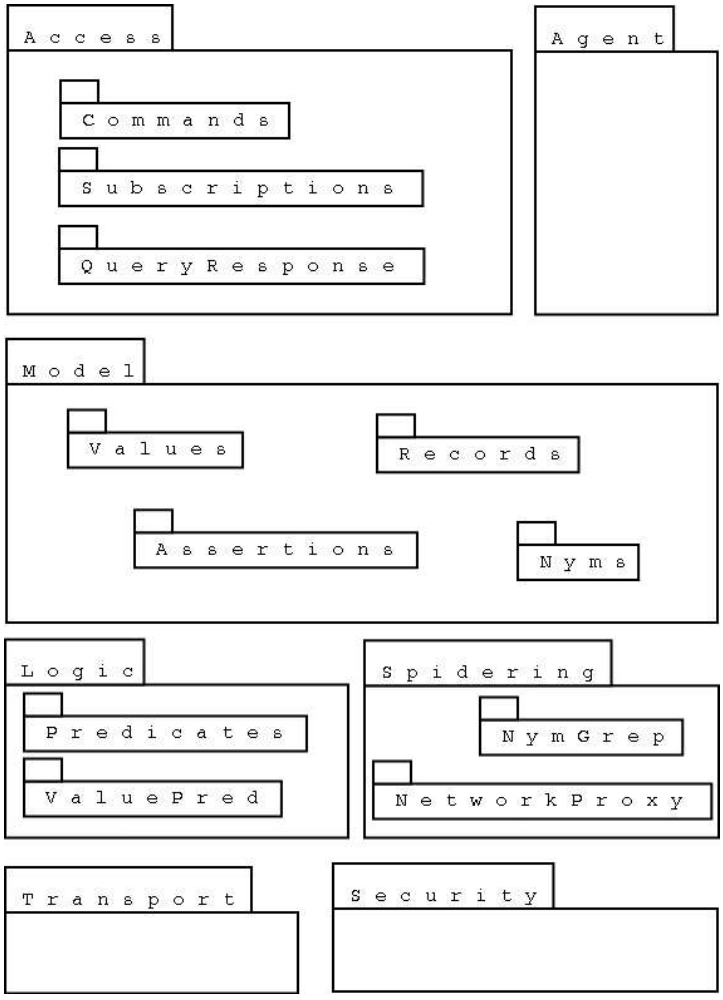


Figure 13.1.: Suggested Package Arrangement

13.1.2. Model

This package should encapsulate all the values that Hearsay uses, as *ValueObject*s in accordance with the *DomainModel* pattern.

13.1.3. Logic

This represents the predicate logic as as functors which evaluate them. Contains logic for 'rolling-up' value predicates into first-class predicates refering to assertions.

13.1.4. Access

*Query Object*s which encapsulates requests to the system.

13.1.5. Security

Defines as *pluggable* class that is called to make policy decision about which information to release in repose to which commands from whom.

13.1.6. Agent

Defines a single class which acts as a container for the whole state of a user's nym. This class will contain *Register*s for the subscription services, and for the assertions themselves. The Agent class handles the external commands defined by the Access package, in accordance with the Security policy.

13.1.7. Spidering

Defines the NymGrep regular language which controls the searching of the user's social networks. Provides evaluation of this language, using *Ghosts* as network proxies to limit network access.

13.1.8. Transport

Basic interfaces for networking. This is best achieved by encapsulating the data to be communicated in within *DataTransferObjects*. These transfers can then take place through the *ServiceLayer* of the transport mechanism used.

14. Technology Options

There are a number of basic communication stacks that could be used for implementing Hearsay. Such a technology stack should allow adherence to the listed principles above, and be simple for the end-user to configure. Above all it must have a large and growing developer base: Hearsay is intended to be a practical technology.

14.0.9. Hypertext Transfer Protocol (HTTP)

Positive

- Simple
- Semantic Web

Negative

- Hard to deploy on the desktop
- Designed for static content
- Only supports polling

Summary Hearsay would be very limited without some publish-subscribe functionality.

14.0.10. Email

Positive

- Has been used before.[25]
- Ubiquitous

- Huge developer base
- Easily understood

Negative

- Not suited to small messages
- Flooded with spam
- No structured data support

Summary Lack of structured data support makes email unworkable.

14.0.11. JXTA

Positive

- Flexible
- Trust and Security Domains
- Multicasting Enabled

Negative

- No structured data standard
- Complex APIs
- Small developer base

Summary This is Sun's Java Peer-to-Peer System.[37] This would make a good choice in principle, it does not have the existing infrastructure or developed base for creating popular systems.

14.0.12. Extensible Messaging and Presence Protocol (XMPP)

Positive

- Easy to write
- Simple to deploy
- Popular
- Allows third-party protocols by design.

Negative

- Advanced encryption still undetermined
- No multicasting

Summary Despite the lack of multicasting, XMPP is easy to write for, supports structured data, and popular. It is therefore the most appropriate choice for Hearsay.

15. Introducing XMPP

XMPP[1] (Extensible Messaging and Presence Protocol) is an XML streaming protocol for presence and messaging routing. Recently ratified as an IETF standard, XMPP serves as the basis for the Jabber instant-messaging system and provides a general language for linking diverse networks. XMPP serves as a universal transport layer for XML structured data. It embeds presence and context sensitivity into that data, which lets the data be routed efficiently to the most appropriate resource.

XMPP is a young, but growing standard. Due to the simplicity of the standard there are currently 11 different server implementations, and numerous clients. There are emerging protocols for public-key exchange and signatures. The flexibility and security of XMPP have made it popular in finance and banking.

The information in this section is referenced from [38, 39].

[1] The system is a formalisation of the earlier Jabber IM system. Therefore many sub-components of XMPP are named after Jabber. The two terms should be considered interchangable.

Name	Namespace	JEP	Status
Core XMPP	n/a	xmpp-core	Active
Request-response	jabber:iq	xmpp-core	Active
Error messaging	n/a	JEP-0086	Active
Entity Time	jabber:iq:time	JEP-0090	Active
Publish-Subscribe	http://jabber.org/protocol/pubsub	JEP-0060	Active
VCards	vcard-temp	JEP-0054	Active
Encrypted Sessions	http://jabber.org/protocol/esession	JEP-0116	Deferred

Table 15.1.: Standard XMPP Vocabularies

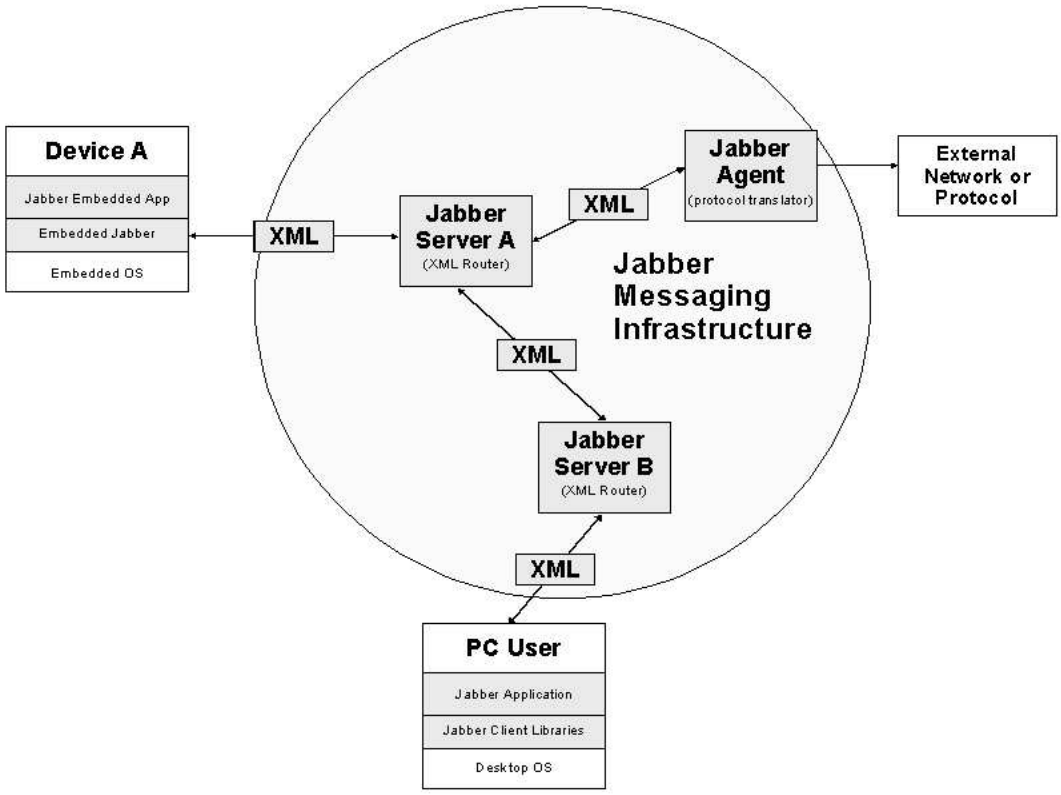


Figure 15.1.: Simple XMPP Scenario

15.1. XMPP Basics

The XMPP protocol consists of a predefined exchange of XML Stanzas. An XML Stanza is a discrete stream of XML events from a predefined vocabulary. Such valid vocabularies are usually defined using XML Schema Definitions (XSD), and are qualified with an XML namespace.

The core of XMPP routing is an logical addressing scheme, represented as nym@domain/resourc

In the XMPP this is referred to as the Jabber ID (JID). The domain portion can be looked up in the DNS, similarly to an e-mail address. As in email, servers connect with one another on behalf of users (the node portion of an SMTP[2] address). The resource portion is a connection identifier that lets a single user be logged on multiple times simultaneously.

[2]Simple Mail Transport Protocol

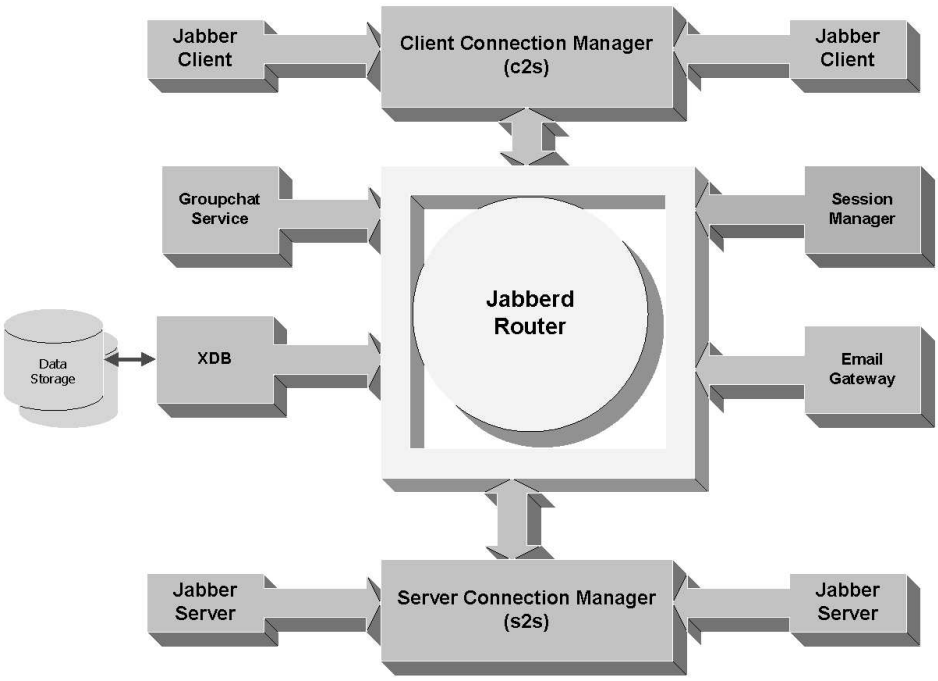


Figure 15.2.: XMPP Modules System

Servers

When clients connect to a server, they authenticate with it, specify a resource to register as, and tell to the server to announce their presence on the roster. Servers find, connect and authenticate to one another, letting any client connected to communicate with any other client regardless of their home server.

Modules

Services on XMPP servers are provided by modules which are connected to the server. Modules intercept, receive transform and send messages to provide functionality.

Gateways

Gateway are modules which connect external resources to an XMPP server to allow transparent interoperability with legacy messaging systems.

Has links back to:

- MSN Messenger

- America Online Instant Messenger
- Yahoo Messenger
- Fax by internet services
- Email (SMTP)

15.2. Encryption

There are two forms of communication in XMPP that can be secured by encryption: client-to-server, and server-to-server. Transport layer security (TLS) is always used to secure server-to-server traffic using secure socket layer (SSL).[42] Client-to-server traffic can be secure in the same way. It this is recommended that this be made manditory in Hearsay systems.

15.2.1. End-to-End Encryption

With the standard methods, the user is *forced* to trust the server operator. This has been recognised as being undesirable, with several methods being proposed for end-to-end encryption. The need for the server to read part of the message, in order to route it, has made this a difficult issue.

OpenPGP A simple wrapper format for OpenPGP messages is currently being used b y many clients as a secure standard. This is secure[23], but somewhat inelegant. It creates verbose messages and does not interoperate well with the XMPP paradigm.

Encrypted Sessions Discussions are in progress to create an official standard for end-to-end encryption. A deferred standard does exist, the IETF hope to produce a final version soon.

15.3. Firewalls

Companies have two policies on firewalls: the firewall must stop anything potentially harmful, and all our applications must tunnel through firewalls to deliver a rich user experience. Fortunately XMPP comes complete with several firewall tunnelling options.

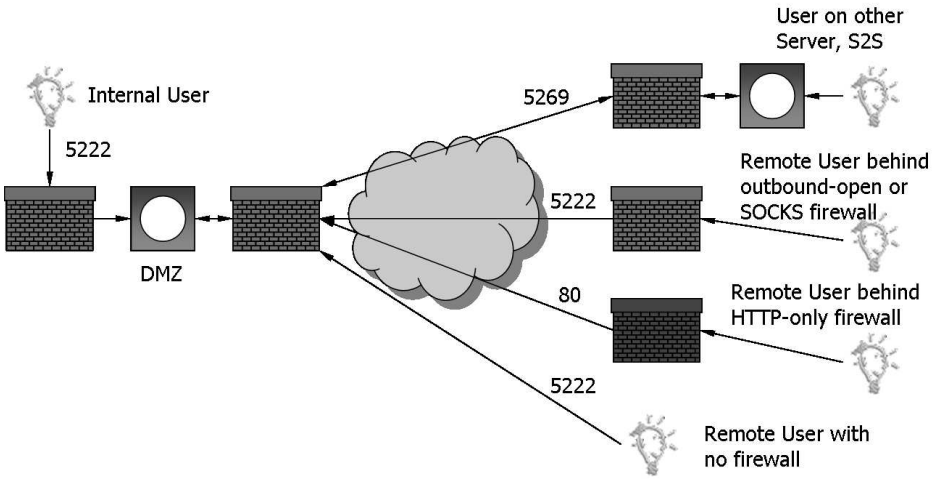


Figure 15.3.: XMPP Firewall Tunneling

16. Hearsay over XMPP

16.1. Concrete Protocol

> "*Programming language people have a term for making the syntax of a language pretty: syntactic sugar. XML is syntactic arsenic.*"
>
> - Patrick Logan

An abstract syntax tree has already been defined, it is therefore only necessary to produce a table mapping between the productions of this tree and approperiate XML serialisations.

Interoperability with Other Serialisations The abstract syntax tree defined by the Hearsay functional specification. Therefore it would be possible for two entirely different Hearsay transport protocols to be bridged onto one another, since they would be isomorphic.

Re-use of Existing Standard Where possible existing standards are being used in the Hearsay XMPP protocol design. For example, the Hearsay error messages cover the full range of potential problems in Hearsay and can be directly used in the specification.

16.2. Deployment Options

Personal Server The user owns and runs their own server. This avoid having to trust a server administrator with user messages. Running a personal server means buying a domain name and providing a server system.

Shared Server If a shared server is used, the user must trust the server administrator (but not the other users). Either this or end-to-end encryption must be used.

Production	XML
message({guid}, {nym}, {content})	XMPP Core
nym	XMPP account address
guid	XMPP generated message id
uri	Standard XML URI
request	XMPP Core
subscribe({query})	<subscribe>{query}</subscribe>
unSubscribe({query})	<unsubscribe>{query}</unsubscribe>
query({predicate})	<query>{predicate}</query>

predicate{ {uri}, ...{...{valPredicate(n)}...}...} }	`<cnf> ...`
	`  <and>`
	`  {...`
	`  <pred recordNamespace=''{uri}''>`
	`  ... {valPredicate(0) ... valPredicate(n)} ...`
	`  </pred>`
	`  ...}`
	`  </and>...`
	`</cnf>`

response{ {...{assertion}...} | {error} }	`<result>`
	`  {...{assertion}...}`
	`  </result>`
	`|`
	`  {error}`

{error}	XMPP Core Error Codes

Table 16.1.: Model Wire Types to XML

16.2.1. Security Implications

- Many messages = lower risk of traffic analysis
- Shared server = must trust owner

16.3. Scalability

- Multicast needed for true scaling, since repeated TCP/IP connections will become increasingly slow.
- XMPP good enough for most purposes.

Good reason for future JXTA[37] implementation, and the ML-defined standard signals would allow for mapping of signals between the two implementations.

17. J2SE Architecture

17.1. External Module Execution Flow

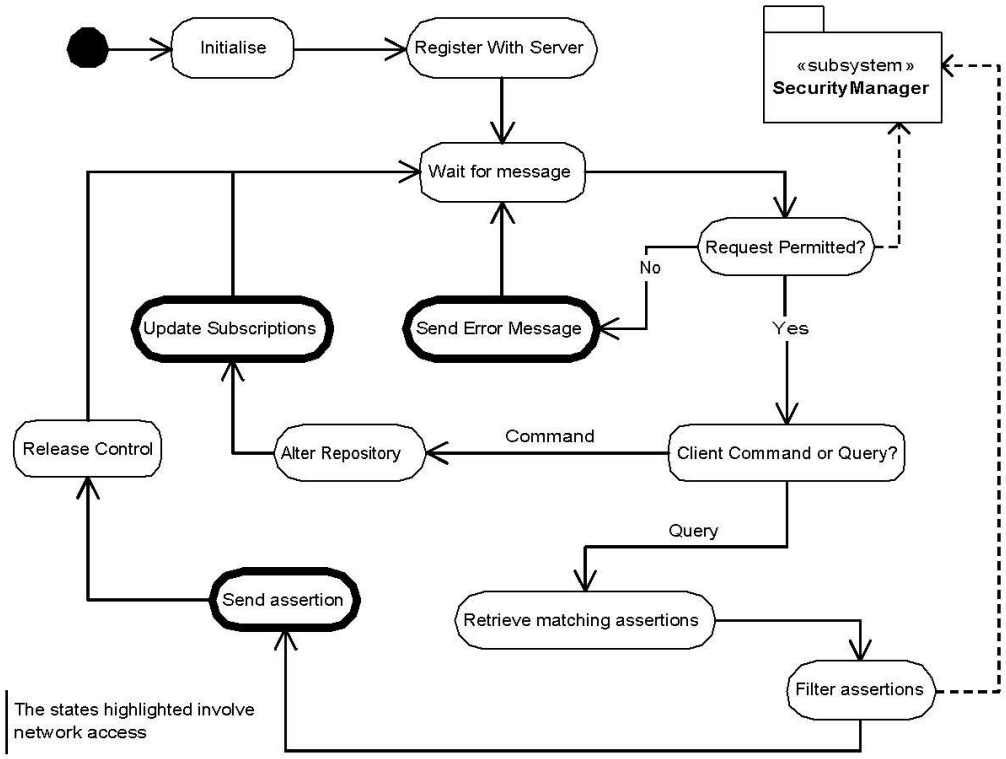


Figure 17.1.: State Diagram for Hearsay Service

17.2. Smack XMPP Messaging API

The Smack API is a Java XMPP library published by Jive Software[40] under the Apache Licence[41].

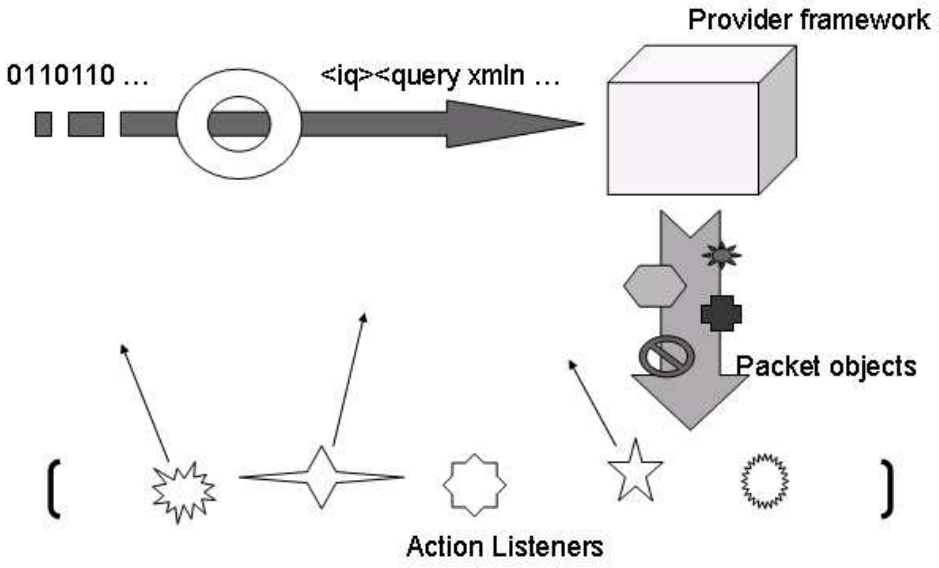


Figure 17.2.: Smack Messaging Process

- Smack is a library for communicating with XMPP servers to perform instant messaging and chat.
- Sending a message to a user can be accomplished in three lines of code.
- XML parsing is simple and fast.
- Will work on resource constrained devices, e.g. mobile phones.
- Handles networking concurrency.
- Creates 'boilerplate' XMPP stanzas

Smack lets you set any number of properties on each message, including properties that are Java objects.

Smack has a very cleanly written object orientated design. Using the Smack API is simple. First create a connection object, login to the connection object, then attach listeners for packets. Filters are used to select the actions your listeners respond to.

```
XMPPConnection connection = new XMPPConnection("jabber.org");
connection.login("mtucker", "password");

// Create a packet filter to listen for new
// messages from a particular user.
// We use an AndFilter to combine two other filters.
PacketFilter filter
    = new AndFilter(new PacketTypeFilter(Message.class),
          new FromContainsFilter("mary@jivesoftware.com"));

// Assume we've created an XMPPConnection name "connection".
// First, register a packet collector using the
// filter we created.
PacketCollector myCollector
      = connection.createPacketCollector(filter);

// Normally, you'd do something with the collector,
// like wait for new packets.
// Next, create a packet listener.
// We use an anonymous inner class for brevity.
PacketListener myListener = new PacketListener() {
        public void processPacket(Packet packet) {
            // Do something with the incoming packet here.
        }
    };

// Register the listener.
connection.addPacketListener(myListener, filter);
```

Packets are objects that represent XML stanzas. These packets are created through processing the incoming XML Stream using provider objects. New providers must be created to process each type of data.

```
public final class ResultsListProvider
                     implements IQProvider
{
    private final ResultsPullReader listReader
              = new ResultsPullReader();
    public final IQ parseIQ(XmlPullParser xpp)
                      throws Exception
    {
        ResultsList result
           = new ResultsList(
                (IAssertionIterator)
                this.listReader.process(xpp)
             );
        return result;
    }
    public static final class ResultsList extends IQ
    {
        private final IAssertionIterator results;
        public ResultsList(IAssertionIterator results)
        {
            this.results = results;
            this.setType(IQ.Type.RESULT);
        }
        public final IAssertionIterator getResults()
        {
            return this.results;
        }
        public final String getChildElementXML()
        {
            return "<result xmlns=\"jabber:iq:hearsay\" />";
        }
    }
}
```

When a packet listener is activated, a method on the listener is called. The listener's 'process' method can then process the packet.

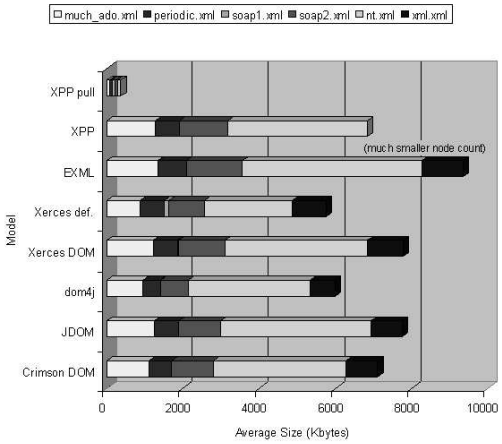


Figure 17.3.: XPP Memory Consumption[47]

17.2.1. About XML Pull Parsing

XML Pull Parser (XPP) is a recent development that demonstrates a different approach to XML parsing. Pull parsing is just one level up from tokenizing XML. XML pull parsing allows incremental (sometimes called streaming) parsing of XML where application is in control - the parsing can be interrupted at any given moment and resumed when application is ready to consume more input.

A restriction of XPP is that it does not support entities, comments, or processing instructions in the document. XPP creates a document structure consisting only of elements, attributes (including Namespaces), and content text. This is a serious limitation for some types of applications. However, XML Streams do not support any of these, so the issue is moot for XMPP.[46]

The pull-parsers works by postponing parsing until a component of the document is accessed, then parsing as much of the document as necessary to construct that component. This allows for very fast document-screening or classification applications, especially in cases where documents may need to be forwarded or otherwise disposed.[47]

These performance characteristics make XML Pull Parsers an ideal choice for working with mobile applications, where both memory and processors are limited.[47, 46] Below is a code sample from the XPP website, which demonstrates use of the API.

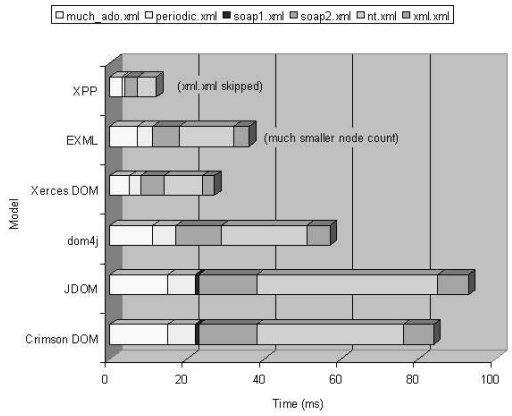


Figure 17.4.: XPP Performance[47]

```
public void processDocument(XmlPullParser xpp)
    throws XmlPullParserException, IOException
{
    int eventType = xpp.getEventType();
    do {
        if(eventType == XmlPullParser.START_DOCUMENT) {
            System.out.println("Start document");
        } else if(eventType == XmlPullParser.END_DOCUMENT) {
            System.out.println("End document");
        } else if(eventType == XmlPullParser.START_TAG) {
            processStartElement(xpp);
        } else if(eventType == XmlPullParser.END_TAG) {
            processEndElement(xpp);
        } else if(eventType == XmlPullParser.TEXT) {
            processText(xpp);
        }
        eventType = xpp.next();
    } while (eventType != XmlPullParser.END_DOCUMENT);
}
public void processStartElement (XmlPullParser xpp)
{
    String name = xpp.getName();
    String uri = xpp.getNamespace();
    if ("".equals (uri)) {
        System.out.println("Start element: " + name);
    } else {
        System.out.println(
            "Start element: {" + uri + "}" + name);
    }
}
```

17.3. Communication Stack Diagram

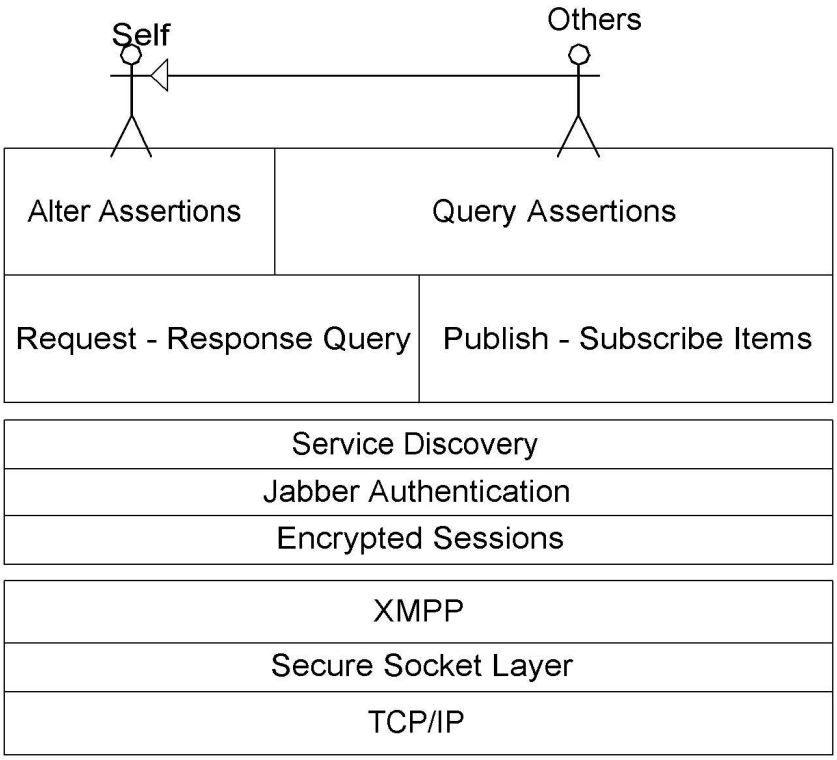


Figure 17.5.: Hearsay Communications Stack

Part IV.

Implementation Work

18. Project Management

18.1. Design Methodology

> "*At least with XP you can cover-up a dodgy design.*" - Anon. Warwick Student

The development of the Hearsay protocol was performed in a deliberately disciplined manner. The design was developed by means of continuous refinement, so that the reader would be able to follow the decisions made and understand them. This is not necessarily the most efficient method of design, but it is the most transparent.

- The use of SML was extremely helpful in properly seperating functional and structural concerns.
- Enterprise Patterns[36] provided simple solutions to common problems.

18.2. Code Standards

Based on Sun Coding Standards and Sun JavaBeans conventions.[48]

Naming

- Names should be as clear as possible. The greater the scope of a variable, the clearer its name should be. A loop variable named 'i' is fine, a member variable named 'i' is not.
- Method names begin with a strong active verb. Accessors (getters) begin with "get", or for boolean variables, optionally, "is". Mutators (setters) begin with "set" (i.e. follow Java Beans style).

- " Interfaces should start with the letter 'I': ICommand, IDataManager.
- Abstract classes should start with the word 'Abstract' or 'Template'.
- Constants are ALL_CAPS, separated with underscores.

Constructors and Getters/Setters.

- Use @pre and @post tags to document any non-obvious pre or post conditions to functions.
- In functions comments describe "why" rather than "how". C-style comments (//) should be used to avoid visual confusion with JavaDoc method headers, and so that code can be temporarily commented out using /*..*/ blocks.

Constants

- No "Magic Numbers" except for 0 and 1. Only time other numbers should occur in code is being assigned to static final constants.
- Use gettes/setters for object constants. This is optimised by the Java compiler and is not markedly less efficient. Has the advantage of much greater future flexibility. e.g. for(int i = 0; i < getNumItemsToDisplay(); i++);

Coding

- if/else/while/do blocks etc. must have braces.
- Single access-point to all private variables: i.e. a private (or protected) accessor method should be the only method to access a private variable.
- Member variables should precede methods in the code, since someone reading source files is interested in implementation rather than interface (for which JavaDoc may be used).
- Matched opening and closing braces should always be on the same column.
- Tabs are 4 spaces wide, and the IDE should be set to convert tabs to spaces.

Exceptions

- throws .. declarations should be underneath the declaration of the function e.g: public void foo(int bar) throws RandomException, AnotherRandomException { }
- try/catch blocks are of this form: try { } catch(RandomException e) { } finally { }
- Exceptions should be checked if, and only if, they are an expected part of the use of an object. If an exception should only be thrown as the result of programming error, the exception should be unchecked.
- It is unlikely that an application will be able to usefully deal with an error resulting from programming or configuration error and catching the error uselessly can tempt an empty catch block, leading to it being ignored (very hard to test for).
- Checked Exceptions are for occurrences outside the normal flow of control, but within the expected usage of a class. Runtime Exceptions are for exceptions which should not occur in the normal running of a program.
- if, while, do, else (etc.) blocks must contain {} braces.

18.3. Source Code Management

The sourcecode was kept under Concurrent Versioning System on a SourceForge[1] account. This provided easy remote access to the source and robust versioning capabilities. These facilities are invaluable when working with complex code.

[1] Sourceforge are a development tools company, who provide free hosting for open-source projects on their servers.

19. Prototype

> "*It is not the critic who counts: not the man who points out how the strong man stumbles or where the doer of deeds could have done better. The credit belongs to the man who is actually in the arena, whose face is marred by dust and sweat and blood, who strives valiantly, who errs and comes up short again and again, because there is no effort without error or shortcoming, but who knows the great enthusiasms, the great devotions, who spends himself for a worthy cause; who, at the best, knows, in the end, the triumph of high achievement, and who, at the worst, if he fails, at least he fails while daring greatly, so that his place shall never be with those cold and timid souls who knew neither victory nor defeat.*"
>
> -Theodore Roosevelt

A prototype was created so as to design the Hearsay system with a full knowledge of the issues involved. The aim of the prototype was not to complete the whole application; it will after all be re-implemented by most of its users. There is also insufficient time to properly complete and test an application of this size.

19.1. Scoping and Specification

The current Hearsay implementation ignores features not needed for a proof-of-concept.

- It does not support publish-subscribe.
- Offer some of the predicates and data types listed in the standard.
- The prototype does little of use to an end-user; currently it can only be used from the API.
- The code uses a simplified XML schema in interpreting the messages it receives.

19.2. Full System Currently Implemented

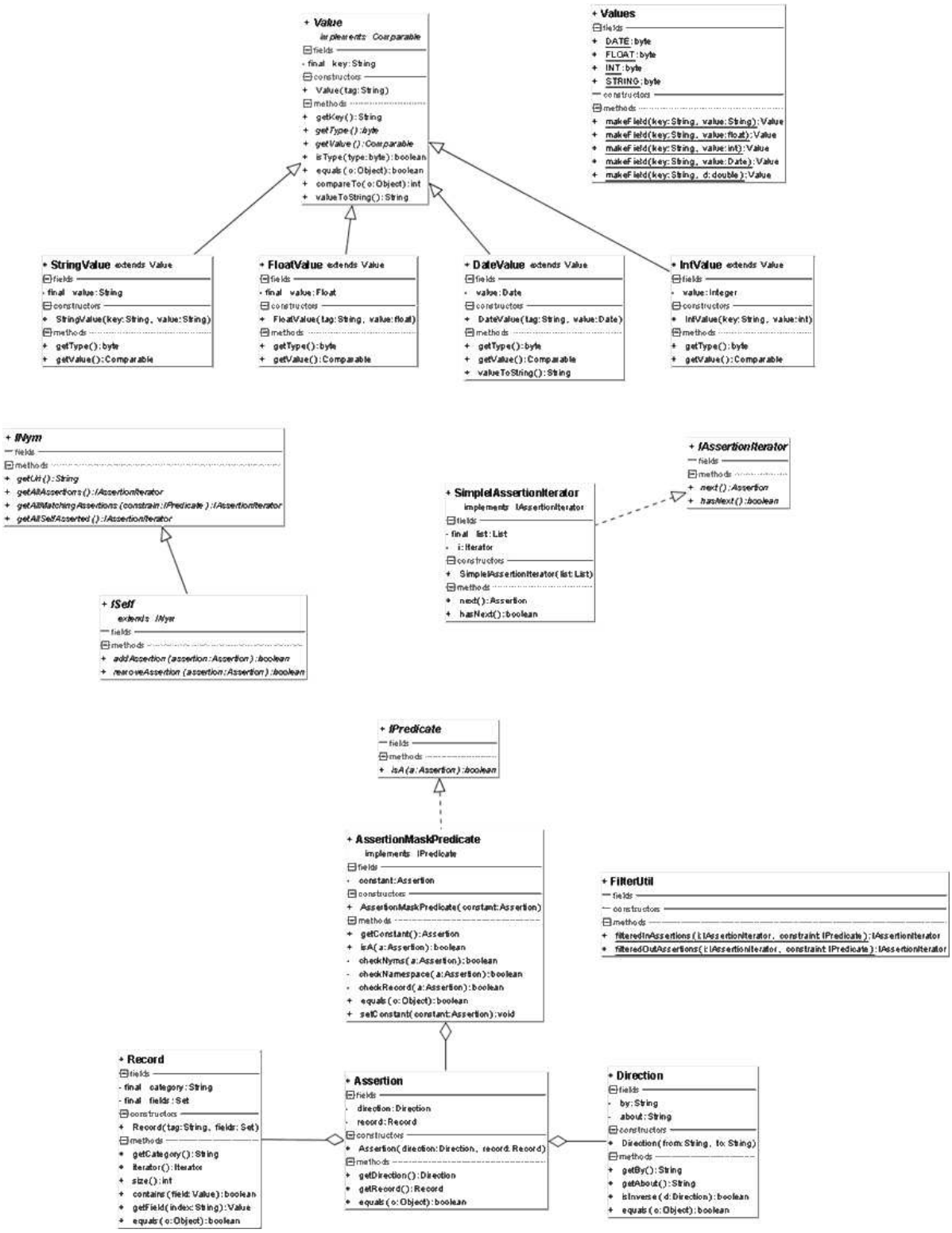


Figure 19.1.: Core UML Classes

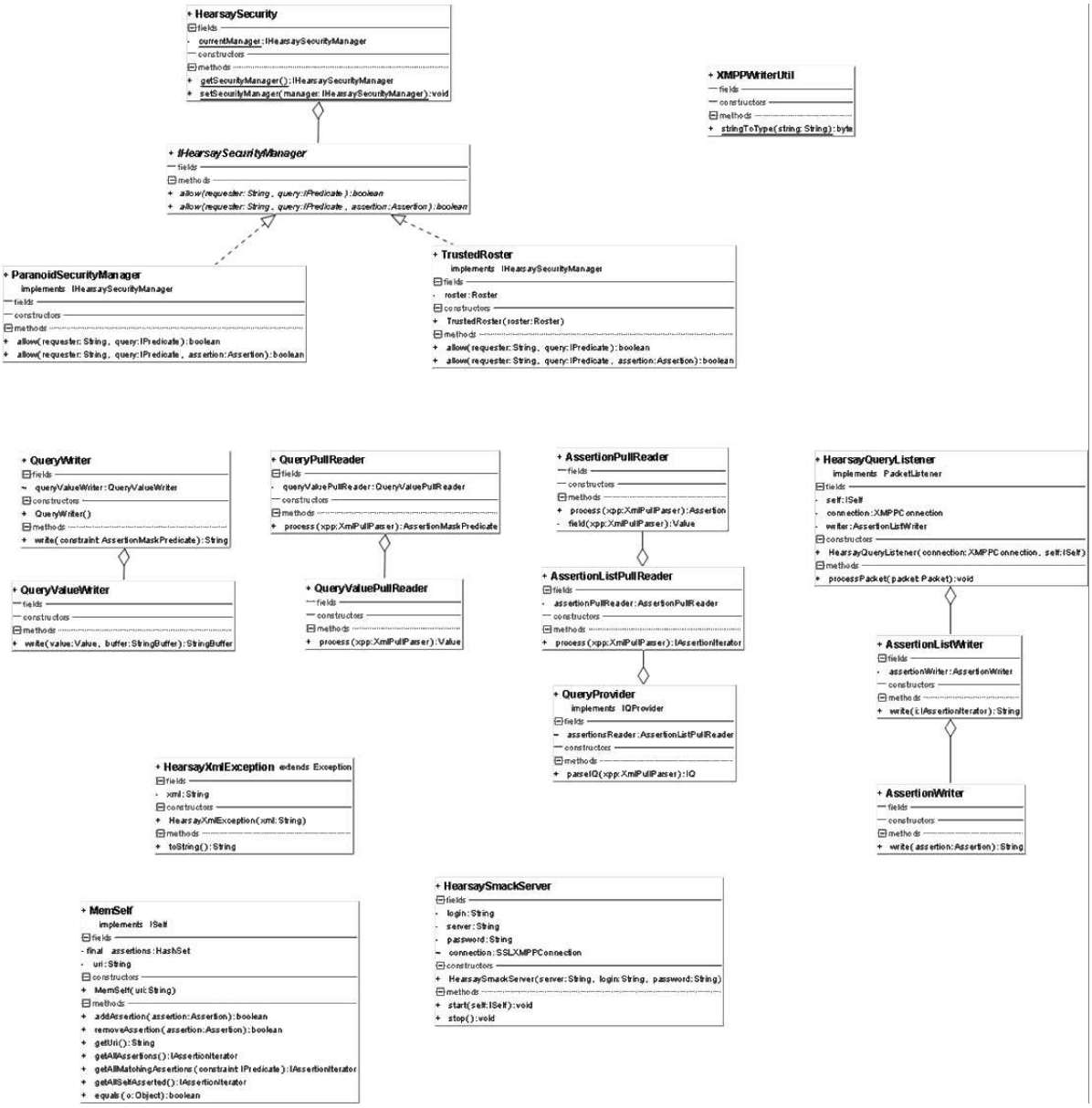


Figure 19.2.: XMPP UML Classes

19.3. Unit Testing

Unit-tests were added continuously to test each feature as it was implemented.

- Core logic was tested with a sample dataset drawn from the MegaTokyo web comic.
- Writers and packet providers were tested with both correct and incorrect data.

List of XMPP test-suites:

- AbstractTestXmpp
- TestXMPPAssertion
- TestXMPPAssertionList
- TestXMPPQuery
- TestXmppQuery
- TestXmppResults

Sample Test Data

```
<list>
    <assertion subject="piro@megatokyo-example.com" category="person">
        <field key="name" type="string">Piro</field>
    </assertion>
    <assertion subject="largo@megatokyo-example.com" category="friend"/>
    <assertion subject="largo@megatokyo-example.com" category="liveswith">
        <field key="place" type="string">tokyo</field>
    </assertion>
    <assertion subject="erika@megatokyo-example.com" category="coworker">
        <field key="place" type="string">megagames</field>
    </assertion>
    <assertion subject="ping@megatokyo-example.com" category="owns">
        <field key="foryears" type="int">2</field>
        <field key="how" type="string">friend</field>
    </assertion>
    <assertion subject="kimiko@megatokyo-example.com" category="teachers">
        <field key="about" type="string">manga</field>
    </assertion>
</list>
```

19.4. Experience Gathered

The prototype provided useful information as to:

- The data transfer objects available in the Smack API.
- The failure modes of XMPP transport, and the required recovery actions.
- How to initialise the available encryption systems in XMPP.
- The importance of version control.

19.5. Conclusion

The prototype performed its function well. It successfully stored and published assertions designating realistic relationship between people. This proved the viability of the design and the XMPP technology.

20. Project Conclusion

This project has been successful in its overall aims. The concept was well conceived, and the scheme-of-work specified the project well, giving it a coherent structure. Research was undertaken, not only into the background of social networking systems, but also the into the *technology and tools* that could be used to create an innovative design.

It also has a good chance of being realised, since the author intendes to prepare a request-for-comment to the Internet Engineering Taskforce committe responsible for XMPP. A project website hosted on the SourceForge system was created to document Hearsay. It is accessable at **http://hearsay.sourceforge.net/** .

Research performed

- Basic historical background
- Overview of current marketplace
- Working client/server with security
- Extensive IM technology research

Analysis and Management This work provided the starting point for effective *problem analysis* of the *technical* issues of social networking systems software.

Project management was good. The work was spread evenly over the 7 months allowed, and so there was no 'last-minute panic'. Setting a structure of work early in the project was the key to achieving this.

Clear Design The software system design is well documented. All the design decisions are clear and follow from previous work. The prototype documentation is insufficiently

detailed, although it did succeed in proving feasible both the concept and chosen technology.

Tools used

- Eclipse IDE
- Concurrent Versioning System (CVS)
- Standard Modelling Language (SML)
- Smack Messaging API
- JUnit
- XML Messaging

Methodologies used

- Prototyping
- Functional Languages
- Enterprise Design Patterns[36]
- Regular Lanuages and Abstract Syntax Trees

Learned Skills

- Time management
- Systematic Testing
- Network application programming
- Practical application of regular languages

Observations

- XML is just as messy as any other serialisation, so do not let it get unmanagable.
- Some theory will ultimately apply to any problem, sometimes it is hard to know which.
- Programmers do it better with a blank piece of A4 and a pencil !

Summary and Critique The author's main criticism of the project would have to be 'scope creep'. Although, well focused, this project has tried to cover too much ground for the time given. It would have been better to concentrate a narrower topic: such as 'A Friendster-clone in J2EE'. As it stands, the project does tackle the issue it set out to answer.

A great deal was learned in the course of this project. The personal satisfaction of being able to learn independently within a young field is what makes Computer Science such a rewarding degree to study.

21. Future Work

21.1. Roadmap

> "*In theory, there is no difference between theory and practice. But, in practice, there is.*"
>
> - Jan L.A. van de Snepscheut

In order to be deployed in the real world, the implementer will need to complete:

- Add additional standard predicates.
- Build the full spider and query interpreter.
- Test for high-loads in real network conditions.
- Investigate trustworthy execution. Not just safer, but safe.

21.2. Research Projects

There are many ways in which the system outlined discussed here could be used as a basis for future work. These are five outlines of projects that the author believes would particularly beneficial.

21.2.1. Porting a Web Application

Description Implement, using the techniques described in this dissertation, one of:

- Personals
- Online Auctioneering
- Academic Peer Review

Any existing libraries or systems can be used, you need not start from a bare base.

Issues

- Is costly signalling necessary for your application? How can it be provided?
- Estimate transactions costs for an individual using your system?
- Does the system provide benefits over a centralised approach?

Hints

- Look at existing web-based systems.
- Use the XMPP data-form and HTML standards from GUI work.

21.2.2. Costly Signalling in Business

Description The process of costly signalling has many uses in social and business situations. Research ways in which costly signalling can be imposed on a peer-to-peer system. Investigate the changes in dynamics that this might cause.

Issues

- Are people's perceptions of one another altered?
- Can costly signalling reduce spam?
- Will money-based costly signalling necessarily exclude the disadvantaged?

Hints

- Complex calculations can be used to impose costs.
- Digital money techiques allow the creation of unforgeable tokens.

21.2.3. Complex Network Analysis

Description Perform analysis of a large existing body of social network data, using a variety of tools and methods. Show how meaningful conclusions can be made about data by applying techniques available to end-users.

Issues

- If performed on a live system, how would your analysis affect the anonymity of the users?
- What are the most effective forms of graph analysis?
- At what scales do clusters and sub-clusters form? What are the basis for these clusters?

Hints

- Look at Yenta's keyword clustering algorithms
- Explore the Avocado Trust Metric

21.2.4. Defeating Traffic Analysis

Description Investigate the effectiveness and efficiency of different traffic data masking techiques. Attempt to establish a link between the level of privacy gained and the proportion traffic wasted .

Issues

- Is 'Onion Routing' appropriate? How can it be implemented?
- Can near perfect privacy be achieved at an acceptable cost?
- How is the effectiveness of the monitoring affected by its duration?

Hints

- Try looking at the Invisible Internet Project[49]

21.2.5. Multicasting with JXTA

Description Build a functioning subset of Hearsay within the JXTA environment. There's is no need to use XML, Java serializations are acceptable.

Issues

- How does JXTA multicasting affect scalability?
- Does the structure of JXTA groups help Hearsay-style architectures?

Hints

- Existing Pipe objects can provide secure transport.
- Look into nestled groups as a clustering mechanism.

Bibliography

[1] Granovetter, Mark. Strength of Weak Ties, American Journal of Sociology, volume 78, 1993.

[2] Schneier, Bruce. Applied Cryptography: Protocols, Algorithms, and Source Code in C, John Wiley & Sons, 1996.

[3] Headmap Manifesto, Headmap.org, 1999.

[4] Havrilesky, Heather. Meatmarket.com, Salon Life, May 2002.

[5] Krim, Jonathan. Spam's Cost To Business Escalates, Washington Post, 2003.

[6] Graham, Paul. A Plan for Spam, http://www.paulgraham.com/spam.html, August 2002.

[7] Junnarkar, Sandeep. Instant messages become new path to victims' PCs, New York Times, March 2004.

[8] Wikipedia. Eternal September, http://en.wikipedia.org/wiki/Eternal_September, Accessed April 2004.

[9] The Huminity Company Website, http://www.huminity.com/, Accessed April 2004.

[10] Dean, Kari Lynn. 'Will Microsoft Wallop Friendster?', Wired News,November 2003.

[11] Kahney, Leander. Making Friendsters in High Places, Wired News, July 2003.

[12] Terdiman, Daniel. Friendster Quickly Gathering Foes, Wired News, November 2003.

[13] Tribe.net Website, http://tribe.net, Accessed April 2004.

[14] LiveJournal Website, http://livejournal.com, Accessed April 2004.

[15] Berners-Lee, Tim. Hendler, James. Lassila, Ora. The Semantic Web, Scientific American, May 2001.

[16] Global Multimedia Protocols Group, Introduction to XFN, http://gmpg.org/xfn/intro, Accessed April 2004.

[17] Brickley, Dan. Miller, Libby. FOAF Vocabulary Specification, http://xmlns.com/foaf/0.1/, March 2004.

[18] Rubhub, http://rubhub.org

[19] People Finder, http://pfinder.com

[20] Ellison, Carl. SPKI/SDSI and the Web of Trust, April 2001.

[21] CNN News. Justice Department drops probe of encryption programmer, CNN News Briefs, January 1996.

[22] Zimmermann, Phil. Introduction to Cryptograph, Network Associates Inc., 1999.

[23] Zimmermann, Phil. Why do you need PGP? .

[24] Foner, Leonard Newton. Political Artifacts and Personal Privacy: The Yenta Multi-Agent Distributed Matchmaking System, PhD. at the Massachusetts Institute of Technology, 1999.

[25] Kautz, Henry. Selman, Bart. Milewski, Al. Agent Amplified Communication, Applications of Artificial Intelligence Conference, Proceedings of the Thirteenth National Conference on Artificial Intelligence, 1996.

[26] Chalmers, Rachel. Even better than Slashdot?, Salon.com, July 2000.

[27] Advocado Website, http://www.advocado.com, Accessed April 2004.

[28] Spinner, Jackie. 'Friendster' Counting Culture, Washington Post, September 2003.

[29] FOAF Explorer, MFD Consult, http://xml.mfd-consult.dk/foaf/explorer/, Accessed April 2004.

[30] Schneier, Bruce. Secrets and Lies: Digital Security in a Networked World, John Wiley & Sons Inc, September 2000.

[31] Ullman, Jeffrey D. Elements of ML, Prentice-Hall, 1997.

[32] Graph Theory, Wikipedia Entry, Accessed April 2004.

[33] Houston, Gary. ISO 8601: 1988 Date/Time Representations, [ftp.informatik.uni-erlangen.de/pub/doc/ISO/ISO8601.ps.Z], Accessed April 2004.

[34] Shkapenyuk, Vladislav. Suel, Torsten. Design and Implementation of a High-Performance Distributed Web Crawler, ICDE, 2002.

[35] Deutsch, Peter. The Eight Fallacies of Distributed Computing, 1992.

[36] Fowler, Martin. Patterns of Enterprise Application Architecture, Addison-Wesley, 2003.

[37] JXTA Website, http://jxta.org/, Accessed April 2004.

[38] Adams, DJ. Programming Jabber, O'Reilly, 2001.

[39] XMPP Website, http://jabber.org, Accessed April 2004.

[40] Jive Software Website, Jive Software, http://www.jivesoftware.com, Accessed April 2004.

[41] Apache Licence, Apache Foundation, http://www.apache.org, Accessed April 2004.

[42] SSL 3.0 Specification, Netscape Corporation, 1996.

[43] Rheingold, Howard. Smart Mobs, Random House, 2002.

[44] USA Patent 6,175,831, United States of America Patent Office, January 2001.

[45] Friedl, Jeffrey E. F. Mastering Regular Expressions, 2nd Edition, O'Reilly, July 2002.

[46] Common API for XML Pull Parsing, http://xmlpull.org/, Accessed April 2004.

[47] Sosnoski, Dennis M. A look at features and performance of XML document models in Java, IBM Developer Works, September 2001.

[48] Code Conventions for the Java Programming Language, Sun Microsystems, 1999.

[49] Invisible Internet Project (I2P). Project Overview, http://www.invisiblenet.net, August 2003.

List of Figures

List of Tables