A Peer to Peer Supply Chain Network

by

Bradley Charles Goldsmith
BCompSc MACS MIEEE MACM

A dissertation submitted to the
School of Computing
in partial fulfilment of the requirements for the degree of

Master of Computing

June 2004

Declaration

I, Bradley Charles Goldsmith, certify that this thesis contains no material which has been accepted for the award of any other degrees or diplomas in any tertiary institution, and that, to the best of my knowledge and belief, this thesis contains no material previously published or written by another person except where due reference is made in the text of this thesis.

Signed: ....

Date: ....

Abstract

Many papers have speculated on the possibility of applying peer-to-peer networking concepts to networks that exist in the physical world such as financial markets, business or personal communication and ad hoc networking. One such application that has been discussed in the literature has been the application of peer-to-peer to corporate supply chains to provide a flexible communication medium that may overcome some classical problems in supply chain management.

This thesis presents the design, development and evaluation of a system which implements a peer-to-peer supply chain system. A general, flexible peer-to-peer network was developed which serves as a foundation to build peer-to-peer data swapping applications on top of. It provides simple network management, searching and data swapping services which form the basis of many peer-to-peer systems. Using the developed framework, a supply chain focussed application was built to test the feasibility of applying peer-to-peer networking to supply chain management.

Results and discussion are presented of a scenario analysis which yielded positive results. Several future directions for research in this area are also discussed.

Acknowledgements

In no order of importance:

I would like to thank my supervisor, Daniel Rolf, for his supervision, assistance and good humour during the course of this dissertation.

Jacky Hartnett for her advice regarding results processes and security features.

The Beatles, David Bowie, Bill Hicks, Joe Jackson, Cake and Quentin Tarantino for his soundtracks. These albums were played constantly during this exercise.

Alistair Atkinson whose advise on many occasions, and over several drinks, proved invaluable for sounding out ideas.

All of the open source contributors that have made OpenOffice a trouble free tool with which to produce this thesis.

Douglas Wright, my long suffering business partner, whose generosity at significant financial expense, allowed me to take the necessary time from work to complete this thesis. Thanks Doug.

And my dear partner, Emma Kirsopp, who provided love and understanding whilst ensuring that I was regularly fed, groomed and walked.

Contents

1.0 Introduction

If we knew what it was we were doing, it would not be called research, would it?
-- Albert Einstein

This chapter provides an introduction to the research conducted and the motivating factors that have encouraged its undertaking. A summary of what the thesis contributes is provided as is an overview of the document structure.

1.1 Motivation

A supply chain is a network of suppliers, manufacturers, wholesalers, transport agents and retailers involved in the manufacture, marketing, distribution and retailing of a product or service. A single product or service is often the result of many organisations working together to produce the component parts, assemble them and deliver the resulting product to the customer. As separate organisations are involved, there is inter-organisational communication in the form of purchase orders, stock availability queries, back orders, etc. This communication, when poorly organised, can lead to shortages in raw material, oversupply of inventory, poor lead times on manufacturing and a host of related problems. Finding new vendors or suppliers of a product or service within a supply chain and integrating them effectively can often be a problem onto itself.

As information technology has evolved it has become an irreplaceable part of the communication of information between organisations along a supply chain. Recent trends towards globalisation and geographic separation between cooperating organisations have made electronic communication even more essential for efficient production. However, when dealing with a variety of organisations a broad spectrum of technical sophistication is likely to emerge making it difficult to effectively communicate with some members of a supply chain. Inconsistencies in stock and control systems used between different organisations make it difficult to obtain timely information when a source of the information is separated by several organisations.

Supply chain problems regarding coordination, redundancy, discovery of vendors and accuracy of information has led many organisations and researchers to look towards Internet based solutions to information management in supply chains. Some of these research groups have considered peer-to-peer networking as a possible candidate for e-commerce however few open implementations exist.

This lack of concrete application and open system availability for research has provided the motivation to undertake this work.

1.2 Overview

This thesis presents research into the design and implementation of a peer-to-peer network system that can be used to manage a supply chain across several organisations.

A development framework for general peer-to-peer research is also provided allowing the rapid application development of peer-to-peer programs that are not tied to any specific application.

An evaluation of the peer-to-peer application to supply chains is presented through a scenario analysis of a supply chain simulation taken from management literature. This scenario analysis, in lieu of an actual commercial trial, is used to determine how well the peer-to-peer supply chain meets its stated goals.

1.3 Contribution

The contributions of this thesis include:

1. A survey of existing peer-to-peer applications, frameworks and protocols.
2. A simple to modify and extend open, decentralised peer-to-peer framework.
3. An implementation of a cut down Public Key Infrastructure system within a peer-to-peer context supporting confidentiality, authentication and certificate revocation.
4. An implementation of a supply chain management solution based on an open peer-to-peer network.
5. An implementation of an example scenario to illustrate how this supply chain management solution can be used to solve several classical supply chain issues.
6. A critical discussion of the strengths and weaknesses of a peer-to-peer solution to managing a supply chain.

1.4 Thesis Structure

Chapter two reviews existing work that relates to this research. As this thesis is merging several different areas of research to provide a result, the literature review covers a wide variety of topics. Supply chain management, peer-to-peer networks, electronic commerce, security and cryptography and data exchange are considered.

Chapter three is in two parts. Firstly, it describes the peer-to-peer framework that was developed to facilitate the supply chain research and then the scenario used to provide supply chain results is described as is the application built to test it.

Chapter four is in several parts. Results and discussion of the peer-to-peer framework are given as are comparisons between it and the commonly available Gnutella protocol. An evaluation criteria for selecting a peer-to-peer network for an application was also developed and is described here along with an appraisal of the security features required to support a supply chain. The results for the scenario analysis as developed in chapter three are presented here and discussed.

Chapter five concludes the thesis with a discussion of the research and how well the system developed achieves its intended goals of successfully implementing a supply chain in a peer-to-peer network. Several directions for future research are also presented in this chapter.

The appendices contain the source code for the peer-to-peer framework and supply chain application. The simulation data is also included as is a brief discussion regarding the classification of the search strings observed on the Gnutella network.

...

2.0 Literature Review

The best way to become acquainted with a subject is to write a book about it.
-- Benjamin Disraeli

This chapter presents a review of current literature relevant to this peer-to-peer supply chain investigation. Supply chains and the issues associated with them are presented as are a brief discussion of the software available for electronic supply chain management. Peer-to-peer networking is described with a focus on this endeavour. Electronic commerce, commerce relevant security and data exchange methods suitable for business are also discussed.

2.1 Supply Chains

Yucesan & Wassenhove (2002) define a supply chain thus: “A supply chain is the network of retailers, distributors, transporters, storage facilities and suppliers that participate in the sale, delivery and production of a particular product.” A supply chain consists of all efforts involved in creating and delivering a product or service. These chains often consist of many different companies ranging from sourcing primary materials, manufacturing to distribution, warehousing and retailing.

Supply chains have become more important as the focus has shifted from the manufacturer dictating supply and availability to the customer driving what should be available in a particular market. Also, parity in quality between competing firms has left service as being the attribute in which a customer uses to distinguish between choices. This is where a firm's supply chain becomes critical in ensuring that a product is manufactured and delivered in as timely a manner as possible.

It becomes clear that information is one of the most important aspects of a supply chain; both its collection and its distribution. Information is collected and stored along each part of the supply chain in both a formal and informal manner, depending on the sophistication of the supply network as a whole. How well this information is communicated can greatly affect the operation of the supply chain. Suppose a manufacturer traditionally reduces output during a particular month of a year due to a market peculiarity that has existed for some time. However, during this month, a retailer of the manufacturer's product receives an unexpectedly large volume of orders. The retailer might be able to successfully notify the wholesaler, who in turn informs the manufacturer by way of continued orders for this product and so continue to supply demand. However, with many firms involved in this communication of information, problems will inevitably occur. Perhaps the retailer only makes their orders at the last moment and the information is too late by the time that it reaches the manufacturer for production to increase. Or, the wholesaler does not communicate the information to the manufacturer because they do not feel that it is important, or worth their trouble, or for any other reason. Perhaps the company receives this information within time and acts on it only to find that the information turned out to be unreliable as the retailer cancelled their orders after all.

These kind of problems do not exist in all supply chains. These descriptions are to illustrate the potential issues that can, and often do, arise. We will now examine the concepts of a supply chain and the common problems and issues that supply chains have.

2.1.1 Supply Chain Concepts

A supply chain encompasses a wide variety of processes with an organisation or a group of organisations.

The concept of a supply chain includes the following members and functions (Houlihan 1985):

- The supply chain describes the process of providing goods or services to the end user.
- It includes all organisations from supplier to customer within a single system.
- The scope of a supply chain covers procurement, production and distribution.
- The supply chain can extended across several organisations.
- The supply chain is managed through a information system accessible to all members.
- Balanced across costs and assets, the primary objective of any supply chain is to service a customer.

2.1.2 Supply Chain Tasks

Schary & Skjott-Larsen (1995) describe five basic processes that a supply chain will comprise of:

Product Design

This phase involves the actual design of a product to meet a market's demands. This process involves the design of the actual product itself, production engineering and any required packaging.

Procurement

Procurement is the sourcing and purchasing of raw materials and components that are assembled into the final product. This process involves the formation of relationships with suppliers of materials or services.

Production

Production is the process that transforms our raw materials from the production process, into the final product that is to be taken to market.

Demand Management

Demand management involves controlling production based on the number of actual or predicted orders.

Distribution

This is the logistical process of getting the finished product to market or distribution of supplies and inputs between supply chain members.

2.1.3 Classical Problems in Supply Chain Management

Now that a basic understanding of the major elements and functions of a supply chain has been established, we shall look at some of the common problems that supply chains experience as defined by Savit (1999).

- Information that flows through the supply chain is often late or wrong.
- Effective planning is difficult or impossible in a distributed multi-firm enterprise.
- Supply chain dynamics impact on both delivery and cost.
- Suppliers usually can't trust forecasts and don't trust forecasters.
- Suppliers can lack technical sophistication.
- Support technologies are inadequate or non existent.

- Standards are lacking for data interchange and enterprise control.

These issues can be more generally stated thus:

- Supply chains consist of heterogeneous sets of agents, each acting on the basis (typically) of incomplete information about the chain as a whole.
- Motivations are generally local to each supply chain node e.g. Maximisation of only profits and costs are only important to a single node. All other considerations to other nodes in the supply chain are either secondary or completely ignored.
- Supply chains are intrinsically dynamic.
- Agents are adaptive.
- There are non-linearities and feedback effects in dynamics and structure of most supply chains. The is often referred to as the "bull whip" effect.

These issues will need to be of importance to any system that attempts to model or assist a supply chain management.

2.1.3.1 The "Bull Whip" Effect

The Bull Whip effect is a frequently observed problem in supply chains that relates to oscillation in orders caused by demand variability or supply uncertainty. This distortion of demand information in the supply chain can cause difficulties in effectively estimating stock or production levels to meet demand.

This effect is also commonly seen to be amplified as it travels higher in the chain from retailers, wholesalers, manufacturers and supplies (Chopra & Meindl, 2004. pp 478).

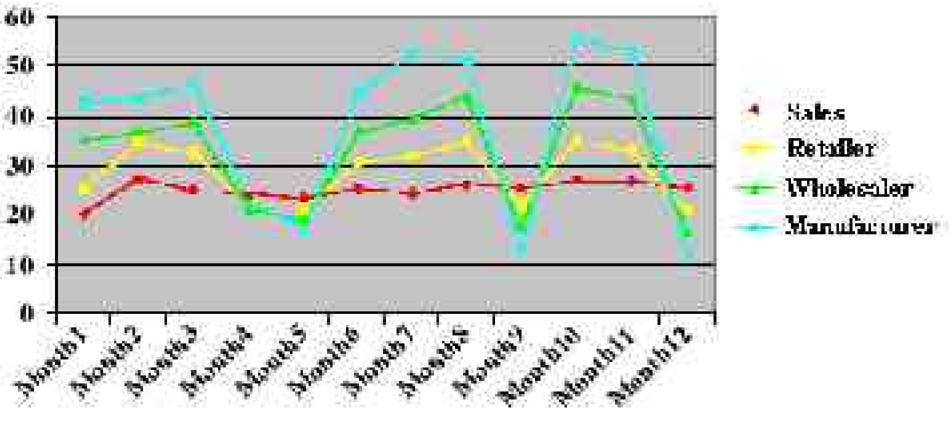


Figure 2.1: The bull whip effect on orders travelling upwards through the supply chain (Carlsson & Fuller 2004).

2.1.4 Existing Supply Chain Software

Many companies have realised for some time that supply chain management is something that can be assisted with computer software. As such, there are many offerings in the market today utilising a variety of methods for inter-organisational communication. Some use “extranets” which are essentially private Internet sites that can have multiple logins for different companies where they can all collaborate on issues relevant to their businesses. These sites, whilst a step in the right direction, tend to be instigated by one member of the supply chain where other members are obliged to use it. This improves supply chain management for the instigator, but might even be considered a hindrance (due to instigator centric implementation) by supplier nodes in the chain.

Several companies exist that produce applications that can be customised to manage supply chain communication between organisations. Firms such as SAP, Bea Systems and Manguistics have been actively providing solutions in this area for some time. At time of writing, none are offering any agent or peer-to-peer based solutions that are available for evaluation.

2.2 Peer-to-Peer Technology

The central idea in a peer to peer system, is that each node on a network is equivalent in function to every other node. Whilst many peer-to-peer systems do not strictly adhere to this principle, they are not in a client-server relationship which would typify many other networked systems. Peer to peer networking is not a new idea. A work group of Windows^{TM} machines sharing files are each peers on a network. Each machine shares files, acting as a server, and each machine accesses other machine's files, acting as a client. Earlier, and even more ubiquitous examples, include the Domain Name System (DNS) and the Network News Transfer Protocol (NNTP). These were both developed in the mid 1980's and share data in a peer to peer fashion.

What is relatively new in peer-to-peer systems are small, single-user peers existing on a wide area network rather than a local one when sharing resources. These peering networks are commonly used to share the resources of these machines for storage, processing or collaboration. This section will examine the commonly available protocols in each genre.

2.2.1 File Trading Systems

File trading peer-to-peer networks can be generally placed into one of three categories; centralised server, distributed server and swarm. Centralised server systems typically have a central servers which maintain a database of shared files and information about the peers that host them. Centralised systems provide an efficient means of search at the expense of a single point of failure; the central servers themselves. Distributed systems spread the search data across the nodes themselves. This produces a network that is harder to control or prevent, however these systems often suffer from the overhead of messages required to maintain network topology. Swarm networks are defined as a collection of agents that execute a schedule of actions (Minar et al 1996). There exists peer-to-peer networks that closely follow this paradigm, by having downloads that begin with a single *seed* with other nodes quickly joining the network which then move across the network as new nodes join and others leave.

Napster will be considered for an example of a centralised peer-to-peer network. Two popular examples of distributed networks, kazaa and gnutella will be shown. BitTorrent will be given as an example of a swarm network.

2.2.1.1 Napster

Napster is considered to be the first popular peer-to-peer file sharing network. The original network itself no longer exists having been shut down after legal action by several interests. There are open source initiatives that have implemented the Napster protocol (OpenNap 2003). These new implementations do not approach the network size, in terms of active users, that the original Napster achieved.

The success, and ultimate failure, of the Napster network can be attributed to its centralised search design. This allowed file searching to be directed at known load-balanced servers. As clients came online, they would report information to the central server about what files that the node has available for sharing. This registration process allowed for an accurate view of available files and clients on the overall network. File transfer would still take place between client nodes on the network, but the discovery of these nodes was via central Napster servers. This central server approach for searching made searching accurate and efficient when compared to most decentralised systems. However, the central server paradigm also presented a single point of failure, one that made it relatively easy to shut the service down. Subsequent peer-to-peer mechanisms have avoided total centralisation wherever possible.

2.2.1.2 Gnutella

Gnutella (pronounced "new-tella") is a distributed search network. It consists of a distributed network of nodes called servents. These servents function as both a <u>serv</u>er and a cli<u>ent</u> and exchange both search and network information between each node in the distributed network.

Each node maintains communication with other nodes essential for the maintenance of the network itself. These messages exist to distribute information about what files the node is sharing and information regarding its logical address on the network. A description of the protocol messages from the protocol specification (Gnutella 2003) is given thus:

Ping - A ping message is used to discover other servents on the network. When another servent receives a ping message it replies to the sender with a pong message.

Pong - The reply to a Ping message. This message includes the TCP/IP address and port of the sender and how much data it is sharing on the network.

Query - This is a search query packet. If the packet matches data being served by the servent, the servent will respond with a QueryHit packet.

QueryHit - This is the reply to a query packet that details the required information to allow the querying servent to transfer the data from the sending servent.

Push - This packet allows servents behind a firewall to request that a file to be transferred be "pushed" to the client. This mechanism allows for ports other than the standard Gnutella port to be used for data transfer.

Each Gnutella servent has a 128-bit number that is used to uniquely identify a node in the Gnutella client. Other hosts on the network are discovered either passively or actively. Passive discovery of other hosts happens when routing, or responding to, packets from other nodes on the network. Active discovery happens on the receipt of pong packets from nodes after broadcasting a ping packet.

When a search is performed, a search packet containing the search string is sent out to the servent's connected nodes. Each recipient of the search packet looks in its own database of shared files to see if the search term is matched. The recipient then decrements the packets time-to-live (TTL) and forwards it on to each on of its Gnutella connections. If a recipient finds that the search term is matched with its files, it sends a *query-hit* packet, which contains the recipients connection details, back along the path that it received the successful query. This query hit packet back propagates back through the network and eventually reaches the originating node. On receipt of the query hit, the originating node attempts to make a direct connection to the replying node. If a direct connection cannot be made, another packet called a "push" can be sent through the Gnutella network to the recipient node to request that they "push" the data to the requesting node. This is used to circumvent blocking when one of the communicating nodes is behind a firewall.

Gnutella is a popular peer-to-peer client that is gaining popularity with both file sharers and researchers alike. Research has suggested optimisation of the network topology and the minimisation of network overhead caused by packet flooding (Ripeanu 2001).

2.2.1.3 Kaaza (FastTrack)

Kazaa is a peer to peer protocol that performs both searching and downloading via other nodes in the network. Where faster computers with higher bandwidth connections to the Internet are detected, these are promoted to a *supernode*. Supernodes contain information about the files that they are sharing, but also for other nodes nearby on the network. This attempts to increase search coverage while minimising connections between nodes and bandwidth.

The actual protocol itself, called FastTrack, is proprietary. However there have been attempts to reverse engineer this protocol. The MLDonkey (MLDonkey 2003) peer-to-peer client reverse engineers a number of file sharing protocols including FastTrack, however fully accurate and reviewed information does not appear to be available at time of writing. Marcus Bergner's master thesis (2003) gives a good description of what is known of this proprietary protocol and some results on its efficiency compared with other protocols.

2.2.1.4 BitTorrent

BitTorrent (Cohen 2003) is a novel approach to online file sharing. Traditionally, when downloading a file a user only uses the download channel of their Internet connection whilst the upload channel is mostly unused. BitTorrent takes advantage of this and allows other users to download the same file that another user is downloading at the same time. This allows for a user to download a file from a user without affecting their download speed.

Rather than the focus being on maintaining a network for search, as most peer-to-peer file distribution systems do, BitTorrent concentrates on maximising the bandwidth available to its members and therefore does not provide its own search facility. BitTorrent relies on existing search methods available to the web to find a “metainfo” file which contains the information required to join a BitTorrent session.

This metainfo file is typically hosted on an ordinary web server and is recognised by it's `.torrent` extension.

The metainfo file contains information about the file that is being downloaded, such as its length, name, number of pieces the file has been split into for distribution, hashes of these pieces, and the URL of a “tracker”. A BitTorrent tracker is a small HTTP based service that allows BitTorrent downloaders to find other downloaders of the same file and connect to them. When a BitTorrent download is started, information about the downloader such as their IP address, BitTracker listening port and so forth, are sent to the tracker. The tracker then responds with the same kind of information about others who are downloading the file. This is how the BitTorrent network is maintained.

BitTorrent also employs many techniques to ensure that the transfer between two nodes is efficient as possible. This is an to attempt at pareto efficiency, making node to node transfer as efficient as possible, to make the network as a whole as efficient as possible. Such techniques include pipelining of HTTP requests, order selection and prioritising of download chunks, choking algorithms and anti-snubbing of peers. See Cohen (2003) for more information.

2.2.2 Existing Peer-to-Peer Frameworks

A brief discussion of two existing peer-to-peer frameworks, JXTA and P-Grid, are given. Whilst there are several more in existence than the two presented here, these chosen for description due to their continued research output, development and proven availability.

2.2.2.1 JXTA

JXTA is a recent initiative from Sun to produce an open collaboration platform that will allow a wide range of distributed applications to operate over it. JXTA is a suite of services that attempt to provide a general purpose distributed computing and networking platform (Milojicic et al 2002).

JXTA provides lower level services than other existing peer-to-peer systems by providing a network for peer-to-peer applications to run on. Each JXTA network application is given a group to operate within. This allows separate peer-to-peer applications to be partitioned on the network. Each node that operates within a group is assigned a unique ID that allows it to be uniquely identified on the network. Data exchanged on the network is in the form of XML documents.

JXTA provides an instant messaging application as an example implementation with its framework. Other applications available on this network include peer-to-peer email, file sharing and content management.

2.2.2.2 P-Grid

P-Grid is a decentralised peer-to-peer system based on a virtual distributed search tree (Aberer et al 2002). Each node in a P-Grid network holds a part of the overall tree which is maintained through the cooperation of each node on the network. Maintaining a search tree allows P-Grid to provide probabilistic guarantees for search times, scalability and bandwidth requirements. This differs from other decentralised networks, such as Gnutella, where nodes are mostly unregulated and search times and scalability are unmanaged.

The source code for P-Grid is now available to other peer-to-peer researchers on request.

2.2.3 Peer-to-peer distributed storage systems

2.2.3.1 Freenet

Freenet is a secure, anonymous, distributed file storage system (Clarke et al 2002). Peers wishing to join the Freenet, download an run a client on their machine. Each peer contributes disk space to the overall network as a whole. The aim of Freenet is to provide a information network on top of the Internet that is highly resistant to censorship attempts.

All communication on Freenet is via strong encryption. Searches are passed from node to node in such a way that it cannot be determined where the search originated from. This means that replies to the search are also copied back from node to node and re-encrypted at each point. This allows popularly requested documents to propagate throughout the network and be readily available. Less popular documents will eventually be removed from the network as nodes leave.

This copy through process is designed with a legal defence in mind. Should interested parties take the time to decrypt the contents of your nodes file store, reasonable doubt exists as to whether you had requested the contents or is it simply the results of passed through search replies.

2.3 Electronic Commerce Systems

There are a variety of electronic commerce systems available on the Internet today. A brief exploration of the genres that would be exploitable by a peer-to-peer implementation is given below.

2.3.1 online Shops

2.3.1.1 Business-to-Customer

Business-to-Customer sites are concerned with offering products and services to the general Internet public. These B2C sites include shopping sites, subscription based news services, Internet banking and a myriad of other sites typified by a high volume of public users.

2.3.1.2 Business-to-Business

Business-to-Business sites encompass existing Internet based corporate collaboration sites as well as wholesalers and distributors that maintain online shops for retailers to make orders. B2B websites are becoming more common in a variety of industries.

2.3.1.3 Customer-to-Customer

Customer-to-Customer sites are defined by those who have products and services being exchanged between two parties where either party is not necessarily a business or organisation. Common examples include auction sites such as eBay or Yahoo! Auctions.

2.3.2 Agent Based Commerce Systems

There are several agent based commerce applications that exist in the academic and commercial domains. Commercially available systems tend to be price comparison engines such as Frictionless e-Market Suite or pricescan.com. The price comparators are essentially web based agents that crawl the web to find products and their prices and index them so that they can be searched and compared.

There are many recent papers that discuss the possibility of intelligent agent based commerce and some of the tools that would be required to facilitate it. A recent masters thesis describes an implementation of an agent based commerce system. This system, Atomic Market, is discussed below.

2.3.2.1 Atomic Market

Atomic Market is described in the Master's Thesis of James Youll (2001). This system is best described by Atomic Market's web page (MIT Media 2001):

“We propose a new type of electronic marketplace, which we refer to as an 'atomic market.' Atomic markets differ from today's electronic marketplaces in that they are open-ended, decentralized and component-based. The atomic market supports short-lived markets created around the individual components of everyday transactions. The traders in an atomic market are agents, software that acts as a proxy for an actual buyer and seller.

The atomic market allows expressive interactions among trading agents, leading to productive, automated agent-based transactions. The focus is on the technical infrastructure for atomic marketplaces, specifically the use of logic as a basis for the decomposition of transactions and the negotiations between the different agents.”

The atomic market is a message based system that operates over secure HTTP within a framework called RosettaNet (See chapter 2.5.2.2). Nodes on the network exchange messages in order to attempt a commercial exchange.

2.4 Security in Electronic Commerce

In order for peer-to-peer commerce to work effectively, security concerns are as vital and the implementation of the peer network itself. Without the ability to foster a reasonable amount of trust within the network, it is likely that it will fail. In normal, real world situations, trust is established through various means including the reputation of a company, the length of time the business has been trading, recommendations from other people, knowledge based on previous experience, etc.

Electronically, these kind of issues also need to be addressed, as well as problems such as authentication, accountability, non-repudiation, authorisation and confidentiality (Datta et al 2003).

2.4.1 Authentication

The peers on the network, and the corporate entities behind them, must be correctly identified by other peers on the network. A widely accepted method for achieving this electronically is through the use of digital signatures in the Public Key Infrastructure (PKI).

2.4.2 Accountability

Openness and accountability are the hallmarks of any system that relies on trust. Past actions of peers need to be accessible to other peers on the network to aid with deciding whether to do business with this peer or not. eBay uses a system of feedback where buyers and sellers are able to provide comments and ratings to describe their experiences in a transaction. Whilst this system can be open to abuse, the comments system allows for a right of reply as well – so unreasonable comments from one party are often accompanied by an explanation (or exclamation) from the receiver. Aberer and Despotovic (2002) describe a system suitable for peer-to-peer trust management.

2.4.3 Non-repudiation

Non-repudiation is a term given to the ability to provide undeniable proof of an entity performing an operation. In a commerce example, non-repudiation is essential to be able to ensure that orders can not be later denied by the ordering party. Digital signatures are used to ensure this.

2.4.4 Confidentiality

Peers on the network may contain information that is not be be made public to other peers on the network. This confidentiality must be ensured to allow an organisation to trust the network. Again, PKI provides the tools necessary to achieve this.

2.4.2 Public Key Infrastructure

The Public Key Infrastructure is a combination of encryption technologies such as public key cryptography and digital signatures, and third party organisations to provide security services to online transactions. The third parties exist in order to provide registration and certification services to add non-repudiation to public key cryptography and digital signature systems. The registration authority (RA) works with an organisation who wishes to use PKI to correctly and accurately identify itself to the RA. The RA then informs the certification authority (CA) who issues a X.509 certificate to the requesting organisation to allow them to work within PKI. CA's also provide run time authentication of an issued certificate to ensure that once the certificate is issued, it retains its integrity.

For an exhaustive discussion of PKI, see the PKI Infrastructure Charter (IETF 2003).

2.5 Data Exchange Paradigms

There are many ways for corporate entities to exchange data electronically. These range from proprietary protocols to open initiatives. The extensible markup langugaue will be briefly described as a popular, flexible method of exchanging data electronically. Two industry sponsored data exchange frameworks will also be explained.

2.5.1 eXtensible Markup Language (XML)

The eXtensible Markup Language (XML) is a subset of the Standard Generalised Markup language that was invented at IBM in various stages from the late 1960's through to the 1970's. XML is a derivative of SGML that shares the same aims of describing a document in an easily parseable and portable manner.

A structured XML document is made up of text, tags and attributes. The text is the information itself whilst the tags and attributes describe the information within the document. The intention of XML is to make the transfer of information across the Internet simple and flexible through formal definition of documents.

Consider the example of an address. If it were to be provided as information only:

```
PO Box 1643
Launceston TAS 7250
Australia
```

The XML example below shows how the same information is presented in a more usable manner where the document itself describes the information contained within.

```
<address>
      <street>PO Box 1643</street>
      <town>Launceston</town>
      <postcode>7250</postcode>
      <state>TAS</state>
      <country>Australia</country>
</address>
```

XML has supporting technologies such as document definition schemas and transforms which allow XML documents to be rigidly defined and manipulated.

See the eXtensible Markup Language homepage (W3C 2003) for many XML resources.

2.5.2 Business Oriented Initiatives

Here is a brief examination of the standards and initiatives that exist for the general and commerce specific exchange of data for business.

2.5.2.1 Internet Open Trading Protocol

The Internet Open Trading Protocol (Burdett 2000) is an initiative to develop a standard, global framework for Internet commerce. It is largely designed around emulating how trading, buying and selling operate in historical terms, that is, by methods that are widely accepted in traditional business exchanges.

The protocol functions by sending IOTP messages between the various parties involved in a business exchange. Each of these messages are actually XML documents whose schemas are well defined in the standard.

Whilst a promising standard, it concentrates on two-party transactions that occur in a rigid framework. It would not be obviously adaptable to a generalised, distributed approach.

2.5.2.2 RosettaNET

RosettaNet is a not for profit organisation that is attempting to standardise supply chain transactions over the Internet. It is an XML based, standards initiative that is focussing on labelling of inventory, description and price data to allow easy communication between companies. It has contributing members such IBM, Microsoft and American Express.

2.6 Summary

This review has explored the fundamental concepts of supply chains, peer-to-peer information sharing, e-commerce systems, security in peer-to-peer networks, and modern methods of data interchange. The broad scope of this proposed research topic has been reflected in the broad nature of this review, where many different fields of study have been examined from several different disciplines.

Many papers now exist in the literature that are heading towards utilising some or all of these ideas for commerce applications. The usage of software agents for electronic commerce has been detailed in He, Jennings and Leung (2003), and also in Ibrahim, Schwinger and Weippl et al (2001), and again in Bartelt and Lamersdorf (2000). Extending this idea for the use of agents in B2B commerce specifically has been discussed by Blake (2001 & 2002). Agent based supply chain discussion exists in Eschenbächer, Knirsch and Timm (2000) and also in a formal, model-based approach in UML by Huget (2002).

...

3.0 Methodology

'There are two ways of constructing a software design.
One way is to make it so simple that there are obviously no deficiencies.
And the other way is to make it so complicated that there are no obvious deficiencies.'
-- Professor Sir Tony Hoare

This chapter presents the design and implementation of a XML based peer-to-peer system called SimpleP2P and a supply chain implementation called SupplyChainDemo. The first part of this chapter describes SimpleP2P and how it operates in building a peer-to-peer network. The second part of this chapter describes the design and implementation of a supply chain network application, built using SimpleP2P, that is used to prove the feasibility of a supply chain solution built using a peer-to-peer network.

3.1 System Overview

The system consists of a peer-to-peer environment where users can perform searches, make data available to other uses of the network and return search results to other nodes.

3.1.1 Application Development

A set of classes have been developed that facilitate the creation of a peer-to-peer network. During development emphasis has been placed on small size, clarity and portability. This has resulted in only 2 major classes classes (`SimpleP2P` and `SimpleNode`) that need to be understood to modify the system for other uses. One of these classes, `SimpleP2P`, has only three public member functions in its API that are enough to start a peer-to-peer network.

Each of these classes will be examined in detail as will the protocol that enables SimpleP2P to function. The full c++ source code for these classes can be found in the appendices.

3.1.2 Network Architecture

The SimpleP2P network is a simple implementation of the decentralised peer-to-peer paradigm. The network consists of a collection of nodes that are connected to one another over TCP/IP. Each node routes traffic between the nodes that it is directly connected to. This traffic consists of XML messages representing searches, network integrity messages and data requests. Each node accepts a finite amount of incoming nodes and attempts to connect to a finite number of outgoing nodes.

A node joins the network by attempting to connect to another known node. This known node serves as a *bootstrap* for the connecting node. It allows the connecting node to join the network via the bootstrap (if resources permit) and to discover more network nodes via a *try* list regardless of actual connection success or failure with that particular node.

SimpleP2P's connection and network maintenance model is based upon the core functionality of the Gnutella network model (see chapter 2.2.1.2) with the following major differences:

SimpleP2P	***GNUtella***
Network messages are swapped in plain text using XML.	Network messages are swapped in single byte aligned c structures.
Query matches result in a direct TCP connection from the matched node to the search owner.	Query matches are back propagated through the network towards the search owner.
Data is passed via the same node to node connection as used for messages. The requested data is just another message.	Files are transferred via a separate HTTP connection between nodes.
Network supports general data communication.	Network supports file trading only.

Figure 3.1: Major protocol differences between SimpleP2P and GNUtella.

3.1.3 SimpleP2P Network Communication Protocol

There are several messages that are passed between nodes in the SimpleP2P network. These messages are *ping*, *pong*, *query*, *connect-reply*, *connect*, *accept* and *deny*. All of these messages follow a simple XML schema in the form of:

```
<simplep2p>
      ...
      <payload>{ping, pong, query, ...}</payload>
      <ip>source ip address to connect back to</ip>
      <port>source port to connect back to</port>
      ...
</simplep2p>
```

Figure 3.2: Default SimpleP2P Message Schema

The start and end tags, `<simplep2p>` and `</simplep2p>` respectively, dictate the beginning and end of a single message and is known as the *root* element. The root element contains all of the other elements within the XML document message. In addition to employing the correct root element, each message must be well-formed XML meaning that each start tag within an element must have a matching end tag and that tags should correctly nest within one another.

There is no reason why the root element could not be changed for other applications of the SimpleP2P network. In fact, the root node itself should be seen as being synonymous with the network purpose itself and provides a quick way of separating different applications using this protocol. The base implementation of SimpleP2P provides services for maintaining the network and sharing small amounts of data. Any deviation from this functionality or purpose should be accompanied by a root element change. This will ensure that networks with different purposes remain separate.

It is worth noting that recent versions of the XML specification describe a "XML declaration" which declares the text to be an XML document and gives the version number of the XML e.g. `<?xml version="1.0"?>.` SimpleP2P omits the XML declaration as it is assumed that the initial XML swapped between nodes i.e. the *connect* and *accept* messages will be in XML version 1.0. Any further specifications or negotiations for message format can be negotiated in elements within the *connect* and *accept* messages. This will allow for future expansion without the cost of 21 bytes being added to the beginning of every single message that the system passes over the network. Harold (2001, p58) states that omitting the XML declaration from an XML document is acceptable providing there is a good reason to do so.

The *payload* field describes the general content of the message. This allows messages to be quickly interpreted by the node and sent to the appropriate functions for processing. The elements within the root element do not need to be in any specific order. The *ip* element contains the Internet address of the node who sent the message and the *port* element contains the port that the node is using for SimpleP2P communication. These two elements provide enough information for a node to connect back to the originating node.

3.1.3.1 Connect

This is the first message that a new node will send another node it is trying to connect to as soon as a network connection is established. It contains the connecting nodes connection details and the name and version of the node.

```
...
<payload>connect</payload>
<version>0.1</version>
<agent>Bare Simple peer-to-peer</agent>
...
```

Figure 3.3: Default SimpleP2P Connect Schema

When a node receives a connect message request, it determines whether it wants to allow this node to connect. This decision is based on available resources, the version of the connecting node and what authentication credentials the node possesses (described later in 3.2). The result of this decision should be conveyed by either an *accept* or *deny* reply.

3.1.3.2 Accept

An accept message advises a connecting node that their connection attempt has been successful.

```
<simplep2p>
      <payload>accept</payload>
      <try>{ip1:port1, ip2:port2, ..., ipn:portn}</try>
</simplep2p>
```

Figure 3.4: Default SimpleP2P Accept Schema

The accept message introduces an important element that enables a node to discover new peers. The *try* element contains a list of the connection details for other peers that the node knows about. These details may have be discovered during the connection process of the node or knowledge preset by the network i.e. bootstrap or friend nodes (As described in 3.1). The connecting node stores the values from the try element into a *bootstrap vector*. This bootstrap vector is used to attempt connections with other nodes on the network.

3.1.3.3 Deny

The deny message is sent to a connecting node to indicate that its connection attempt has been unsuccessful. The deny message is structurally identical to the accept message. In the default implementation of SimpleP2P, no reason is provided for the rejection. A try list is provided to enable the connecting node to further increase its chances of joining the network.

3.1.3.4 Ping

The ping message provides the synchronisation and integrity services for the network. A connected `SimpleNode` object, running on a thread, will expect a ping message every so often (as specified by the individual network application) to let its connected peer know that it is still alive and operating as a useful part of the network. Failure to receive a ping after a specified timeout will cause the `SimpleNode` object to assume that its connected peer is dead and that it should release its connection resources and attempt to connect to another peer on the network.

3.1.3.5 Pong

The pong message is the reply to a ping message. Where a ping message asks the node "*are you alive?*", the pong message serves as a reply as "*yes I am alive*". When a node receives a ping it should reply with a pong to maintain the connection.

3.1.3.6 Query

The query message allows for a search string to be propagated across the network. Each node that receives the message can compare its string payload with the data it maintains. Each node passes the query message to its other connected notes with the exception of the connected node that passed the message to it. This broadcast propagation has the potential to flood the network. To address this each query message has a *"time to live"*field. This field begins with a positive number and is decremented each time a node broadcasts it to its connected nodes. Once the TTL reaches zero, the node drops the message. This ensures that the message has a finite time being passed around the network.

Should the strings match, the node can use the address information contained in the query message to connect back to the originating node and let them know the matching nodes connection details with a *connect-reply* message.

3.1.3.7 Connect-Reply

The response to a Query message, the connect-reply message is a modified form of the connect message which a node connects directly to another to report a data match. This connection is only a short term connection which reports the matching node's connection details and the matching data that it contains. Once this information is provided, the connecting node closes the connection.

3.1.3.8 Signed

A signed message may be any message other message wrapped in a signed XML envelope. In the SimpleP2P implementation, only crl-update is supported for signed messages, however there is no reason that this could be extended to all messages.

A digital signature of the another SimpleP2P XML message is calculated and the original message is then placed within signed-data tags. The digital signature produced is placed in a signature element. This allows the receiver of the message to authenticate the contents of the signed-data element against the signature contained in the signature element. Based on this authentication, the receiver can trust the message.

An example of a signed message is given below with the signed message itself given in bold below:

```
        <simplep2p>
                <payload>signed</payload>
                <signed-data>
                        <simplep2p>
                        <payload>crl-update</payload>
                        <issuer>d0b2e4e5766a39fedc071baa035603a2e28d429e</issuer>
                        <serial>356a192b7913b04c54574d18c28d46e6395428ab</serial>
                        </simplep2p>
                </signed-data>
<signature>067974EA44EC7B4EAF54307EC5712BE23FD0AB08530F0F482B73C0E9405F57DC2F9782BFD4
D6421CFF7DC79F5424A841055F092116224D9130979526B6F89D7ABCED45410EA49AA0279C49C40D28ADC
DC7ABC6A7A7AB85CEB44EECB93AB2F7C79373A85EEA3382C3F45716DC0F30AD61E8467164F304FF4402FE
C60FB30D4B3E</signature>
        </simplep2p>
```

3.1.3.9 Crl-Update

The crl-update is a signed message that identifies an otherwise valid certificate as being revoked by the network. Each node on the network has a public key that it trusts as an authority for revoking certificates. This would usually be the public key of the authority that issued all of the certificates on the network.

The revocation message contains hashes of both the issuer of the certificate to be revoked and the serial number of this certificate. Providing this message is authenticated, each node that receives it stores it in a database that it later referrers to when checking incoming certificates from attempted SSL negotiations. Checks are performed by extracting the matching data from the incoming certificates (i.e. issuer, serial number), calculating hashes and then searching for these issuer and serial number hashes in the database. The node can then refuse connections based on their revocation status. Extra x509 certificate data such as a hash of the key itself, or indeed any other vital statistic, can easily be implemented as required by an application.

3.1.3.10 Crl-Request

The Crl-Request message is propagated to adjoining nodes and asks for these nodes to reply with the crl-updates they possess from a given time. The time given is usually the UNIX timestamp of the last received crl-update message. This allows a node to get a peer assured list of crl-updates that may have occurred whilst the node was offline.

3.2 Authentication and Security

Any network protocol designed to carry company data must include security features to protect the confidentiality of data transferred and to ensure the identity of other nodes on the network can be authenticated. SimpleP2P addresses confidentiality, authentication and revocation through using secure sockets and public key cryptography.

3.2.1 Confidentiality

Confidentiality is achieved through using the OpenSSL toolkit. OpenSSL implements the Secure Sockets Layer and Transport Layer Security protocols (OpenSSL 2004). Each node on the network must have its own signed public certificate and private key. The node will only communicate through a successfully established SSL connection.

The default implementation of OpenSSL uses an implementation of DES-CBC3-MD5 for transport security. See Applied Cryptography (Schneier 1994) and the IPSec RFC's (IETF 2003) for a description of triple DES in cipher block chaining mode and MD5.

3.2.2 Authentication

Authentication is managed in two different ways in SimpleP2P for two different reasons. Authentication of nodes joining the network is managed by SSL, where each node needs to have a certificate signed by the same signing authority as explained in 3.2.1. Authentication of data received is managed by digital certificates. The crypto++ library was chosen as a digital signature provider due to its open nature, wide acceptance and easy integration with c++.

A flexible signing method was designed which allows any piece of XML to be signed with a digital signature. This allows a receiving node to authenticate a piece of XML received by authenticating the digital signature accompanying it.

3.2.3 Revocation

Certificate revocation exists where a still valid certificate needs to be rendered invalid. In the case of SimpleP2P, a flexible certificate revocation system has been designed where multiple reasons for certificate revocation can be supported. In a system where reputation is important, reasons for the revocation of certificates must be made extremely clear. In addition, separating the certificates used for signing information and transport security, allows the flexibility of having multiple certificates for different types of transactions or different sub networks with a larger communication network.

The certificate revocation message itself is a signed XML message, as per authentication in 3.2.2, containing a list of hashed public certificate identifiers that represent the certificates that have been revoked. Each node on the network must trust the certificate that signs the certificate revocation. It is up to the particular implementation as to what extra data can accompany the revoked certificate hash. This extra data can be included in additional XML elements within the signed message. Each node on the network should keep a private database of revoked certificates so that they can be tested against certificates of new nodes attempting to connect.

Nodes receive a certificate revocation update by one of three means. When a node bootstraps off a trusted node, it should provide a certificate update for the connecting node containing any revocations since last time the node was on the network. Another way to receive a revocation update is whilst a node is connected to a network a revocation update may be received as part of a revocation broadcast from another node on the network. And the third is where a node may wish to request a certificate revocation list. This is performed as a small search where each node who wishes to participate returns a certificate revocation update for the requesting node.

These three methods have been designed as a trade-off between required bandwidth and decentralisation. A discussion of the strengths and weaknesses of this approach is given as results in 4.2.4.

3.3 Core Components

There are two main classes in the SimpleP2P system. The `SimpleP2P` class manages the overall connection for the node on the peer-to-peer network. It provides an interface to try to connect to other nodes on the network and listen for incoming connections. It manages the associated threading and data communication.

The `SimpleNode` class manages the data for each individual connection between SimpleP2P nodes. It is what forms the basis of the network itself. A `SimpleP2P` object will contain several `SimpleNode` objects each running on its own thread. The threading library chosen for implementation was the POSIX thread library due to its wide availability on many platforms.

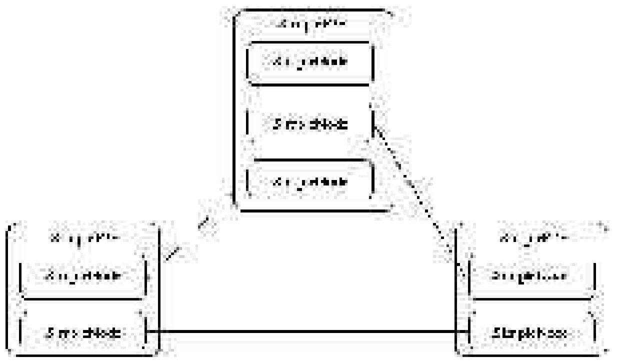


Figure 3.5: Showing the relationship between `SimpleP2P` *&* `SimpleNode`

Each `SimpleP2P` object acts as both a server and a client. At start up some of the `SimpleNode` objects are instructed to try to connect to others on the network whilst some `SimpleNode` objects are instructed to listen for incoming connections.

3.3.1 SimpleP2P

This section will describe the `SimpleP2P` class in detail. SimpleP2P provides an API to add bootstrap nodes for initialisation, create listening nodes, create connect nodes, set local data and start searches.

```
SimpleP2P(ip, port)
void push_bootstrap_node(ip, port)
void start_listen_threads()
void start_connect_threads()
void start_search(std::string search, Address source);
void set_local_data(std::string data);
```

Figure 3.6: The core public members of the `SimpleP2P` *class.*

Also, in the implementation of the simpleP2P class, there are several containers used for communication between threaded SimpleNode objects and the controlling SimpleP2P class.

```
std::vector<Address> connected_nodes;
std::vector<Address> bootstrap_nodes;
std::vector<Message> message_container;
std::vector<Message> reply_container;

struct Address {
      ...
      std::string ip_addr;
      unsigned short port;
      ...
};

struct Message {
      ...
      std::string message;
      std::string uuid;
      Address source;
      ...
};
```

Figure 3.3: The data containers for interprocess communication and the Address and Message classes.

We will now look at the functions in the `SimpleP2P` class in detail and how they are to be used in an application.

3.3.1.1 constructor

The `SimpleP2P` constructor takes the Internet address and port that the object will use as a point to accept connections. The selection of the Internet address is up to the user. The port used, for an unregistered application, should be one of the dynamic and/or reserved ports as specified by the Internet Assigned Number Authority (IANA 2004) so as not to upset other services running on the network.

3.3.1.2 push_bootstrap_node

As described in 3.1.3 and 3.1.3.2, each peer on the network discovers other peers through each node sharing the connection details of the nodes it knows about during the connection process. A new node must know of one or more *bootstrap nodes*. These bootstrap nodes should be nodes that have a history of reliability and availability on the network.

This function allows for known bootstrap nodes to be pushed onto the `bootstrap_nodes` vector for use by the listen and connect `SimpleNode` objects.

3.3.1.3 start_listen_threads

This function starts a thread to call the listen function on a `SimpleNode` object. `SimpleNode` handles its own threading so start_listen_threads need only start on object and `SimpleNode` will create its own threads to handle connections as necessary.

This is examined in detail in 3.3.2 as a part of the `SimpleNode` description.

3.3.1.4 start_connect_threads

This function takes a variable to determine how many connect nodes, that is nodes that actively try to connect to other listening peers on the network, should be started and creates a thread for each one.

Each thread calls the `connect_thread` function which begins to work through the bootstrap vector trying to connect to new nodes elsewhere on the network as shown in the pseudo code in the figure below.

```
loop forever
      if (number of nodes in bootstrap vector > 0) then
            node_to_try = next node on bootstrap vector
            if (node_to_try not myself) then
                  if (not already connected to this node) then
                        create SimpleNode object
                        SimpleNode attempt connect to node
                  end if
            end if
      end if
```

Figure 3.8: connect_thread algorithm

3.3.1.5 start_search

Mentioned in 3.3.1 are the two vectors `message_container` and `reply_container`. These two vectors provide a means of communicating with the independently running `SimpleNode` objects. Application specific command messages and replies are posted onto these vectors.

In the default search implementation, search strings are posted on the `message_container` and addressed to each `SimpleNode` via its UUID. The node then creates a *query* message and begins to propagate them as per the description given in 3.2.3. Subsequent replies are posted back to the `reply_container` by the `SimpleNode` that gets a *connect-reply*. Its then up to the application as to what to do with the *connect-reply* data.

3.3.1.6 set_local_data

This function sets the data which the node shares on the network. It simply sets a `std::string` which holds the data which can then be accessed by `SimpleNode` objects for comparison with incoming queries.

3.3.2 SimpleNode

This section will describe the `SimpleNode` class in detail. `SimpleNode` provides lower level functions required to manage communication and synchronisation over the SimpleP2P network.

```
SimpleNode(const std::string ip_addr,
            const unsigned short port);

void connect(std::string ip_addr, unsigned short port);
void connect_with_reply(std::string ip_addr,
            unsigned short port);
void listen(std::string ip_addr, unsigned short port);
static void parse_connect_string(std::string& data);
static void process_client_responses(std::string data,
            ServerSocket& server_socket, Address source);
...
protected:

void process_messages(ClientSocket& client_socket);
void add_address_to_bootstrap_nodes(Address address_);
...
```

Figure 3.9: The core members of the `SimpleNode` class.

3.3.2.1 constructor

The constructor is passed the Internet address and port as given in the SimpleP2P constructor (see chapter 3.3.1.1).

The node also generates a unique universal identifier (UUID) to uniquely identify an `SimpleNode` instance on the network. This function uses the `uuid_generate` function that produces a unique 128 bit value using a high quality random number generator. This UUID is also used for identifying and addressing `SimpleNode` objects within the `message_container` and `reply_container` for inter-process communication.

3.3.2.2 connect

The connect function contains the main functionality for the “connect mode” of a `SimpleNode` object. This function is called once a socket connection has been made to another SimpleP2P node on the network. It marshals the functions required to determine if the connection is successful and then enters a loop to repeatedly call `process_messages` to manage network communication.

3.3.2.3 connect_with_reply

`connect_with_reply` makes a single connection to another node in the SimpleP2P network for the purpose of announcing that it has matched a previous query message.

3.3.2.4 listen

This function waits on a socket for incoming connections. When a socket connection occurs, this function will create a thread and pass it a pointer to the socket to manage the new connection. The thread runs the listen_node function which either accepts or denies the incoming connection and, on connect, loops in the `process_client_responses` function.

3.3.2.5 parse_connect_string

Called from the listen_node function, this code will determine what type of connect the client is attempting to establish e.g. *connect*, *connect-reply, etc*. It will parse the incoming connect message marshal functions which determine whether there connection should be accepted or not.

3.3.2.6 process_client_responses

This function is called from the `listen_node` function. This function processes messages sent from a node that is in a client relationship with this particular instance of `SimpleNode`. This function will process query messages, ping messages, etc. and coordinate and send an appropriate response to each incoming message.

3.3.2.7 process_messages

This function is called by the connect member to process messages that appear on the `messages_container` vector from the controlling `SimpleP2P` object. It iterates through the vector, examining each message, and calling the appropriate functions to process them should the message's UUID match the UUID of the `SimpleNode` instance checking the container. For example a search command message might be added to the `messages_container` vector that will cause the process_messages function to compose a query message and send it to its corresponding `SimpleNode` object on a connected peer.

3.3.2.8 add_address_to_bootstrap_nodes

Called by the `connect` function, this function adds the given address to the `bootstrap_nodes` container providing the address does not already exist in the bootstrap container.

3.4 Applications

Two applications have been developed using the framework described in sections 3.1, 3.2 & 3.3. The first application, SimpleP2PTest, is designed to test the effectiveness of the framework when compared to an existing peer-to-peer solution. In this instance, Gnutella was selected as the comparing candidate due to it's open, decentralised nature, wide usage and availability. The second application, SupplyChainDemo, implements a simple, fictitious supply chain data network to explore the suitability of peer-to-peer computing when tested against situations typical of classical supply chain problems from the literature.

3.4.1 SimpleP2PTest Application

SimpleP2PTest was implemented as a test bed for the SimpleP2P protocol. It provides a graphical user interface to easily explore the features of the framework.

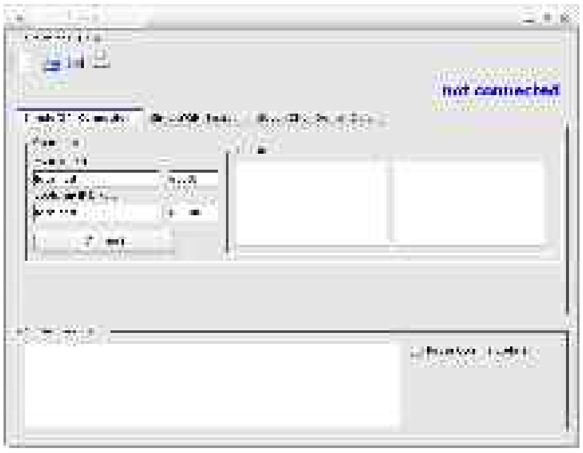


Figure 3.10: The SimpleP2PTest GUI

The user is able to enter a local IP address and port and stipulate the bootstrap IP address and port. Clicking the connect button starts a predefined number of connect and accept threads through the functions start_connect_threads and start_listen_threads. As threads start, they are assigned a universal unique identifier which shows in the statistics group to the right along with successfully established connections. All communication between nodes and debug logging is shown in the list box to the bottom of the tab page as shown in figure 3.11, 3.12 & 3.13.

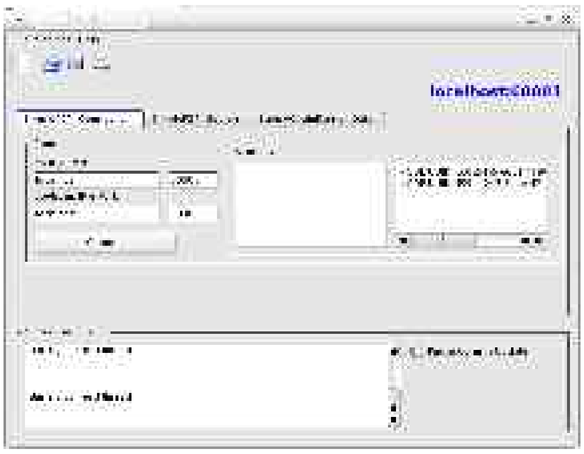


Figure 3.11: Connect & Listen Threads Starting

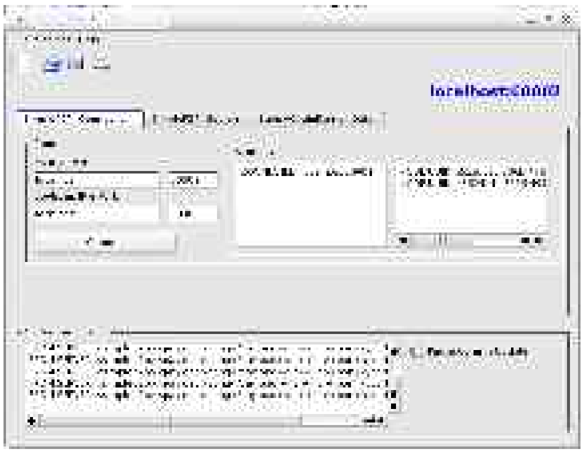


Figure 3.12: Successful connection and logging of data communication.

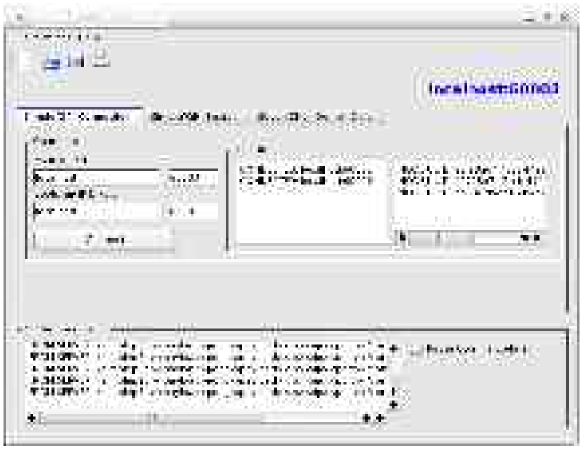


Figure 3.13: Further communication and logging as this new node boots off 60001, is told about 60002 and then connects.

3.4.2 SupplyChainDemo Application

To ascertain the effectiveness of peer-to-peer computing applied to supply chain management, a sample application was developed using the SimpleP2P framework and a limited simulation of usage was undertaken to explore the effectiveness of the proposed solution. The application of a fictitious corporate supply chain has been taken from the literature and it's communication channels have been limited to messages travelling over the SimpleP2P network. Discussion of fitness as results has been presented in Youll (2001) when determining the results of e-commerce systems and the same approach is to be taken here.

3.4.2.1 The Scenario

The classic example of the production of a shirt has been taken from the literature and the peer-to-peer solution has been applied to it (figure 3.14). One end of the chain is the selling or retail channel. Before this is the shirt manufacturer itself that takes materials from three other suppliers. One of these suppliers also has its own two suppliers of raw materials.

Each object in the supply chain was represented by a node in a peer-to-peer to peer network and experiments were conducted to explore the suitability of a peer-to-peer solution.

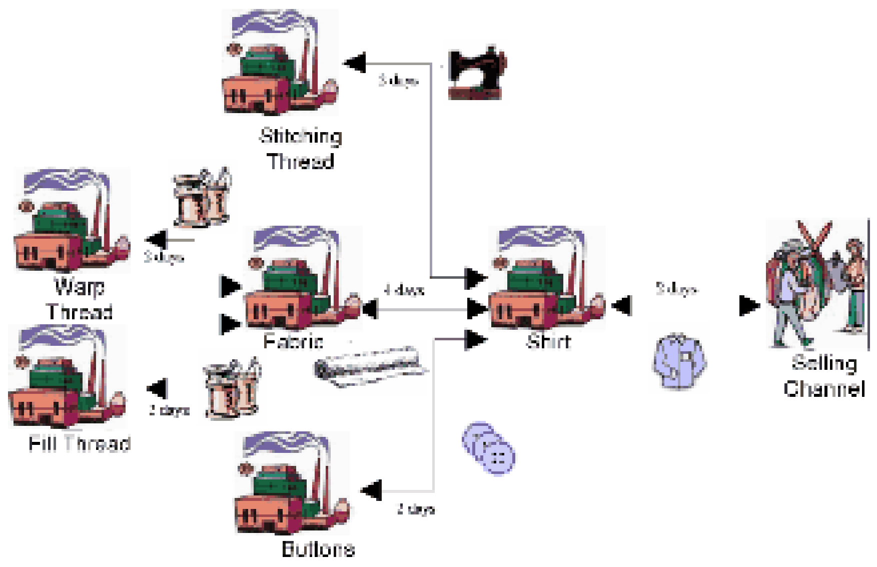


Figure 3.14: A shirt supply chain (Alodar 2004)

3.4.2.2 Implementation

A XML parser was added to SimpleP2P to allow nodes to share XML data. Shared data could be local data only, that is, data that has been added to the network wholly by that node. Or, shared data could be the result of local node data combined with data gathered from other nodes. Data is merged then transformed using XML style sheets (XSL) producing an XML output.

Consider the simple example given below. Two nodes on the network are each sharing a single piece of XML data with a single element in it. This data has been requested by a third node which combines the data and can then transform it into XML, HTML or whatever it likes e.g.

```
<data>
      <simplep2p>
            <data_item_1>data from node one</data_item_1>
      </simplep2p>
      <simplep2p>
            <data_item_2>data from node two</data_item_2>
      </simplep2p>
</data>
```

Data could then be transformed into presentation by adding a simple HTML transform thus:

```
<?xml version="1.0"?>
<xsl:transform version="1.0">
      <xsl:template match="local_data">
      <HTML>
            <xsl:apply-templates/>
      </HTML>
      </xsl:templates/>
      <xsl:template match="simplep2p">
      <P>
      <xsl:apply-templates/>
      </P>
      </xsl:template>
</xsl:stylesheet>
```

Resulting in the output:

```
<HTML>
<P>
data from node 1
</P>
<P>
data from node 2
</P>
</HTML>
```

This generalised XML sharing and transforming application was implemented to enable an implementation of the scenario described in 3.4.2.1. Each node will have its own data that it shares on the network. All of the XML schemas and XSL style sheets for each node in the simulated supply chain are available in the Appendices. Here we will consider the data that the button and shirt manufacturers publish on the network and the data of the retail channel.

Our fictional button manufacturer supplies a limited range of buttons suitable for shirts and trousers.

```
<button>
      <diameter>6</diameter>
      <material>polyester</material>
      <sew_holes>4</sew_holes>
      <colour>white</colour>
      <inventory>1234</inventory>
      <production>1000</production>
      <lead_days>2</lead_days>
</button>

<button>
      <diameter>8</diameter>
      <material>acrylic</material>
      <sew_holes>2</sew_holes>
      <colour>clear</colour>
      <inventory >321</inventory>
      <production>100</production>
      <lead_days>3</lead_days>
</button>
```

Our fictional shirt manufacturer also has a limited range. Short sleeve or long sleeve shirts are available in a selected range of colours.

```
<shirt>
      <id>12</id>
      <sleeve>short</sleeve>
      <front_buttons>7</front_buttons>
      <fabric>poly cotton</fabric>
      <colour>white</colour>
      <inventory>987</inventory>
</shirt>
<shirt>
      <id>26</id>
      <sleeve>long</sleeve>
      <front_buttons>7</front_buttons>
      <cuff_buttons>4</cuff_buttons>
      <fabric>poly cotton</fabric>
      <colour>blue</colour>
      <inventory>12</inventory>
</shirt>
```

Our retail channel or wholesaler simply has inventory and pending orders for each shirt.

```
<orders_by_product>
      <26>
            <inventory>100</inventory>
            <pending>1000</pending>
      </26>
      <12>
            <inventory>1000</inventory>
            <pending>250</pending>
      </12>
</orders_by_product>
```

Depending of the direction of the supply chain one is looking, it is now possible for the button manufacturer to gain an understanding of upcoming demand through pending sales from the retail channel and the retail channel is able to understand lead time through the shirt manufacturer. The shirt manufacturer knows how long it can expect for buttons to arrive and how many it is going to need through existing orders. Privacy and relevance of information is controlled through XSL transforms.

3.5 Summary

This chapter has explored the peer-to-peer system framework that was developed, and also two example applications that were built to demonstrate the use of this system.

In chapter four, we will presents the performance results gathered from the first application, SimpleP2PTest, and a discussion of suitability of a supply chain application of peer-to-peer computing.

...

4.0 Results & Discussion

Consistency is the last refuge of the unimaginative.
-- Oscar Wilde

This chapter presents the performance results obtained from testing the two applications described in chapter three. These results are used to justify whether supply chains can be effectively implemented as a peer-to-peer network.

4.1 Testing Environment

All testing requiring multiple machines was carried out using four 800Mhz Pentium III machines with 256Mb of RAM. Single machine testing was carried out at using a 1.7Ghz Celeron machine with 256Mb of RAM.

In all cases, the operating system used was Mandrake Linux 10 community edition.

4.2 SimpleP2P Framework - SimpleP2PTest

4.2.1 Performance Comparison with Gnutella

4.2.1.1 Search Bandwidth & Performance

The large bandwidth requirements of Gnutella are a known and much explored issue of the protocol. SimpleP2P, being modelled on the Gnutella, suffers from the same geometric increase in data generated during a search. SimpleP2P potentially consumes more bandwidth than Gnutella due to its larger overall packet size due to XML being used rather than binary structures as its communication language.

To address this problem, compression was tested to see under what conditions the bandwidth requirements of SimpleP2P can be improved to approach or exceed Gnutella's requirements.

From Ritter (2001), the bandwidth generated by a search request on a balanced Gnutella network can be given by the following formulae:

A function describing the maximum number of reachable users that are at least ***x*** hops away, but no more than ***y*** hops away.

f(n, x, y) = Sum[((n-1)^(t-1))*n, t = x->y]

A function describing the maximum amount of bandwidth *generated* by relaying a transmission of ***s*** bytes given any ***n*** and ***t***. *Generation* is defined as the formulation and outbound delivery of data.

h(n, t, s) = n*s + f(n, 1, t-1)*(n-1)*s

To quantify the sizes of search strings that are typical on Gnutella, search strings were taken from the Gnutella network over 3 20 minute sessions at 3 random times over 3 days. These strings were then sorted from lowest to highest and were wrapped in Gnutella and SimpleP2P packet headers.

Applying Ritter's formulae shows that the smallest string observed (5 bytes) on the network can potentially generate 388.5Mb across SimpleP2P and 205.2Mb across Gnutella. This is simply due to the fact that even with compression of XML, SimpleP2P adds more overhead to the string than Gnutella does.

Bandwidth Generated in Bytes (Search Packet: 53 bytes)

	T=1	T=2	T=3	T=4	T=5	T=6	T=7
N = 2	106	159	212	265	318	371	424
N = 3	212	477	848	1325	1908	2597	3392
N = 4	318	1113	2756	5565	9858	15953	24168
N = 5	424	2385	8480	22525	49608	96089	169600
N = 6	530	4929	25652	90365	248358	576905	1187624
N = 7	636	10017	77168	361725	1242108	3461801	8313792
N = 8	742	20193	231716	1447165	6210858	20771177	58196968
N = 9	848	40545	695360	5788925	31054608	124627433	407379200

N = Number of connections per node T = Time to live (TTL) of packets

Figure 4.1:SimpleP2P bandwidth generated for a 5 byte search string.

Bandwidth Generated in Bytes (Search Packet: 28 bytes)

	T=1	T=2	T=3	T=4	T=5	T=6	T=7
N = 2	56	84	112	140	168	196	224
N = 3	112	252	448	700	1008	1372	1792
N = 4	168	588	1456	2940	5208	8428	12768
N = 5	224	1260	4480	11900	26208	50764	89600
N = 6	280	2604	13552	47740	131208	304780	627424
N = 7	336	5292	40768	191100	656208	1828876	4392192
N = 8	392	10668	122416	764540	3281208	10973452	30745568
N = 9	448	21420	367360	3058300	16406208	65840908	215219200

Figure 4.2 :Gnutella bandwidth generated for a 5 byte search string.

However, as the string gets longer, the compression of the overall packet tends towards Gnutella sized packets. With the longest string observed (111 bytes), SimpleP2P can potentially generate 916.3Mb whilst Gnutella would generate 982.3Mb.

Bandwidth Generated in Bytes (Search Packet: 134 bytes)

	T=1	T=2	T=3	T=4	T=5	T=6	T=7
N = 2	268	402	536	670	804	938	1072
N = 3	536	1206	2144	3350	4824	6566	8576
N = 4	804	2814	6968	14070	24924	40334	61104
N = 5	1072	6030	21440	56950	125424	242942	428800
N = 6	1340	12462	64856	228470	627924	1458590	3002672
N = 7	1608	25326	195104	914550	3140424	8752478	21019776
N = 8	1876	51054	585848	3658870	15702924	52515806	147139504
N = 9	2144	102510	1758080	14636150	78515424	315095774	1029977600

Figure 4.3:Gnutella bandwidth generated for a 111 byte search string.

Bandwidth Generated in Bytes (Search Packet: 125 bytes)

	T=1	T=2	T=3	T=4	T=5	T=6	T=7
N = 2	250	375	500	625	750	875	1000
N = 3	500	1125	2000	3125	4500	6125	8000
N = 4	750	2625	6500	13125	23250	37625	57000
N = 5	1000	5625	20000	53125	117000	226625	400000
N = 6	1250	11625	60500	213125	585750	1360625	2801000
N = 7	1500	23625	182000	853125	2929500	8164625	19608000
N = 8	1750	47625	546500	3413125	14648250	48988625	137257000
N = 9	2000	95625	1640000	13653125	73242000	293932625	960800000

Figure 4.4: SimpleP2P bandwidth generated for a 111 byte search string.

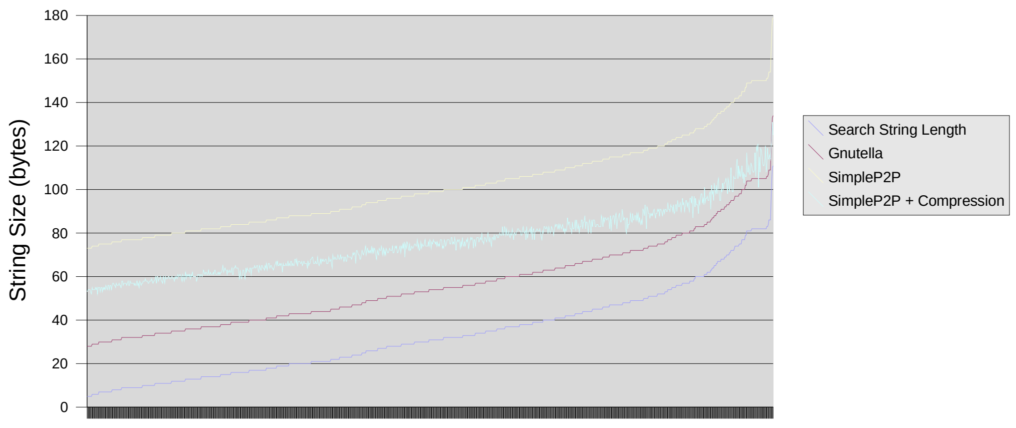


Figure 4.5: Average search strings from Gnutella shown with relative packet sizes. The original string length, the Gnutella packet length, the SimpleP2P packet length without compression and the SimpleP2P packet length with compression enabled.

As can bee seen in the figure 4.5, XML compression has lowered the overall packet size in SimpleP2P and trends toward Gnutella packet size as observed strings head towards 110 bytes in length. Another observation is that XML compression provides positive results over the full range of search string lengths observed.

Name	***Compression Algorithm***	***Compressed Size***
Gzip	Lempel-Ziv	105
bzip2	Burrows-Wheeler & Huffman coding	101
xmlppm	XML aware Prediction by Partial Matching	61

Figure 4.6: Compression size results for a 81 bytes of XML containing a 13 bytes search payload.

Other compression techniques were considered, however they did not approach the results given by xmlppm. Lempel-Ziv and Huffman coding produced outputs larger than their inputs across the whole spectrum of string sizes. One string measurement is included in Figure 4.6 as an example.

The bandwidth generated on a reply differs greatly between Gnutella and SimpleP2P. Gnutella passes reply messages are back propagated through the network incurring a large amount of generated traffic as per a query itself (See Ritter 2001 for an in-depth analysis of this). Gnutella does this to minimise the effect of firewalls. If you can connect your client and make a query, you will be able to receive a query on that same socket connection. This allows firewalled clients to receive a query hit without needing to reply on an outside node making a direct connection. SimpleP2P simple makes a direct connection and sends a *connect-reply* message back. This was considered suitable for SimpleP2P given that is has been designed with corporate use in mind where port allowances will be made. This is something that would need to be considered if SimpleP2P is to be used for real world applications where firewalls may pose a problem.

4.2.1.2 Bootstrapping

Bootstrapping on SimpleP2P behaves similarly to Gnutella. The time taken to find a open space to join a network depends on the overall size of the network, how it is organised, how many connections are available on each node in the network, how many threads are searching on the joining node, etc. There are too many variables that can change almost randomly to accurately reflect time taken in a formula. A recent study by Karbhari et al (2003), measured the different bootstrap times for four different Gnutella implementations. This was done by modifying several available clients (i.e. GTK-Gnutella, Mutella, etc.) to include statistical information and having them connect to the Gnutella network. GTK-Gnutella, the most similar to SimpleP2P in terms of bootstrapping, had a range of approximately 20 to 190 seconds depending on the time of day a connection was attempted. Lacking a large community of users, SimpleP2P was tested using the small network of machines described in 4.1.

To test the bootstrap time of SimpleP2P, a worst case search tree (a list) was constructed by bootstrapping all of the nodes of the same node with only one thread for new connections. The time taken for each new node to join was taken.

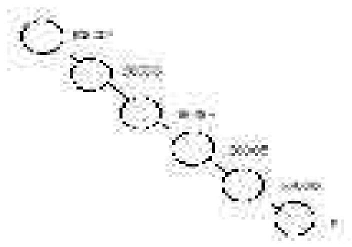
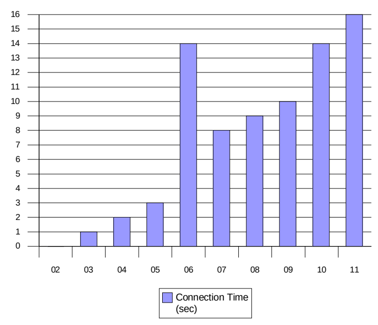
Node	Connection (first ping reply)	
60002	0	  0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 02 03 04 05 06 07 08 09 10 11 Connection Time (sec)
60003	1	
60004	2	
60005	3	
60006	14	
60007	8	
60008	9	
60009	10	
60010	14	
60011	16	

Figure 4.7: Bootstrap times in list.

The results show that the connection time is roughly linear. The discrepancy in some results, such as node 60006, can be explained by the connection thread and the listening thread being synced in such a way that the sleep time of one second caused an extra delay in communication between the two nodes. Also the possibility of operating system influence is not unreasonable.

A second experiment shows connection in a binary search tree. 7 nodes were arranged in a binary search tree, with a open connection available on the lower, rightmost node.

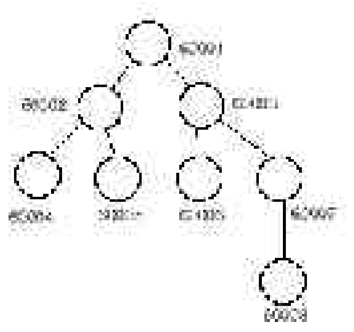
Node	Connection (first ping reply)	
60002	0	
60003	0	
60004	0	
60005	1	
60006	1	
60007	1	
60008	1	

Figure 4.8: Bootstrap times in binary network.

With 3 threads operating, bootstrap until connection time was 1 second as shown in the table above.

4.2.2 Application Implementation with SimpleP2P compared to other frameworks

SimpleP2P grew out of realisation that there were no available peer-to-peer network frameworks suitable for an investigation such as this one. As a result of this, a comparison can be drawn between SimpleP2P and other available frameworks for how much code is required to produce a similar result of a decentralised peer-to-peer network.

A brief comparative study of the sizes of various frameworks that compare to SimpleP2P, in terms of core functionality, were quantified using the SLOCCOUNT tool (WHEELER 2001).

Framework	Lines of Code
JXTA	93366
SimpleP2P	3646
Gtk-Gnutella	79470

Figure 4.9: Comparative size of code bases.

This was investigated simply to compare how well SimpleP2P fares against other incumbent networks so far as code usability is concerned. SimpleP2P can be used by calling as few as three functions from a c++ class (as discussed in 3.3.1) providing a general peer-to-peer network where specific needs can be added without removing existing code. This consideration alone is why another existing network like Gnutella was not used as a starting point for this supply chain investigation. The average Gnutella client contains so much extraneous functionality (i.e. user chat, GUI skins and other customisations, etc.) that stripping it bare so that the underlying protocol can be used for an academic investigation is more of a task than rewriting a client.

4.2.3 Criteria for Evaluating Suitability of peer-to-peer frameworks

As a part of selecting a peer to peer paradigm to apply to supply chains, an evaluation of publicly available peer-to-peer software and frameworks was undertaken. The results of this evaluation can now be presented as a criteria to identify a suitable peer-to-peer paradigm to use based on the requirements of the application. Seven criteria were identified as most descriptive in terms of the features available in peer-to-peer systems at time of writing. Criteria for evaluating peer-to-peer networks has also appeared in Milojicic et al (2002) & Minar (2001).

Generality	Mentioned in 4.2.2, generality of a peer-to-peer network is an important consideration. Some networks such as JXTA, and now SimpleP2P, are generalised peer-to-peer where no specific application, i.e. file sharing, collaboration, etc., has been built into the network itself. This contrasts to more popular peer-to-peer networks in use such as Gnutella where the network has evolved around file sharing. The focus of a network needs to be considered if adaptation to another purpose is to be considered.
Security	The first consideration for security is to whether or not the protocol contains any security considerations at all. This can range from no security whatsoever (Gnutella) to security being the primary consideration (Freenet).

Topology	Network topologies range from centralised, ring, hierarchical, decentralised or hybrid which can be a combination of several.
Scalability	Network scalability and bandwidth usage are of primary importance when selecting any network protocol. Can the scalability of the network be proven mathematically or predicted accurately?
Extensibility & Generality	Is the network general an easy to extend? Or will it need first to be stripped down and refocussed for a new purpose? What is the method of data exchange? Is it structured and difficult to change or fluid such as an XML implementation?
Reliability	Has the network been designed to be fault tolerant to nodes joining an leaving? Are searches guaranteed to reach an acceptable or expected number of nodes?
Legal Robustness	Has the network an appropriate development license for a particular requirement? Does the network contravene any known laws or is it already associated with illegal activity? Can the removal of any one firm or organisation cause the network to fail?

4.2.4 Security Appraisal

The discussion of the security aspects of SimpleP2P are to be discussed in terms of how well what has been built satisfies the security goals of the system. This is a similar approach taken in other papers, such as Schneier & Ferguson's (1999) evaluation of IPSec, where a system utilising cryptographic primitives are being put into practice. The same approach will be taken here as a formal mathematical evaluation is not relevant given that an actual protocol is not being presented. The goals of confidentiality, authentication, certificate revocation and non-repudiation will be discussed.

4.2.4.1 Confidentiality

Confidentiality was implemented and described in 3.2.1. Secure sockets are used where a security negotiation must be successfully completed before a encrypted data channel is established so a node is unable to exchange any data unless the SSL connection is correctly established. SSL is an accepted method of maintaining a reasonable level of confidentiality over the Internet. The implementation of SSL used in SimpleP2P uses DES-CBC3-MD5 which is widely considered to be strong encryption at time of writing.

It is possible for conspiring nodes to exchange data in clear text between one another whilst maintaining secure connections with other nodes at the same time. This would be a risk if sensitive information is being distributed over the network as a whole and these nodes provide a weakness. This is lessened by there not being a clear text option during negotiation.

4.2.4.2 Authentication

Authentication of a node at time of connection with another node is managed through them both needing to have their SSL certificate signed by a commonly accepted signing authority. No check is made at runtime with a third party as this would generate both an unacceptable amount of bandwidth to a single point as well as a single point of failure. This method is susceptible to conspiring nodes where a modified node could allow a non-authenticated node onto the network. This could be mitigated slightly by encryption of the actual XML, and was considered, however little stops the authenticated conspirator from decrypting the XML and passing it on as plain text. Revocation for node malfeasance was considered and implemented to allow nodes to be excluded from a network.

Authentication of individual XML messages was also considered and implemented. Each message can be signed so that nodes can be assured of its signer. The usage of digital signatures is accepted practice subject to legal acceptance of validity in the country of use (Schneier 1994 pp 454).

Both of these authentication solutions are susceptible to private key compromise hence a certificate revocation feature was implemented.

4.2.4.3 Certificate Revocation

Each mode of receiving a certificate revocation, as described in 3.2.3, will be discussed. Receiving a revocation update from a node at bootstrap depends entirely on the level of trust that the nodes share. The revocation list itself cannot be easily forged due to its digital signature but a bad node can withhold an update. Potential withholding of updates is also an issue with the second way of receiving an update which is through broadcast whilst connected. A bad node could drop certificate update packets maliciously, though the decentralised nature of the network will make this less effective as a node will likely receive the same update from another connection. Obviously a singly connected node will be particularly susceptible to this. The problems of deliberately dropped packets was considered and a third way to gain a revocation update is through making a revocation request. A revocation request also allows for booting from non-trusted nodes which lessens the points of failure of the network; ideal trusted nodes need not exist on the network for booting from.

The revocation request works as a small search and allows for many nodes to report on revocation. This alleviates the problem of misbehaving nodes dropping packets but introduces a bandwidth dimension to the security of the network. The revocation request must be propagated over the network which costs at a magnitude similar to a search as per section 4.2. This also introduces a denial of service possibility. A node would be able to make repeated revocation requests and consume network resources. This should be mitigated by a network specifying a low TTL for revocation requests (as a node should be satisfied with a consistent response measured in 10s of responses rather than 1000s or 10000s). Revocation requests, indeed any packet, with a higher than acceptable TTL can just be dropped. So a revocation denial of service attack should be able to remain localised however it is still a tangible weakness.

4.2.4.4 Non-Repudiation

Whilst not explicitly implemented in the version of the code accompanying this thesis, it is clear to see that the signing scheme used in 4.2.4.3 for certificate signing could be used with any data on the network. Where non-repudiation is required, each packet could be signed before transmission allowing the receiving node to be assured that the data is actually from the signing party.

For “stronger” non-repudiation, there is no reason why an intermediary node could not be used to store transaction details for both nodes. However, this is beyond the scope of implementation at present.

4.2.4.5 Security Summary

It has been shown that the major facets of network security have been addressed and discussed in terms of their impact on a peer-to-peer network. The largest susceptibility has been shown to be malicious nodes who can remove confidentiality and authentication by masquerading as a good node with some connections whilst conspiring with others.

Another problem, shown earlier in the search results, is the overall susceptibility of the network to denial of service attacks. Indeed, compared to some other topologies, a standard search on a decentralised network can be considered a denial of service attack! This has been identified by many researches in the field and research into new designs continues.

4.3 Supply Chain Application – SupplyChainDemo

Determination of results through the gathering of actual data is something that was not possible to do given the resource constraints of this thesis. Instead, a fictional supply chain was implemented (proposed in section 3.4.2) and the classical problems that such a supply chain would likely face, as described in the literature, are discussed in terms of the performance of the underlying peer-to-peer architecture (SimpleP2P).

4.3.1 Scenario Bandwidth

The amount of bandwidth expected across the peer-to-peer supply chain depends on the number of participating nodes, how much data they are publishing, how much data they are requesting and how often a request occurs. Appendix 7.2 shows the data from a mock implementation of the shirt supply chain. This data shows what may be published for a single product across the supply chain, in our case a white, long sleeve shirt.

Data requirements were calculated from each node in the network as a request to each other node in the network plus their data reply. This represents a bandwidth worst case of every node asking every other node for their data at once. These results are given in the table below.

Node	***Request***	***Reply***	***Data***	***Total***
Warp	66 * 6 requests	165 * 6 replies	725	2111
Fill	66 * 6 requests	165 * 6 replies	725	2111
Stitching	66 * 6 requests	165 * 6 replies	720	2106
Fabric	66 * 6 requests	165 * 6 replies	717	2103
Button	66 * 6 requests	165 * 6 replies	679	2065
Shirt	66 * 6 requests	165 * 6 replies	610	1996
Wholesaler	66 * 6 requests	165 * 6 replies	714	2100
Total Data Per Exchange (bytes)				14592

Figure 4.10: Total data exchanged over network in a request from every node to every other node.

This suggests that for one product in the scenario, 14.25kb of data is transferred over the network for a complete swap of information between each and every node. To put this into daily terms we also need to cater for the network management information to be transferred as well. A ping/pong pair is 114 compressed bytes on the network.

The amount of overhead the network management messages generate depends on how the network is organised. A best case scenario would be if one node is connected to only one other node placing the network in a list. However this is not ideal for searching or robustness. Following the scenario, the network has been arranged to match the physical relationship between the various companies. This will results in 6 ping/pong pairs being swapped every network update.

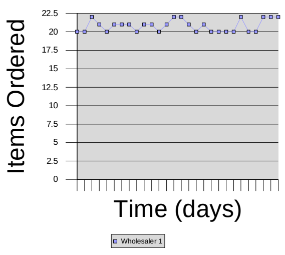
Based on the network exchanging keep-alive pings every minute, this would result in 6 * 114 * 1440 = 984960 bytes or approximately 962kb per day. Swapping all node data every hour would add 342kb per day resulting in 1304kb or slightly over 1.3Mb per day plus TCP/IP overhead. Scaling this result from one product, as shown, through to 10,100,1000,10000 products on this network results in projected daily bandwidth requirements of 3.340Mb, 33.398Mb, 333.984Mb and 3339.844Mb per day. Even the worst case of transferring 3.340Gb per day on the network results in approximately 139Mb per hour which is within reach of a modest broadband

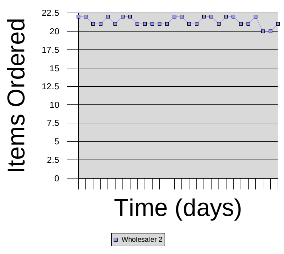
Internet connection at time of writing.

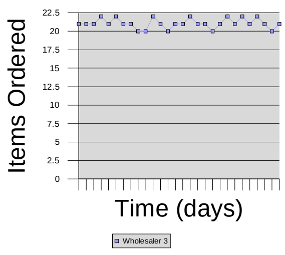
4.3.2 Minimisation of the "bull whip" effect

As discussed in 2.1.3.1, the bull whip effect is where variation in demand at one end of a supply chain can cause large effects on ordering and inventory at the other end. Amplification of this effect along the chain occurs for a variety of reasons. Poor communication with other stages in the supply chain is a major cause as is poor decision making at some points. Problems in delivery time (between order and stock arrival) can also contribute to the bull whip effect. Whilst a peer-to-peer supply chain solution can do nothing for bad decisions or logistics, it does allow for communication between all nodes in as effective a manner as each node wishes.

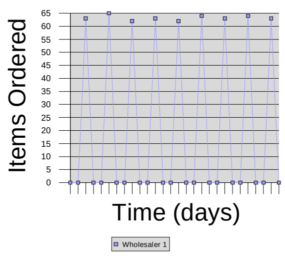
An experiment was designed to show the bull whip effect and test how well a peer-to-peer solution performed. 3 fictitious retailers, each selling 20±2 items each day, were generated with random values.

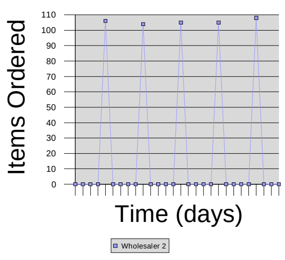


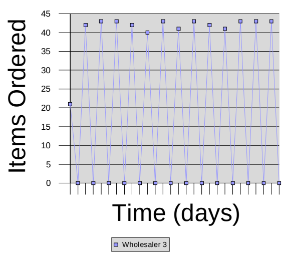




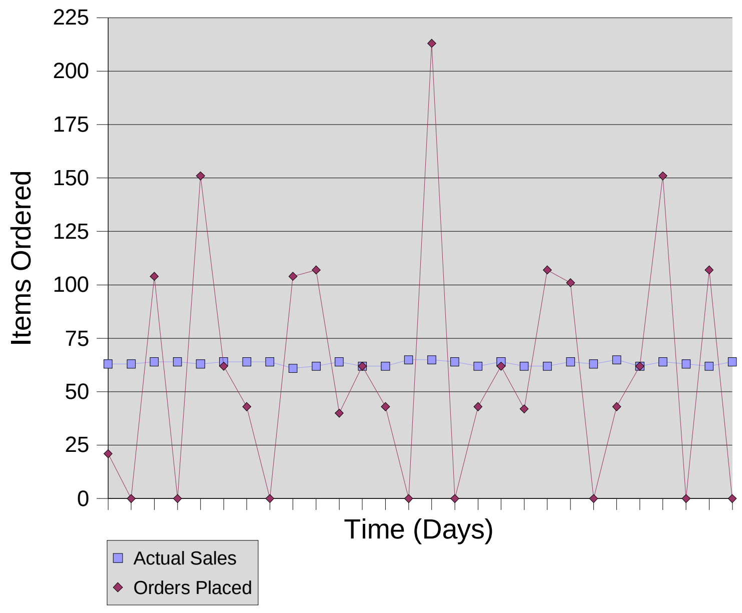
Each retailer orders from the wholesaler on different days. Each retailer batches their orders to economise on delivery costs and/or the costs of making the order itself in terms of resources. Retailer 1 makes their orders every 3 days, retailer 2 every 5 and retailer 3 every 2 days. Each retailer has little variation in the amount that they order and the difference between the period in which they order.







This simple scenario from the perspective of the manufacturer provides a different view.



An implementation of this supply chain using SimpleP2P allows the manufacturer to instantly see that demand is an average of 63.5 units per day. This compares with the 59.86 units per day that the manufacturer would predict on its own. The manufacturer also has no way of knowing the trend in actual sales. As we can see from the figure, sales are almost constant hovering an average with a low standard deviation (1.11). This is not possible to determine from the orders placed with the manufacturer where the standard deviation is (55.9).

4.3.2.1 Centralisation of demand information

Centralisation of demand information is a known goal of many supply chain implementations, especially where there are multiple retailers and dispatch warehouses (Chopra & Meindl 2004 pp 317). The peer-to-peer solution offers demand information at either end of the supply chain. Indeed it is possible at, or at any point along, the chain to gain information about any other node's situation.

A stated problem with supply chains is that sales spikes and dips can often be hidden from partners in a supply chain for up to weeks at a time. Hedging of information supply can cause sales data to be delayed up to the update time multiplied by the number of hops in the chain.

SupplyChainDemo allows for any node to request data from any other node.

In our scenario it is possible for a warp thread node at the supplier end of the chain to see an order immediately at the retail channel. In a real world implementation this responsiveness would depend on the level of integration between the peer-to-peer supply chain and the gathering point of data. It is not unreasonable to expect that timely access to point of sale terminal data would be available, this would allow manufacturers to gauge the sales of their products and forecast appropriately.

4.3.2.2 Sharing of customer information

Customer information gathered at the retail channel can be shared with the rest of the network. This data could then be used in identifying customer trends, possible opportunities, etc. for the benefit of the supply chain as a whole rather than just the node which collects the information. This may result in a more coordinated approach to customer servicing rather than a myopic view where common aims and goals are not accurately considered.

4.3.2.3 Sharing of organisation information

Similar to the sharing of customer information, sharing organisation information is beneficial to the supply chain as a whole. Differing frequencies of order runs between firms in a supply chain can cause problems. If a retail channel receives a large order, placing an order with its suppliers a week later, then this supplier places its orders to suppliers to fulfil the order some days after this and so on, it can be shown that significant amounts of time can pass before knowledge of the large retail order reaches all involved parties. This can add unnecessary stress to suppliers. As shown in our SupplyChainDemo, orders placed at the retail end of the chain are recognised by the opposite, manufacturing in near real time.

An anecdotal result is that the ease of communication may remove a current variable when looking at supply chain efficiency thus helping identify other non-communication related problems.

4.3.3 Location of other products and vendors

This solution offers new features that are not available in traditional supply chain software solutions. An open network of vendors allows for searches to be performed by another node to find products and services in a very efficient manner. Section 4.2.1.4 showed how a search in even a large number of nodes provided results in a timely manner. This compared with a more traditional approach to locating new stock i.e. word of mouth, advertising, trade shows, etc. could potentially show pleasing results.

4.3.3.1 Increased reliability through multiple vendors

As a result of the ability to locate new vendors quickly and communicate with them easily, it is envisaged that extra suitable vendors could be integrated into the supply chain as necessary.

Multiple vendors can be integrated in a manual or automatic fashion. Flow and decision control structures in XSL can allow for straightforward, automatic decision making at supply points e.g. an additional button supplier could be added to the supply chain with a decision being made at the shirt manufacturer node. If a suppliers lead time raises to an unacceptable level they can be automatically ignored in favour of the other supplier.

The results below show two suppliers for buttons. The extra supplier was added to the network at the same level as the existing button supplier. Both button suppliers provided the following data to the network:

```
<button>
	<company>ABC Buttons Inc</company>
	<diameter>8</diameter>
	<material>acrylic</material>
	<sew_holes>2</sew_holes>
	<colour>clear</colour>
	<inventory >1234</inventory>
	<production>100</production>
	<lead_days>3</lead_days>
</button>

<button>
	<company>XYZ Buttons Pty Ltd</company>
	<diameter>8</diameter>
	<material>acrylic</material>
	<sew_holes>2</sew_holes>
	<colour>clear</colour>
	<inventory >321</inventory>
	<production>100</production>
	<lead_days>3</lead_days>
</button>
```

Selection between the two suppliers, based on their inventory was found possible through using an XSL transform on the button data at the shirt manufacturer node. The XSL fragment below was included and it was found that automatic selection between the two suppliers was possible.

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
<xsl:template match="local_data">
```

```
<xsl:for-each select="button">
<xsl:choose>
<xsl:when test="inventory &gt; 1000">
<button>
<company>
<xsl:value-of select="company" />
</company>
<inventory>
<xsl:value-of select="inventory" />
</inventory>
</button>
</xsl:when>
</xsl:choose>
</xsl:for-each>
</xsl:template>
</xsl:stylesheet>
```

The supplier with an inventory greater than 1000 is chosen and the following XML is then used by the Shirt manufacturer for its own forecasting. The wholesaler or retail channel would also then be able to automatically see what the production capabilities of the shirt supplier is based on the materials that are available to it.

```
<?xml version="1.0" encoding="UTF-8"?>
<button>
<company>ABC Buttons Inc</company>
<inventory>1234</inventory>
</button>
```

Another results was found that, instead of selecting between them, aggregation was also possible. This would allow the shirt manufacturer to automatically take supply data from two nodes and use that data for its forecast data. The XSL below allows for an aggregation of the two button nodes inventory data.

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
<xsl:template match="local_data">
<button>
<inventory>
<xsl:value-of select="sum(descendant::inventory)" />
</inventory>
</button>
</xsl:template>
</xsl:stylesheet>
```

Producing the output thus:

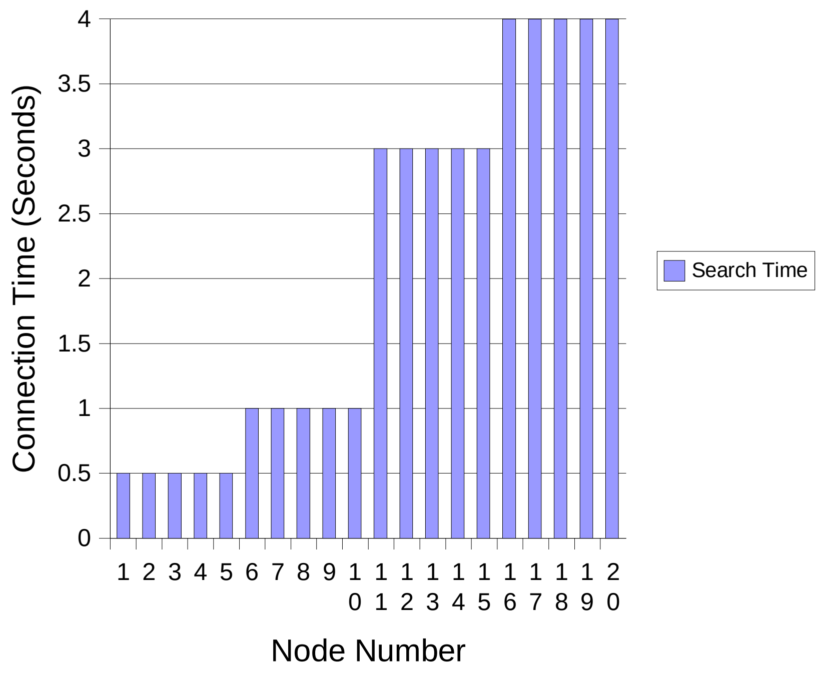
```
<?xml version="1.0" encoding="UTF-8"?>
<button>
<inventory>1555</inventory>
</button>
```

Whilst only trivial examples were shown, it is clear to see that the use of XML and XSL transforms allows for automation of selection or aggregation of suppliers. More elaborate business rules could be implemented leading to a more effective solution. An investigation of real automation of decision at supply nodes, using more complicated techniques, is discussed as further work in section 5.5.

4.3.3.3 Efficient comparison of suppliers

The ability to quickly search potentially thousands of nodes and quickly communicate with them would allow for suitable suppliers to be found based on products offered, price, lead time, etc. This would allow a straightforward method of improving a supply chain and cost performance through continual comparison with other nodes available to the greater network.

A search experiment was performed where 20 nodes (5 nodes by 4 machines) were created and each was given similar data. Target data was placed on the first node, and searches for that data were performed at the 5th, 10th, 15th and 20th nodes.



This compares favourably with manual methods of selecting a supplier where even a superficial search could be seen to consume far more time if relying on offline methods.

4.3.3.4 Third Party Authentication, Security and Non-repudiation

In a secure implementation of a SimpleP2P network, each node on the network must possess a correctly signed certificate in order to join the network (see chapter 3.2). As such, the certification authority that authorises the certificate signing should authenticate and appraise the candidate entity wishing to enter the network to a standard accepted as a minimum by participants in the network. This allows some reassurance to nodes that all other nodes have passed a minimum standard of suitability and identity. This implied reassurance, whilst generally a positive thing, should be treated with caution. A well implemented network with a thorough certification authority will make sure that a node is suitable for the network as a whole. A poorly run certification authority may just check the nodes contact details or nothing whatsoever. This must be considered when evaluating a node in any secure situation where there is 3rd party authentication. The addition of a reputation management system may go some way to alleviating this problem and is discussed as further work in 5.1.6.

In any case, even in light of the new problems that security measures introduce, the fact that they introduce security at all to the supply chain network is an important result.

4.3.4 Easy Integration and Adoption

The open nature of SimpleP2P/SupplyChainDemo allows for any supplier or buyer to join a supply chain network and operate within it quickly. The technical requirements are low. A modest computer, a free operating system (i.e. GNU/Linux) and an Internet connection able to cope with the bandwidth requirements as discussed in 4.2.1.1, is enough for the node to join the network. A working knowledge of XML/XSL would also be required to be operate effectively, however the example shirt supply chain given shows that this working knowledge required is not excessive. Indeed it is envisaged that in a real implementation the XML/XSL schemas required would be dictated by the owner of the network and a intuitive user interface would lessen the burden on the operator.

As shown by the flexible nature of XML, existing stock control systems or manufacturing systems could be added to the chain in a trivial manner. At the same time, manual systems can also be supported with data being hand entered. This, of course, comes with the caveats of accuracy and timeliness, but it provides an opportunity for one supply chain to capture data from a variety of methods; both manual and automatic.

4.3.5 Perceived Risks from a Business Perspective

The greatest risk is that the openness of the network is its greatest strength and its greatest weakness. The full benefits and positive results only exist when every node is open and participates fully. This may require a sea change in management methods yet this is not without precedent. Running companies in an open and honest method has been proven in the literature to work well and increase profits. And has been shown earlier in the discussion of the bull whip effect and in Chopra & Meindl (2004 pp. 483), that the lack of information coordination between nodes makes this effect much worse.

However, wide availability of data in a supply chain is always going to be commercially sensitive and therefore valuable to competitors. A malicious node could easily capture packets from a supply chain and sell the information to an interested party. This is not new and a manual supply chain is just a susceptible. However the electronic nature of a peer to peer supply chain would make malfeasance of this type much less effort. Even without sharing any inventory or production information, the open nature of searches in a peer-to-peer network (many nodes see search packets which include the search payload and who it was from) potentially allow nodes to infer the intentions of a firm from the searches that it is conducting. This could be lessened through the proxying of searches through other nodes, e.g. the addition of an obfuscating middleman, yet it is still an issue to be considered.

4.4 Summary

The results will be summarised in the same two part method as has been the theme of this chapter. SimpleP2P will be considered first and then SupplyChainDemo.

4.4.1 SimpleP2P

The first important result that has resulted from the implementation of SimpleP2P is the fact that it is possible to produce a decentralised, peer to peer network in a short period of time with relatively few lines of code. In addition, XML was shown to be able to compete with the binary structure approach, providing a suitable compression algorithm was used. Smaller string lengths fared worse with SimpleP2P but tended positively toward Gnutella's packet size as the string payload became larger. This becomes a classic tradeoff between flexibility and resource requirements. SimpleP2P has the flexibility due to XML, where new data can be added without affecting the existing implementation, at the cost of it requiring more bandwidth in some situations.

SimpleP2P also highlighted many of the problems common to decentralised peer-to-peer networks such as the difficulty in quantifying scalability due to lack of organisation, bandwidth requirements of searches and issues with joining the network itself.

4.4.2 SupplyChainDemo

It has been shown through an analysis of a simulated application of peer-to-peer computing to a supply chain, that an open, XML based, peer-to-peer solution would enhance the ability of each member organisation to have centralised view of overall supply chain data. The novel approach of using an open, peer-to-peer network based paradigm suggests that ad hoc associations between companies could form very quickly due to very little extra overhead. Costs of fully integrating a new company into the an existing supply chain are also very low when compared with customised proprietary or one-sided web site based solutions.

However, it was shown that the limitations of the underlying network flowed through to the supply chain implementation. SimpleP2P's issues of bandwidth and scalability also proved to impose this limitation on the supply chain simulation. This was particularly evident when searching for data on the network and swapping supply chain information between all nodes in the network. It was also shown that the security aspects of SimpleP2P may introduce new issues into the supply chain that would be a new consideration when compared to manual supply chain solutions.

The limited time available in this single semester thesis has prevented a more detailed analysis of data beyond a discussion of simulation. The gathering of real statistical data through a process where a peer-to-peer supply chain solution, such as the one presented here, were to run in parallel with an existing software based system would provide a more quantifiable and tangible result.

...

5.0 Further Work & Conclusion

Every composer knows the anguish and despair occasioned by forgetting ideas which one had no time to write down.
-- Hector Berlioz

In chapter three, we described the design and implementation of a peer-to-peer framework suitable for use in research and provided two example applications to demonstrate its suitability. The results of testing and a discussion of its feasibility for use in a supply chain application were discussed in chapter four.

This chapter will provide some possible options for further research in the areas explored and summarise the findings of this thesis.

5.1 Further Research

The development and testing of the peer-to-peer system developed, and its application in supply chains, has shown many potential avenues for further research and development.

5.1.1 Security in decentralised peer-to-peer networks

Security in peer-to-peer networks is still an area that needs significant research. This thesis introduced a method for maintaining a cut-down PKI suitable for authenticating nodes and managing certificate revocation. Peer-to-peer security is an interesting area as, depending on application, contrasting requirements often exist. In electronic commerce, identity and secrecy are vitally important whilst in systems such as Freenet, anonymity is paramount.

Attacks and vulnerability to compromise are problems common to all networked systems. Peer-to-peer systems are especially vulnerable due to their ability to provide a medium in which virii and worms could spread very quickly should a compromise be found.

5.1.2 Bootstrapping in decentralised peer-to-peer networks

Decentralised networks, by their very definition, have no central authority to manage where new nodes should connect to the network. This problem is somewhat addressed by networks such as Gnutella by bootstrapping where known nodes (through word-of-mouth, cache, etc.) advise connecting nodes of other potential connection candidates.

5.1.3 Searching and message routing in decentralised peer-to-peer networks

Packet routing is an established area of research in the Internet router domain. A long history of evolution has seen individual algorithms evolve to improve performance and support for a wide variety of network applications. An investigation into the similarity and differences of peer-to-peer routing needs and traditional network routing would be interesting to see if improvements could be made to the largely ad hoc routing system in popular peer-to-peer networks today.

5.1.4 Industry testing and review of supply chain system developed

As alluded to in the results and discussion, the system proposed for a peer-to-peer supply chain would benefit greatly from a real world implementation. Gathering of real data would allow a concrete comparison of performance between existing supply chain solutions and a peer-to-peer application. This would require the committed cooperation of several organisations or departments and would be a non-trivial exercise indeed!

Failing a full scale, controlled test installation, a concentrated coding and design effort on the user interface and documentation would potentially allow for ad hoc field testing by any interested party. Results could then be obtained through voluntary provision of log data and survey reports.

5.1.5 Addition of intelligent agents at decision points

Some simple logic in the form of decision on suppliers based on stock levels was shown in SupplyChainDemo (see chapter 3.4.2). This was given as an example of rudimentary processing being a part of XML transforms. This demonstrates how decisions can be made automatically within the electronic supply chain. An investigation of more complex decisions systems i.e. Rule based decision making, fuzzy logic, etc. could prove effective in applying business policy to supply chain dynamics.

5.1.6 Reputation management through network memory

Reputation management is an issue that was briefly considered in the security discussion within this thesis. As can be seen in many large scale e-commerce ventures, i.e. eBay, reputation of buyers and sellers plays a critical role in determining a measure of trust. This perceived trust often plays a large role in selecting potential suppliers or customers.

An interesting investigation would be to examine reputation management in a truly decentralised network. A possible approach would be to examine concepts from redundant peer-to-peer storage systems, i.e. Freenet, in order to maintain a database of node reputation data.

5.1.7 Various other applications for general communication and e-commerce

Supply chains were chosen as an application for implementation here due to the lack of open, implemented examples; despite their popular discussion in the literature. However, it is obvious that peer-to-peer programming could be applied to the wider e-commerce problem. Nodes exchanging data as a part of a financial transaction, along with supply chain data, is clearly possible. Other nodes could act as an intermediary for transactions e.g. Credit card transaction nodes could take credit card and order details and return results of the transaction to each node. Similarly, authentication, logistics and anonymity services could also be possible.

In light of other projects, such as RosettaNet (see chapter 2.5.2.2), to standardise message schemas between transacting parties online, a more general investigation of e-commerce using SimpleP2P as a lightweight option may prove interesting.

5.1.8 Compression of XML

Discussed in 4.2.1.1, compression of the XML message before transmission yielded favourable results for larger messages when compared with Gnutella and uncompressed XML. XML aware compression techniques have been investigated in projects such as XMILL (Liefke 1999) and XMLPPM (Cheney 2001). Both of these projects appear to have been left dormant for several years and may benefit from a fresh investigation in light of the continued expansion of XML in online communications.

5.1.9 Decentralised, Distributed Computing

Distributed processing is a popular an wide area of research. Recent work, such as Atkinson 2003, showed that good results can be obtained with relatively simple peers working with suitable parallel problems. Solutions such as this still rely on a client/server approach where a central authority manages communication between nodes and submissions of jobs etc. A peer-to-peer approach may be able to remove the potential central point of failure in these designs.

A shared memory approach may be feasible, again using concepts from the Freenet project, where processing tasks could be submitted to a decentralised network of nodes who maintain computation states and coordination in shared, redundant memory. A goal of this research would be to produce a computing network where any node can fail and the computing task can still be completed.

5.1.10 Categorisation of Data on Peer-to-Peer networks

During the testing of the bandwidth and compression performance, strings were captured to provide a set of data to set compression techniques on. The search strings themselves provided an interesting insight into what kinds of things the Gnutella network was being used for. I quickly categorised the strings gathered and provided a brief discussion of them in the appendices as a sideline for interested readers. A recent paper, Miller 2004, has attempted to categorise searches and files on the Gnutella network based on file types and claimed that the majority of searches over the network were audio files closely followed by video files. A different approach was taken in the appendices where the strings themselves were examined to determine the intention of the search. This produced different results to those of Miller and suggested that music was not the primary search target on Gnutella.

The results presented in the Appendices are presented as an interesting observation and are by no means the result of a thorough investigation (hence being included outside of this work). A formal investigation of the searches being performed across various file sharing networks, reaching beyond just Gnutella, may provide interesting results to support or refute the widely held, or at least advertised, belief that file sharing equals music piracy.

5.2 Summary

This thesis has presented an investigation into the feasibility of applying peer-to-peer computing paradigms to supply chains. As part of this investigation, a generic, XML based peer-to-peer network framework was developed and tested. A scenario analysis of a simulated supply chain was also performed using the shirt example as shown in 3.4.2.1. The approach taken was to ensure that classical problems, such as the bull whip effect, can first be simulated with the peer-to-peer solution and then tests were ran to see how these simulated issues could be overcome using the peer-to-peer tools that were built.

An investigation into the amount of bandwidth generated on searches on both the developed framework and Gnutella was provided. As a part of this investigation, the feasibility of using XML as a communication medium in a peer-to-peer network was undertaken finding that, through the use of compression, XML could preform no worse than binary structures when dealing with larger search strings. SimpleP2P was also tested for performance characteristics with the time taken for searches and network bootstrapping. Both of these results returned similar performance to the Gnutella specification.

An appraisal of the security aspects of the system was also undertaken. Confidentiality & Authentication in the network was based on secure sockets allowing all traffic on the peer-to-peer network to be encrypted. Authentication was managed through each node needing to have a SSL certificate signed by a certificate trusted by the network as a whole. This allows any node to have a measure of trust of another connecting one providing that they have a similarly signed certificate. Revocation of otherwise valid certificates was an issue that was found that needed to be dealt with. Standard x509 version 3 digital certificates allowed for revocation to come with a reason, but it was found not to be flexible enough for a peer-to-peer environment where reason for revocation must be clear so reputation is not damaged. An approach was taken where certificate revocation was managed through signed XML containing hashes of revoked certificate public keys being passed to a node during the network join or bootstrap process. Should the node wish for more a more trusted result, especially in the case of bootstrapping of a non-trusted node, a certificate revocation request can be sent through the network, just like a small search, as the request propagates nodes can respond and reassure the connecting node through their same responses. Possible attacks weaknesses of this security approach we also discussed as results.

The bull whip effect was simulated in the peer-to-peer network by having three retailer nodes each ordering a random number of products with a very low variation in orders. Each retailer ordered 20 plus or minus 2 items per day, yet hedged their orders at 2, 3 or 5 days. This hedging produced results as can be seen in red on the graph. These wild swings which hardly represent the constant demand at the retailer end are what is seen by the manufacturer traditionally. With the peer-to-peer solution, the upstream manufacturing nodes were able to see that the combined demand, which is shown in blue. This allowed the manufacturer to see that actual demand. Other scenarios were examined in the thesis including sales spikes and dips. Foreknowledge of spikes and dips as they occur allowed the manufacturer to respond appropriately with an increase or decrease in production which resulted in quantifiable savings.

In addition to the bull whip effect, several other results were also gained through the scenario analysis of the shirt example. Bandwidth in the shirt supply chain is based on how much data is being advertised by a node and how often it is requested by other nodes. Total values for daily bandwidth requirements were shown and discussed. Integration and multiple suppliers in a supply chain were shown by new button supplier nodes being added and it was shown how this data became instantly available to all of the other supply chain nodes. New features of supply chains through the peer-to-peer implementation were shown such as aggregation of data between button suppliers as was selection of a button supplier based on its inventory. Small XSL applications were also devised to show the possibilities of automatic decisions based on lead time or inventory.

It was also shown that data is able to be agreed on or transformed to suit each node in the network through XML transforms. This allowed disparate data inputs from 3 different button manufacturers to be standardised for the shirt manufacturer.

Openness of the network was its greatest strength and weakness in a supply chain environment. The open availability of information allows companies to inter operate quickly and with minimum effort. However this openness may create issues with companies that do not want to share this kind of information for fear of it being used against them.

It was also found that the supply chain implementation still suffered from underlying problems of the peer-to-peer network. Network scalability and bandwidth requirements pose issues for very large supply chain networks due to the nature of the decentralised implementation. Also new problems such as worms and viruses that may run over the network introduce a new dimension that has certainly not been a problem with manual supply chains. Also, and possibly the most important thing to recognise with peer-to-peer networks and supply chains, is that it is only going to solve issues where information access is the problem. However, one could argue that supply chains with other, non-information related problems, may be helped with this as it does remove the information propagation doubt – leaving one less variable in the equation when looking for a problem.

In conclusion, this thesis presented a thorough investigation into using decentralised peer-to-peer networks as a solution to supply chain communication. Its provision of a concrete implementation of a supply chain with appropriate testing and results sets it apart from other papers that have only described a solution or presented a feasibility discussion. The discussion of the results and the further work presented earlier in this chapter, show that peer-to-peer supply chains are worthy for continued research with a view towards a commercial trial and implementation.

.o0o.

6.0 References

Aberer, K., Despotovic, Z. 2002, '*Managing trust in a peer-2-peer information system',* Proceedings of the Tenth International Conference on Information and Knowledge Management.

Aberer, K., Cudré-Mauroux, P., Datta, A., Despotovic, Z., Hauswirth, M., Punceva, M., Schmidt, R., Wu, J. 2002, '*Advanced Peer-to-Peer Networking: The P-Grid System and its Applications',* Distributed Information Systems Laboratory, École Polytechnique Fédérale de Lausanne (EPFL). Viewed 14 June 2004, <www.p-grid.org/Papers/PIK2002.pdf>.

Alodar Systems 2004, Supply Chain Management (SCM) Demo, Viewed 14 June 2004, <http://www.alodar.com/scm/demo.html>.

Bartelt, A., Lamersdorf, W. 2002, '*Agent-Oriented Concepts to Foster the Automation of E-Business'*, Proceedings of the 11th International Workshop on Database and Expert Systems (DEXA 2000)

Bergner, M. 2003, '*Improving Performance of Modern Peer-to-Peer Services'*, Master Thesis, Umea University, Sweden.

Blake, M.B. 2001, '*Innovations in Agent-Based Business-to-Business Interoperability'*, Workshop on Agent-Based Approaches to B2B at the Fifth International Conference on Autonomous Agents (AGENTS 2001) pp 1-6, Montreal, Canada, May 2001

Blake, M.B. 2002, '*B2B Electronic Commerce: Where Do Agents Fit In?'*, Proceedings of the AAAI-2002 Workshop on Agent Technologies for B2B E-Commerce, Edmonton, Alberta, Canada, July 28, 2002

Burdett, D. 2000, '*RFC 2801: The Open Internet Trading Protocol'*, Viewed 1 Oct 2003, <http://www.faqs.org/rfcs/rfc2801.html>.

Carlsson, C., Fuller, R. 2001, '*Reducing the Bullwhip effect by means of intelligent, soft computing methods'*, Proceedings of the 34th Hawaii International Conference on System Sciences, 2001.

Cheney, J. 2001, '*XMLPPM: XML-Conscious PPM Compression'*, Viewed 16 June 2004, <http://www.cs.cornell.edu/People/jcheney/xmlppm/xmlppm.html>

Chopra, S., Meindl, P. 2004, '*Supply Chain Management. Strategy, planning and Operations'*, 2nd Ed, Prentice Hall, New Jersey.

Clarke, I., Miller, S., Hong, T., Sandberg, O., Wiley, B. 2002, '*Protecting Free Expression online with Freenet'*, IEEE Computing, Jan-Feb 2002.

Cohen, B. 2003, '*Incentives Build Robustness in BitTorrent*', Viewed 1 Oct 2003, <http://bitconjurer.org/BitTorrent/bittorrentecon.pdf>.

Datta, A., Hauswirth, M., Aberer, K., '*Beyond "web of trust": Enabling P2P E-commerce*', Proceedings of the IEEE Conference on Electronic Commerce (CEC'03), June 24-27 2003, Newport Beach, California, USA.

Eschenbächer, J., Knirsch, P., Timm, I. J. 2000, '*Demand Chain Optimisation by Using Agent Technology*'. In IFIP WG 5.7 - International Conference on Integrated Production Management: Information and Communication Technology in Logistics and Production Management, Tromso, Norway, pages 285-292. 2000.

Gnutella 2003, '*The Gnutella Protocol Specification v0.4*', Viewed 1 Oct 2003, <http://www9.limewire.com/developer/gnutella_protocol_0.4.pdf>.

Harold, R. H. 2001, *'XML Bible'*, 2nd Edition, Hungry Minds Inc., New York.

He, M., Jennings, N. R., Leung, H. 2003, '*On agent-mediated electronic commerce',* IEEE Trans on Knowledge and Data Engineering 15 (4) 985-1003.

Houlihan, J. B. 1985, '*International Supply Chain Management*', International Journal of Physical Distribution and Materials Management, 15, 1, 22-38.

Huget, M. 2002, '*An Application of Agent UML to Supply Chain Management',* Viewed 14 June 2004, <http://www.csc.liv.ac.uk/research/techreports/tr2002/tr02015abs.html>.

Ibrahim, I. K., Schwinger, W.,Weippl, E., Altman, J.,Winiwarter W. 2001, '*Agent Solutions for E-Business Transactions*', 12th International Workshop on Database and Expert Systems Applications September 03 - 07, 2001 Munich, Germany

IANA 2004, '*Port Numbers*', Viewed 14 June 2004, <http://www.iana.org/assignments/port-numbers>.

IETF 2003, '*Public-Key Infrastructure (X.509)',* Viewed 1 Oct 2003, <http://www.ietf.org/html.charters/pkix-charter.html>.

Karbhari, P., Ammar, M., Dhamdhere, A., Raj, H., Riley, G., Zegura, E. 2003, '*Bootstrapping in Gnutella: A Measurement Study*', Georgia Institute of Technology, Atlanta, USA.

Liefke, H. 1999, '*XMill – An Efficient Compressor for XML*', Viewed 14 June 2004, <http://www.research.att.com/sw/tools/xmill/>.

Minar, N., Burkhar, R., Langton, C., Askenazi, M. 1996, '*The Swarm Simulation System: A Toolkit for Building Multi-agent Simulations*', Viewed 14 June 2004, <http://xenia.media.mit.edu/~nelson/research/swarm/>

Minar, N. 2001, '*Distributed Systems Topologies*', OpenP2P, O'Reilly, Viewed 14 June 2004 , <http://www.openp2p.com/pub/a/p2p/2001/12/14/topologies_one.html>.

Milojicic, D., Kalogeraki, V., Lukose, R., Nagaraja, K., Pruyne, J., Richard, B., Rollins, S., Xu, Z. 2002, '*Peer-to-Peer Computing*', Hewlet Packard.

Miller, J. 2004, '*Characterization of Data on the Gnutella Peer-to-Peer Network*', IEEE Consumer Communications and Networking Conference, January 2004.

MIT Media 2001, '*The Atomic Market',* Viewed 1 Oct 2003, <http://aex.media.mit.edu/>.

MLDonkey 2003, '*The mlDonkey Homepage*', Viewed 1 Oct 2003,

<http://www.nongnu.org/mldonkey/>

OpenNap 2003, '*OpenNap: Open Source Napster Server*', Viewed 1 Oct 2003 , <http://opennap.sourceforge.net/>.

OpenSSL 2004, '*OpenSSL: The Open Source toolkit for SSL/TLS*', Viewed 14 June 2004 , <http://www.openssl.org/>.

Ripeanu, M. 2001, '*Peer-to-Peer Architecture Case Study: Gnutella Network*', Computer Science Department, The University of Chicago.

Ritter, J. 2001, '*Why Gnutella Can't Scale. No, Really*', Viewed 14 June 2004, <http://www.darkridge.com/~jpr5/doc/gnutella.html>

Savit, R. 1999, '*Agent Based Models of Supply Chains*', Viewed 1 Oct 2003, <http://www.ise.ufl.edu/Supplychain/done/Day3/Savit/>.

Schary, P. B., Skjott-Larsen, T. 1995, '*Managing the Global Supply Chain*', Handelshojskolens Forlag, Copenhagen.

Schneier, B. 1994, *'Applied Cryptography: Protocols, Algorithms and Source Code in C',* John Wiley & Sons, Inc., New York, USA.

Schneier, B., Ferguson, N. 1999, '*A Cryptographic Evaluation of Ipsec',* Viewed 14 June 2004, <http://www.schneier.com/paper-ipsec.html>.

W3C 2004, *'Extensible Markup Language (XML)'*, Viewed 14 June 2004, <http://www.w3.org/XML/>.

Wheeler, D. A. 2001, '*More Than a Gigabuck: Estimating GNU/Linux's Size*', Viewed 14 June 2004, <http://www.dwheeler.com/sloc/redhat71-v1/redhat71sloc.html>.

Youll, J. 2001, '*Peer to Peer Transactions in Agent-mediated Electronic Commerce',* Master's Thesis, MIT.

Yucesan, E., Van Wassenhove, L. N. 2002, *Supply-Chain.Net: 'The impact of web-based technologies on supply chain management'*, Insead R&D, Fontainebleau, France.

...

7.0 Appendices

7.1 Sales Data for Bull Whip Experiment

Daily Sales

Daily	Wholesaler 1	Wholesaler 1	Wholesaler 1	Manufacturer Orders
1	20	22	21	63
2	20	22	21	63
3	22	21	21	64
4	21	21	22	64
5	20	22	21	63
6	21	21	22	64
7	21	22	21	64
8	21	22	21	64
9	20	21	20	61
10	21	21	20	62
11	21	21	22	64
12	20	21	21	62
13	21	21	20	62
14	22	22	21	65
15	22	22	21	65
16	21	21	22	64
17	20	21	21	62
18	21	22	21	64
19	20	22	20	62
20	20	21	21	62
21	20	22	22	64
22	20	22	21	63
23	22	21	22	65
24	20	21	21	62
25	20	22	22	64
26	22	20	21	63
27	22	20	20	62
28	22	21	21	64

Batched Sales

Daily	Wholesaler 1	Wholesaler 2	Wholesaler 3	Manufacturer Orders
1	0	0	21	21
2	0	0	0	0
3	62	0	42	104
4	0	0	0	0
5	0	108	43	151
6	62	0	0	62
7	0	0	43	43
8	0	0	0	0
9	62	0	42	104
10	0	107	0	107
11	0	0	40	40
12	62	0	0	62
13	0	0	43	43
14	0	0	0	0
15	65	107	41	213
16	0	0	0	0
17	0	0	43	43
18	62	0	0	62
19	0	0	42	42
20	0	107	0	107
21	60	0	41	101
22	0	0	0	0
23	0	0	43	43
24	62	0	0	62
25	0	108	43	151
26	0	0	0	0
27	64	0	43	107
28	0	0	0	0

7.2 Data for supply chain bandwidth example

```
<simplep2p>
      <company>Warp Thread Pty Ltd</company>
      <inventory>10000</inventory>
      <production>1000</production>
      <lead_days>3</lead_days>
</simplep2p>
<simplep2p>
      <company>Fill Thread Pty Ltd</company>
      <inventory>100000</inventory>
      <production>10000</production>
      <lead_days>3</lead_days>
</simplep2p>
<simplep2p>
      <company>Stitching Thread Pty Ltd</company>
      <inventory>100000</inventory>
      <production>10000</production>
      <lead_days>3</lead_days>
</simplep2p>
<simplep2p>
      <company>Fred's Fabric Pty Ltd</company>
      <inventory>987654</inventory>
      <production>1000</production>
      <lead_days>3</lead_days>
</simplep2p>
<simplep2p>
      <company>ABC Buttons Inc</company>
      <diameter>8</diameter>
      <material>acrylic</material>
      <sew_holes>2</sew_holes>
      <colour>clear</colour>
      <inventory >1234</inventory>
      <production>100</production>
      <lead_days>3</lead_days>
</simplep2p>
<simplep2p>
      <company>The Shirt Company Pty Ltd</company>
      <product>long sleeve shirt</product>
      <size>46</size>
      <colour>white</colour>
      <style>buttoned collar with reinforced cuffs</style>
      <desc>http://shirtcompany.com/product.php?id=12345</desc>
      <inventory>10000</inventory>
      <production>1000</production>
      <lead_days>3</lead_days>
</simplep2p>
<simplep2p>
      <company>Shirt Wholesaler Inc</company>
      <inventory >1234</inventory>
      <avg_daily_sales>967</avg_daily_sales>
      <sales_to_date>872346</sales_to_date>

</simplep2p>
```

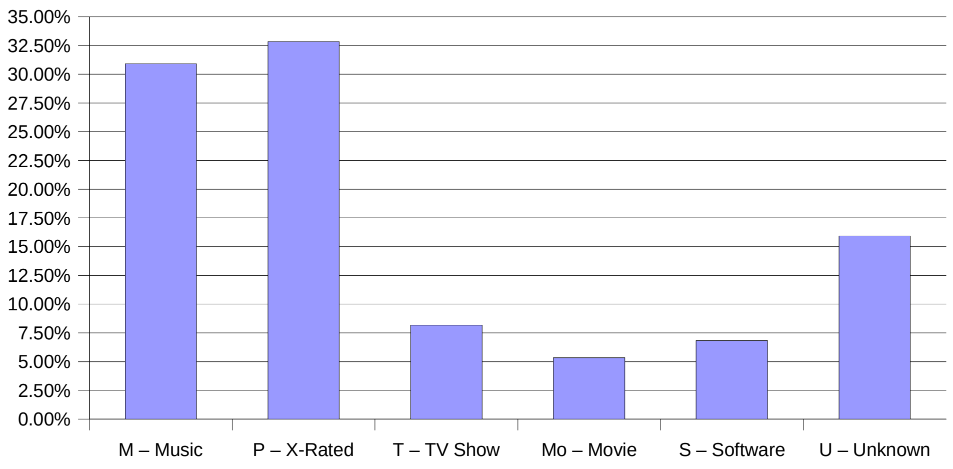
7.2 Categorisation of Search Strings Observed

For the XML compression experiments presented in 4.2.1.1 a corpus of actual strings from the Gnutella network were required if an accurate comparison between the two protocols were to be undertaken. These search strings were taken from the Gnutella network over 3 20 minute sessions at 3 random times on 3 separate days.

These strings were then categorised into 6 different categories (as shown on the x-axis on the graph) by the author by the following simple method:

1. If the string is obviously related to a category, assign the string to that category.
2. If unsure, search for the string on Google. If the top three results agree, assign the string to that category inferred from the search.
3. If unknown place it in the unknown category.

This exercise produced the results in the graph below:



These results were included in this appendix as an interesting aside as an informal indication of what the Gnutella network is being used for.

7.3 Source Listing

Please see the accompanying CDROM for a full source code listing.