UNIVERSITY OF SOUTHAMPTON
Faculty of Engineering and Applied Science
Department of Electronics and Computer Science

A project report submitted for the award of

BSc Computer Science with Distributed Systems and Networks

Project supervisor: Dr David E Millard

Second examiner: Dr Alun S Vaughan

SotonOne Project

by Dimitrios Michelinakis

8th May 2003

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

Faculty of Engineering and Applied Science
Department of Electronics and Computer Science

A project report submitted for the award of
BSc Computer Science with Distributed Systems and Networks

by Dimitrios Michelinakis

Existing distributed systems do not provide a viable solution for information distribution over wireless ad-hoc networks. Existing implementations are missing vital features for a decentralised network and information caching. Distributed systems over ad-hoc networks require a solid service implementation which is portable enough to reliably distribute arbitrary data. This projects aims to solve these problems by designing and implementing a service which offers information distribution based on location awareness over decentralised networks. The service will also provide advanced features for object caching, pre-emptive caching and location awareness. The solution offered in this report is a solid software implementation which can be used in real life situations, over large or small areas with wireless connectivity, to distribute information between peers.


SotonOne Project

Table of Contents

TABLE OF FIGURES ..... **v**

ACKNOWLEDGMENTS ..... **vi**

CHAPTER 1 INTRODUCTION **1**

1.1 ABOUT THE PROJECT 1
1.2 HOW THIS REPORT IS ORGANISED 2

CHAPTER 2 BACKGROUND **4**

2.1 P2P TECHNOLOGIES 5

CHAPTER 3 PROJECT SPECIFICATION **7**

3.1 RAD PROCESS MODEL 7
3.2 REQUIREMENTS SPECIFICATION PROCESS 9
3.3 REQUIREMENTS SPECIFICATION DOCUMENT 10
3.4 ENVIRONMENT AND PEER INTERACTION 11
3.5 SEARCH QUERIES AND REPLIES 13
3.6 LOCATION AWARENESS 13
3.7 CACHING AND PRE-EMPTIVE CACHING 14

CHAPTER 4 PROJECT DESIGN AND IMPLEMENTATION **17**

4.1 RISK ANALYSIS 17
4.2 FUNCTIONAL DESIGN DESCRIPTION 18
4.3 USER INTERFACE PROCESSING 19
4.4 PROCESS AND CONTROL 20
4.5 INPUT PROCESSING 22
4.5.1 Discovery Manager *22*
4.5.2 Location Manager *22*
4.5.3 Peer Input Manager *24*
4.5.4 Proxy Manager *25*
4.6 OUTPUT PROCESSING 26
4.7 LOCAL STORAGE 27
4.8 SEARCH AND REQUEST OBJECTS 27

CHAPTER 5 BEHAVIOURAL DESCRIPTION **30**

5.1 SEARCH QUERY SEQUENCES 30
5.2 OUTGOING SEARCH QUERY THROUGH LOCAL CACHE 30
5.3 OUTGOING SEARCH QUERY WITHOUT LOCAL CACHE 31
5.4 INCOMING SEARCH QUERY 32

CHAPTER 6 TESTING **34**

6.1 DESCRIPTION OF THE SIMULATION TEST 34
6.2 LOGISTIC REGRESSION 35
6.3 APPLICATION OF THE DATA ON THE LOGISTIC MODEL 35
6.3.1 Fitting the logistic model in Dataset 1 ..... *36*
6.4 RESULTS OF THE SIMULATION TEST 37
6.5 BLACK-BOX TESTING 37

CHAPTER 7 CONCLUSION **39**

7.1 THE ROAD AHEAD 40

REFERENCES **46**

BIBLIOGRAPHY **46**

APPENDICES **46**

Table of Figures

Figure 3.4-1 - Use Case for search and location 11
Figure 3.4-2 - Use Case for add and remove objects 12
Figure 3.4-3 - Use Case for remote searches 12
Figure 3.7-1 - Use Case for search with add 15
Figure 4.2-1 - Architecture Context Diagram 18
Figure 4.3-1 - SotonOne UML class 19
Figure 4.3-2 - SotonOneServer UML class 19
Figure 4.4-1 - Agent classification 20
Figure 4.4-2 - QueryAgent UML class 21
Figure 4.5-1 - DiscoveryManager UML class 22
Figure 4.5-2 - LocationManager UML class 23
Figure 4.5-3 – AbsTwoDimensionalLocation UML class 23
Figure 4.5-4 - AbsThreeDimensionalLocation UML class 23
Figure 4.5-5 - Location UML class 24
Figure 4.5-6 - PeerInputManager UML class 24
Figure 4.5-7 - ProxyManager UML class 25
Figure 4.6-1 - PeerOutputManager UML class 26
Figure 4.7-1 - CacheManager UML class 27
Figure 4.8-1 - Search query message 28
Figure 4.8-2 - Unsuccessful reply message 28
Figure 4.8-3 - Successful reply message 28
Figure 5.2-1 - Outgoing search query through local cache 31
Figure 5.3-1 - Outgoing search query without local cache 32
Figure 5.4-1 - Incoming search query 33
Figure 6.3-1 - SotonOneSimulation example 35
Figure A.1 - SotonOne peer on RedHat Linux 9 47
Figure A.2 - Object browser 48
Figure A.3 - Cache storage 48
Figure A.4 - Status monitor 49
Figure A.5 - Discovery monitor 49
Figure A.6 - Web browser example 50

Acknowledgments

This report is dedicated to my family for their unwavering trust and support.

I also wish to extend my sincere thanks and appreciation to my tutor Dr John Carter for his support over the years, my project supervisor Dr Dave Millard for his useful comments on my report and Nikolaos Tzavidis for his help and suggestions.

In memory of my uncle Nikolaos Paravantis

Chapter 1

Introduction

The internet is an electronic network which spans across the earth via a multitude of mediums, from cable, wireless, or even satellite connections. Even though the internet does go through these theoretical geographical regions and falls under the rules of each particular country or nation, it follows an additional set of its own rules.

The internet has been governed by the Client and Server model implementation, where the population consists of clients, and the governing bodies of servers. The clients are then offered a set of services by the servers. These services can be regulated based on the rules imposed by geographical locations, although that is not the case in the real world. The server services are governed by many sets of rules, either restrictions of use by the hardware of the server, or restrictions by the administrators controlling the servers.

Up until recently clients had to obey the rules of a server in order to use it and they had no abilities of their own, other than to connect to the servers. That is no longer the case. With the introduction of broadband services, un-metered access and always-on internet connections the internet has reached a turning point. Clients are not just clients anymore, their ability to act as a server has given internet users the freedom to offer their own set of services based on their own rules.

In addition, new mobile devices now possess some form of network capability, thus the mobile device sector has been given the opportunity to offer similar services. At first mobile devices acted as clients only, but with the introduction of GPRS[1], high-speed always-on mobile connectivity, the mobile device has become an important asset. This project takes advantage of these mobile devices and the new technologies in wireless communication.

1.1 About the project

The objective of this report is to specify, design, and implement a Peer-to-Peer service which allows distributed systems on ad-hoc networks to share information based on location awareness. The project aims to use current technologies, and to build a new service of information distribution on top of those technologies. “A distributed system

[1] GPRS: General Packet Radio Service. Data service for mobile telephone networks.

is one which the failure of a computer you didn't even know existed can render your own computer unusable" (quote attributed to Leslie Lamport), and that is an effect of distributed systems this project tries to prevent from happening.

This report is about a P2P[2] service. The term P2P is very hard to define because of its abstract nature. Although peer to peer systems already existed for many years, the meaning of the term P2P has changed. P2P systems could be any computer connected to another computer, since they form a peer to peer network, also many current protocols like NNTP and DNS work as peer to peer. The current meaning of P2P was introduced during the late 90's by the early Gnutella[3] pioneers. P2P is now defined as "a style of computing that makes the network interactions more symmetrical", thus P2P is no longer about centralised or decentralised networking, but it is about the application and services of the peer.

The project makes use of current technologies to achieve its goal. Thus this report is not a description of the available technologies, instead it uses these technologies as tools in order to design and implement the final software product.

The project was deemed important enough to release to the open source community for future development. Thus, the project has been published as open source under the Sun Project JXTA Software License on Sun's JXTA[4] web site; please visit http://www.jxta.org for more information.

The university spotlight page on Sun's JXTA web site features the author and the development of this report; http://www.jxta.org/universities/southampton.html

1.2 How this report is organised

A structured software engineering process has been followed throughout the specification, design and implementation lifecycle of the project. The structure of this report does not match the actual timeline used to build the project, but it follows a structured approach to allow for a better understanding of the intricacies of the system.

Due to the nature of this project diagrams are used in a variety of places to describe the system and its behaviour. A choice was made to use UML[5] as the standard modelling language throughout this report. UML is the successor to the wave of object-oriented analysis and design methods that were used during the late 80's and early 90's, it unifies several existing methods into one, and has become an OMG[6] standard (Fowler and Scott 2000).

[2] P2P: Peer-to-Peer. **peer** [piə] n. an equal in rank, merit or quality (Hornby 1952).
[3] Gnutella: P2P protocol developed by Nullsoft, which was later acquired by AOL.
[4] JXTA: Project JXTA. http://www.jxta.org (Last access date: 9 January 2003).
[5] UML: Unified Modeling Language. Object oriented analysis and design modelling language.
[6] OMG: Object Management Group. Organisation on the standardization of analysis/design methods.

UML is the most current and formally acceptable by the international community as the standard for software modelling (Fowler and Scott 2000). SSADM though a viable option is not as easy to understand and time constraints on the project did not allow its use. RUP[7] was another option but was quickly dropped due to the commercial nature of the tools used to design the system.

This report is structured as follows: Chapter 2 is about the background and history of similar systems and a detailed analysis of the available technologies. Chapter 3 is the specification and analysis of the project. Chapter 4 is a detailed design and a throughout implementation of the project. Chapter 5 explains the interaction and methodology of the implementation. Chapter 6 goes through the rigorous testing of the implementation. Chapter 7 is the conclusion which also talks about any future possibilities of the project.

[7] RUP: Rational Unified Process. Unified object oriented design process made by the authors of UML.

Chapter 2

Background

The change in the client/server model became apparent during 1999. At first, the hardware available to the home user became cheaper and more powerful, the mainframe computer stopped been cost-effective, since the home PC became powerful enough to match mid-range servers.

Dial-up networking to the internet was surpassed during the late 90's when broadband connections became available, and always-on internet became a low cost possibility for home users. Along with the powerful processing power of the home computer, storage greatly improved in capacity. At first storage was a bottleneck where disk space was not enough, hard drive transfer rates and latency were causing delays in transactions. With the introduction of cheap high-speed high-capacity hard drives storage is no longer an issue. With the introduction of low cost storage and P2P sharing networks, the only question that remains is; where does it reside in a globally shared infrastructure?

P2P networking has given us the Servents[8], a combination of client and server, which offers as well as requests information. The home user now has the ability to create, offer, and distribute content and services over the internet, under the restrictions of the ISP[9]. A home user on an always-on connection may run his own email server, web server, news server and almost every other server service. Static IP addresses may help but they are no longer required because free dynamic DNS addresses are provided by organisations like DynDNS[10].

P2P systems first became famous in early 2000 with Napster, one of first popular MP3 file sharing programs. Napster became so famous that it caused its own downfall, "according to industry tracker Webnoize the Napster user base was estimated at 1.6 million users in February 2001. The exchange of MP3 files that month alone reached 2.7 billion" (Flenner 2003). The media corporate giants fought against the small Napster start-up company; finally the court ruled against Napster and suspended file sharing on July 2001. However, the end of Napster was not the end of P2P, but it was the start of decentralised P2P networking and of a multitude of new technologies like Pervasive computing, Agents, Ubiquitous Computing and the Grid.

[8] Servents: Term used within the Gnutella and P2P community for nodes that are both server and client.
[9] ISP: Internet Service Provider.
[10] DynDNS: www.dyndns.org, dynamic and static domain name services.

2.1 P2P Technologies

This project is based on top of many existing technologies. It would have been impossible to re-invent new technologies and implement them all to their full potential within the project time constrains. Thus several existing technologies had to be selected among the numerous available technologies.

JXTA has been selected as the core communication module; it takes care of the device independence and low level communication between the devices. JXTA has chosen over three other candidate Peer-to-Peer network solutions, the Freenet[11] implementation, the FastTrack[12] implementation and the Gnutella protocol.

The Freenet implementation was developed by Ian Clarke, an undergraduate at Edinburgh University, and released in June 1999 as a 3rd year project. Freenet's architecture focuses on freedom of speech by implementing privacy, security and anonymity. Even though Freenet is a solid implementation, it does not allow for custom services to be implemented on top of the existing network, thus it was not a viable option.

FastTrack is a new protocol first released to the public in March 2001 by a team of programmers from Amsterdam, in the Netherlands, implemented on the KaZaA client. On the 21st of January 2002 the FastTrack network suddenly changed ownership to Sharman Networks, an off-shore company specifically created to avoid the law. The ownership of FastTrack was split between Sharman Networks, AltNet (formerly Brilliant Digital) and Blastoise Joltid. FastTrack uses a similar implementation to that of Gnutella; it uses HTTP as an underlying communication protocol. FastTrack is not a viable option since it is a closed source protocol and requires a license to connect to the network.

The Gnutella protocol was developed by Nullsoft[13], which later because a subsidiary of America Online, and was released to the public on the 14th of March 2000. It was withdrawn by America Online on the 10th of April of the same year after it was declared a rogue project. During the availability time the application had already been downloaded and distributed to the public, thus once the project was withdrawn the Gnutella protocol was reverse engineered and new Gnutella clients were developed.

The JXTA specification was selected over the Gnutella protocol because of several reasons, some of them are:

- JXTA takes care of the low level communication, thus the developers only have to worry about the actual application.

- Gnutella uses the HTTP[14] protocol which has reached its limit in features; instead JXTA uses custom protocols which can be configured for custom uses.

[11] Freenet: Freenet Project. http://freenet.sourceforge.net (Last access date: 9 January 2003).
[12] FastTrack: History. http://www.slyck.com/fasttrackhistory.html (Last access date: 9 January 2003).
[13] Nullsoft: Authors of Gnutella, later acquired by AOL.
[14] HTTP: Hyper Text Transport Protocol: RFC specification: 1945 (v1.0) and 2616 (v1.1).

- JXTA is a specification, not an implementation, thus it is platform and language independent. Based on the specification a developer may choose any programming language under any platform.

JXTA has been implemented in Java and C, also work is undergoing for Perl and Python implementations. Since JXTA is the most suitable choice of P2P protocols, the choice for the JXTA implementation lies between Java and C. The choice of Java was made because of the faster development lifecycle of Java, due to the time constraints of the project.

Chapter 3

Project Specification

This chapter describes the specifications for the SotonOne project. It gives a clear depiction of the specifications and interprets the requirements behind them. A detailed analysis of the architecture is then given, based on those requirements.

3.1 RAD Process Model

This project was designed and implemented under a formal software engineering process. Computer science deals with the hardware design, theorems and computer algorithms. Software engineering looks at computer science as a tool, and applies that tool to real life problems. This method has seven key factors which affect the software engineering process, economics, user interfaces, networking, time to market, waterfall model problems, object oriented technology and desktop computing (Wasserman, 1996).

The first step in the software engineering practice is to define a process model, or life cycle. A process model usually involves a set of techniques and tools, which are used in a predefined way throughout the life of the project. The process defines all major activities and resources, and it is composed of sub-processes which are inter-linked with a beginning and an ending. Activities are then organised in sequences based on priority and dependencies. Throughout the process there are constraints and rules which apply to all the entities of the process. Some of the major process models are:

Process Models

- Waterfall model
- V model
- Prototyping model
- Operation Specification model
- Transformation model
- Phased development model
- Spiral model

This project is based on a Rapid Application Development linear sequential software development process model (Pressman 1997). The description and comparison of the

models is out of the scope of this document, but it is important to state the reason why the RAD model was the most applicable to this project. The RAD model offers an important feature that no other model offers, and that is an extremely short development cycle. Due to the fact that this project has very limited time constraints, very limited resources and the whole development was done by a single person, the choice of the RAD model was the most appropriate.

The RAD process model encompasses the following phases (Kerr and Hunter 1994):

- Business modelling. The information flow among business functions is modelled in a way that answers the following questions: What information drives the business process? What information is generated? Who generates it? Where does the information go? Who process it? The basic model is specified in sections 3.3 and 3.4.

- Data modelling. The information flow defined as part of the business modelling phase is refined into a set of data objects that are needed to support the business. The characteristics of each object are identified and the relationships between these objects are defined. Data modelling is identified throughout this chapter.

- Process modelling. The data objects defined in the data modelling phase are transformed to achieve the information flow necessary to implement a business function. Processing descriptions are created for adding, modifying, deleting, or retrieving a data object. Process modelling is achieved in the early stages of the design in chapter 4.

- Application generation. RAD assumes the use of fourth generation techniques. Rather than creating software using conventional third generation programming languages, the RAD process works to reuse existing program components (when possible) or create reusable components (when necessary). In all cases, automated tools are used to facilitate construction of the software. Application generation is described in the implementation of the project in chapter 4.

- Testing and turnover. Since RAD process emphasizes reuse, many of the program components have already been tested. This reduces overall testing time. However, new components must be tested and all interfaces must be fully exercised. Testing is analysed in chapter 6.

One of the very early steps in the software development cycle was to create a project schedule of activities and milestones. A project schedule is a description of the project split into different activities which are dependant of each other for the ultimate completion of the project. Milestones in the lifecycle of the project mark critical development turning points. These mark the finish of a key feature which allows further development of the project to continue without any dependencies. An initial project schedule was submitted with the progress report on the 10th of January 2003. The complete schedule has been included in the appendix of this report.

The initial project schedule was correctly followed throughout the project, although many tasks had to be rearranged, and some new tasks were added. Most importantly, both the location awareness module and the pre-emptive cache module had to be moved to the end of the project schedule. The initial suggestion to work on a location awareness system so early in the project was a misjudgement, due to the fact that the communication module had to be completed before hand. Also, because the pre-emptive cache module depends on the location module it had to be moved as well.

The most important addition to the project schedule was the implementation of a secondary application, the SotonOneServer peer, which covers the requirements to predefine locations with initial content of information to share to peers.

The project had several setbacks, mainly during the implementation process. One of the most important setbacks was the upgrade from JXTA v1.0 libraries to JXTA v2.0 libraries. The upgrade caused a large number of the API to become deprecated and thus required at least two weeks of work to learn and implement the new v2.0 API.

3.2 Requirements Specification Process

Our understanding of system intent and function starts with the examination of the requirements. Requirements elicitation is the first step of the requirements process, which is the understanding of what the project is about. At first we separate the requirements into three categories as stated in Pfleeger (1998): requirements that absolutely must be met, requirements that are highly desirable but not necessary and requirements that are possible but could be eliminated.

The project is about a distributed system on ad-hoc networks based on location awareness. The system must be able to run decentralised, without requirements for a single point of contact. It must tolerate failure of communication, thus run on ad-hoc networks. Finally it should take into account location information provided by the peer or the environment around the peer.

The information that is transferred via this system must be abstract, and can be any type of data, binary, structured documents, like XML[15] or plain text. Information distributed via the system must have detailed information about their originating location or the location they belong to.

Information should be accessible via a user interface, but the actual implementation should be independent of a user interface thus it should be portable enough to be used under a multitude of different implementations. The system may be possible to run on mobile devices.

[15] XML: Extensible Markup Language. http://www.w3.org/XML/ (Last access date: 19-04-2003)

3.3 Requirements Specification Document

The specification requirements are split into sub-categories to distinguish between different requirements and areas of implementation. Each requirement includes a letter which marks it as a [F]unctional or [N]on-function requirement. Functional requirements describe an interaction between the system and its environment. Non-functional requirements describe a restriction on the project that limits the choices for constructing a solution to the functional requirements.

Physical Environment
[F] The equipment to operate the system is distributed on a university campus.
[F] There is only one location that the system exists in.

Interfaces
[F] The input is coming from many different peer systems and one location system.
[F] The output is based on the input. A single input receives a single output.
[F] Output for queries made by the peer itself generates multiple outputs to all available peers.
[N] The transport method of the data is over a TCP/IP network.
[N] The underlying service is portable enough to use different user interfaces.
[N] The system may be used remotely or locally by a browser, without the need for a user interface.

Users and Human Factors
[F] The system will be used by people walking around a university campus.
[F] It should be fairly easy for a user to understand the system.

Functionality
[F] The system will distribute information.
[F] The information will be distributed on user request, or based on pre-emptive cache.

Data
[F] Data distributed by the system can be binary or text, of any size.
[F] Data may be cached based on certain rules.
[F] All data should contain location information.

Resources
[F] The location information must be constructed before adding data to the system.
[F] The offered data must be setup prior to using the system.
[F] Wireless network coverage must be up and running.
[F] Peer software must be distributed to users around the area of the system.
[N] Deadline for the project is the 8th of May 2003.

Security
[F] Access to the system is not controlled in any way.
[F] All data distributed in the system are publicly available.
[N] Peers should not reveal their location in anyway to other peers.
[N] Peers may be limited in their ability to add new data to the system.

3.4 Environment and Peer Interaction

The specified requirements are consolidated into two peer systems. The first peer system should be able to provide a local cache, a search facility for local and remote searches, and a location module. This peer was named SotonOne, after the name of the project. The second peer system is only required to initially provide peers with content and offer no other functionality. To accommodate that requirement a choice was made to create a thin-peer based on the first peer system which does not provide any user interaction other than to add or remove content, and to specify the current location. Although it is a stand alone peer, its purpose is to initially provide the network with content, thus it was called a peer server, named SotonOneServer.

The system has three actors defined; the 'user', the 'peer' and the 'web browser' as seen in Table 3.4-1.

Actors	Role
User	User or administrator
Peer	Other peers on the network
Web Browser	User controlled web browser

Table 3.4-1 - Actors

The 'user' actor is any type of user or administrator who is using the peer, the difference between a plain user and an administrator is the administrator is allowed to setup the initial state of the network. The first ability of a 'user' actor is to make search queries. The second ability of a 'user' actor is to manually change their current location. Both abilities can be seen in Figure 3.4-1.

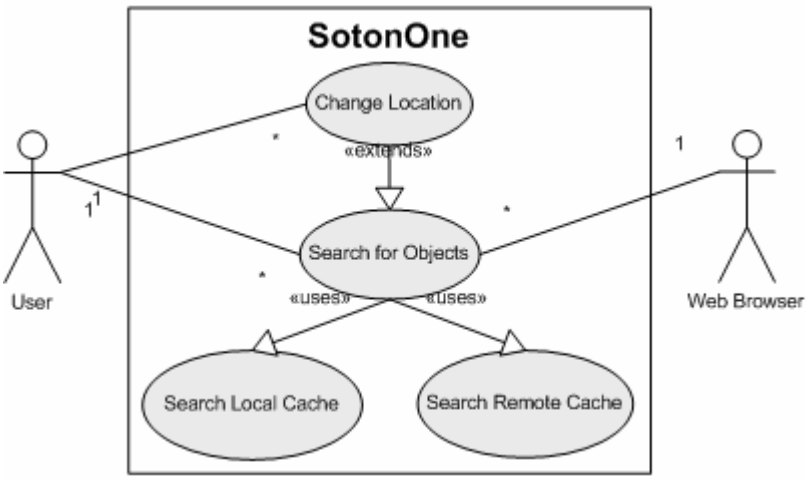

Figure 3.4-1 - Use Case for search and location

The third ability of the 'user' actor is to add and remove content from the local cache, as seen in Figure 3.4-2. Optionally, the administrator of a network may distribute peer software which has this ability disabled. The content of the network will then be distributed by the peer servers (SotonOneServer). The peers with the initial content

will be predefined by the administrator based on their location. The peer cache is explained in more detail in section 3.7.

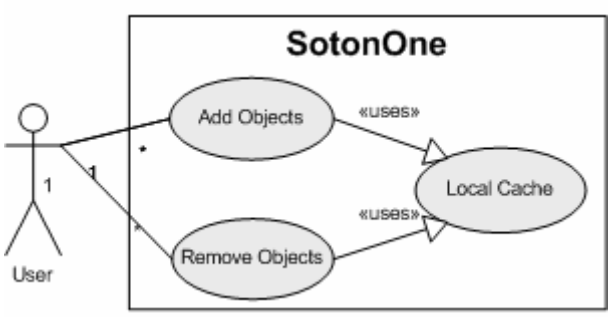

Figure 3.4-2 - Use Case for add and remove objects

The ‘peer’ actor represents other peers on the network. ‘Peer’ actors initiate remote queries over the network based on search queries made by their users or their pre-emptive cache, as seen in Figure 3.4-3. The ‘web browser’ actor is a requirement based on the non-functional requirement for a web browser to be able to access a peer as a proxy so as there won’t be any need for a graphical user interface, as seen in Figure 3.4-1. The user of the browser should be able to specify the peer IP address and Port number as a proxy server to forward HTTP requests. Even though the ‘user’ and ‘web browser’ actors effectively make the same search queries on the system, the web browser actor does not have the ability to manually change his location information or to add or remove information from the local cache, unless the user has access to a peer with a user interface.

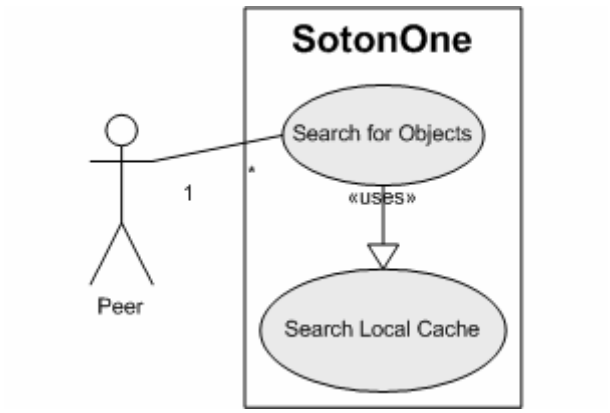

Figure 3.4-3 - Use Case for remote searches

3.5 Search Queries and Replies

Peer search queries are when an actor requests for an object. A peer may receive a search query in three ways:

- From a user via the user interface
- From a web browser via the proxy module
- From another peer via JXTA messages

Search queries from a user are received via the user interface of the peer. Search queries from a web browser are received by the proxy module within a peer, which listens on a standard IP address and Port number. Finally, search queries from another peer are received via JXTA messages. Once a search query has been received, a response is created depending on the entity which made the search query.

A search query issued by a 'user' actor generates a local search first. If an object was not found, a remote search is then issued as seen in Figure 3.4-1. A search query issued by a 'web browser' actor generates a similar search method, first the local cache is queried and then a remote search is performed. A search query issued by a 'peer' actor generates a search on the local cache only, as seen in Figure 3.4-3.

3.6 Location Awareness

Location awareness is an important research topic. As of this moment there are no marketable products based on location awareness other than GPS[16] receivers. Even though the market is ready and willing to pay large sums of money for solutions with such technology, we are yet to find a solid and reliable technology which offers location information.

The only fully working world wide location system is GPS. GPS is a satellite-based navigation and positioning system made up of 24 low-orbit satellites. The advantages of GPS include 24 hour world wide coverage and extremely high accuracy of up to 3 meters. The problem with GPS is the heavy loss of precision when tall buildings are around and the requirement to work outdoors. GPS signals work as line of sight, so they do not pass through solid objects, thus GPS receivers will not work indoors, underground or underwater. Heavy loss of precision is very easy to occur. For example, a large tree could block enough signals to make a GPS receiver inoperative.

Ideally, a PDA device or a laptop could easily receive GPS location information and via the location module translate the coordinates into a usable position which matches

[16] GPS: Global Positioning System. Satellite-based positioning system. Information provided by https://www.peterson.af.mil/GPS_Support/ (Last access date: 19-04-2003) and http://www.schriever.af.mil/gps/ (Last access date: 19-04-2003).

locations covered by the system. Unfortunately, GPS will not work indoors or even at a university campus within a busy city.

An alternative to GPS is the use of the GSM[17] mobile phone network, which spans across Europe and many other continents. The idea was presented by Ericsson at the Wrox Wireless Developers Conference 2000[18], and was described as a consumer of position data asks a positioning server the position of a list of known subscribers and calculates the relative position of the requesting device. Although GSM is the only mobile phone technology used in Europe, in the U.S. and some other countries there are many other technologies in use, like CDMA[19] and TDMA[20], Making them all work under the same location protocol would not be feasible.

Another alternative is indoors or outdoors ping transmitters. Such devices can transmit a ping signal based on their location. A mobile device receiving such a ping would be able to identify the ping and match it with a list of known locations. It is possible to either use a radio ping or Infrared transmitter. The mobile device should have the matching receiver, and in the case of Infrared it should have a line of sight to the transmitter and should be pointing to it directly.

Due to the limited time schedule a decision was made not to implement the module to receive location information from a hardware device. Thus the SotonOne peers require manual entry of location position. Even though the module to receive location information from a hardware device was never implemented, several location technologies were taken under consideration and the requirements and design of such a module have been researched for future development.

3.7 Caching and Pre-emptive caching

Caching is a temporary storage of recently accessed data, so in case that data are requested again they will be offered via the cache and not their original location. Pre-emptive cache is the power to anticipate future requests and thus request them beforehand. One such example of pre-emptive cache is Read Ahead on SCSI RAID controllers were the controller begins using read-ahead if the two most recent disk accesses occurred in sequential sectors. If all read requests are random, the algorithm reverts to normal operation (LSI 2002).

Since the transfer of information between peers is done in a request/response method, it is possible to use several widely used cache technologies. The only drawback is the need to design and implement a custom pre-emptive cache algorithm. The following is a list of technologies which were considered for the design of the cache.

[17] GSM: Global System for Mobile Communications.
[18] Wrox Wireless Developers Conference: http://www.wireless3.com (Last access date 19-4-2003).
[19] CDMA: Code Division Multiple Access: http://www.qualcomm.com (Last access date 19-4-2003).
[20] TDMA: Time Division Multiple Access: http://www.3gamericas.org (Last access date 19-4-2003).

- HTCP/0.0, Hyper Text Caching Protocol (Internet Draft)
- ICPv2, Internet Cache Protocol (RFC[21] 2186)
- Application of ICPv2 (RFC 2187)
- Squid implementation of ICP
- GDSF, Greedy-Dual Size Frequency
- LFUDA, Least Frequently Used with Dynamic Aging
- LRU, Least-Recently-Used

HTCP and ICP both play an important role to this project since they are the inspiration for a caching module. Squid[22] also played an important role to this project because it implements a wide range of caching techniques, which this project used as an example for the design of a cache module.

The peer cache has four main requirements. It should be able to add objects, remove objects, search objects based on a keyword and pre-emptively request objects from remote peers. Adding new objects to the cache should be done in two ways, either by manual user interaction from the peer user interface, as seen in Figure 3.4-2, or by the peer itself when it receives new content from remote peers as seen in Figure 3.7-1.

Removing of objects should be done in two ways as well, either by manual user interaction or by the peer when it receives new objects and the cache is full.

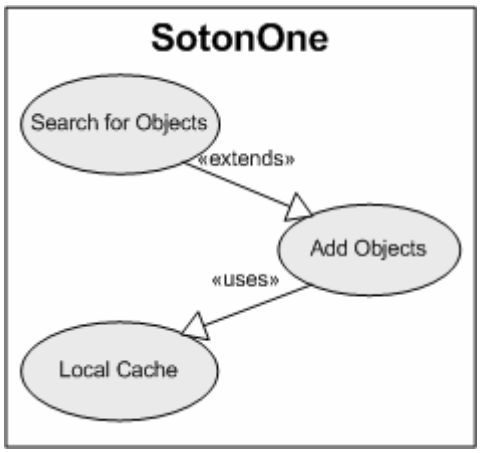

Figure 3.7-1 - Use Case for search with add

Three of the most important replacement algorithms are the GDSF, LFUDA and LRU. They are explained in the Squid documentation as:

- The heap **GDSF** policy optimizes object hit rate by keeping smaller popular objects in cache so it has a better chance of getting a hit. It achieves a lower byte hit rate than LFUDA though since it evicts larger (possibly popular) objects.

- The heap **LFUDA** policy keeps popular objects in cache regardless of their size and thus optimizes byte hit rate at the expense of hit rate since one large,

[21] RFC: Request For Comments: Internet standards, specifications and documents.
[22] Squid: Squid Web Proxy Cache. http://www.squid-cache.org/ (Last access date 19-4-2003).

popular object will prevent many smaller, slightly less popular objects from being cached.

- The **LRU** policy keeps recently referenced objects. It keeps a time stamp of the arrival or creation data of objects, and performs comparisons via those time stamps only. Fast and simple to implement, but not as optimised as the other policies.

In order to minimise complexity, a choice was made to use LRU which is the fastest and simplest to implement.

Pre-emption is achieved on top of the basic cache functions. The peer itself should initiate the pre-emptive cache when the peer changes locations. Thus when moving from one location to the other, the peer will already have a pre-emptively requested and cached objects related to the new location.

Ideally, the change of location should happen automatically via the location module supported by a location aware hardware device as explained in section 3.6. The pre-emptive caching of objects is enabled when moving from one location to another and should be fairly transparent. The user will not notice the effects until he actually requests an object based on his new location, which could be already in his cache since the pre-emptive module requested the object when the user changed locations.

Chapter 4

Project Design and Implementation

Due to the nature of software development, the design and implementation of the project are closely related with each other. Thus this chapter combines both the design and the implementation in a structured approach; at first, the theoretical aspects of the project are analyzed, then the five parts of the implementation are illustrated individually.

4.1 Risk Analysis

Requirements Risks: The risks to build such a system cover a wide range of topics. At first, there is a risk of designing the project the wrong way which will later on require a redesign and probably a re-implementation of parts of the code. The most important risk is the design of the wrong system entirely.

Technological Risks: The mobile device implementation is a risky design, which poses certain requirements on the system. Java2 Micro Edition has many restrictions, thus the implementation should be consistent with both J2SE[23] and J2ME[24]. The project also depends on the JXTA core for most of the low level communication. Future changes to the JXTA core may require code updates in the implementation and design of the project code as well.

Skill Risks: Time constraints are also important. The project requires extensive work on the location awareness and pre-emptive caching modules, and there may be not enough time to implement them properly.

Political Risks: Political risks could affect the project in many ways. At the moment the project is published under an open source license on the JXTA project site. There was a possibility that the University would not allow the project to be published.

[23] J2SE: Java2 Standard Edition
[24] J2ME: Java2 Micro Edition

4.2 Functional Design Description

If the specification of the project is “the problem”, then the design is the creative process of finding “a solution” to the problem. SotonOne project has been designed for high portability, in order to create a single code base for different peer client designs. One such design is written in J2SE’s Swing and an alternative design is written in J2ME. Due to limited resources and a limited time schedule, the J2ME interface was not implemented.

The main interface of SotonOne project has been designed into five major parts, each part controls a specific area of the project and the code is portable enough for future development to replace a module without many code changes. The five parts of the design, as seen in Figure 4.2-1, are:

- User Interface Processing
- Process and Control
- Input Processing
- Output Processing
- Local Storage

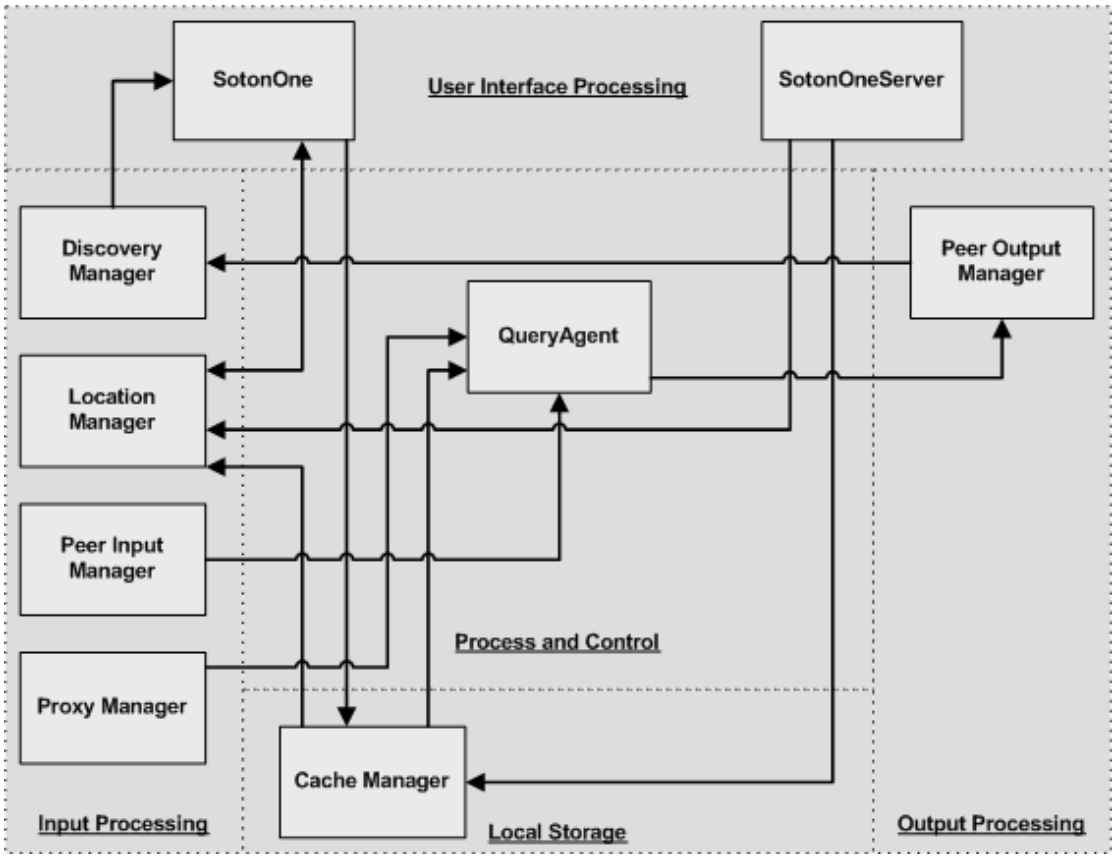

Figure 4.2-1 - Architecture Context Diagram

The following sections describe the five parts of the design in detail.

4.3 User Interface Processing

The user interface is a graphical (or a plain text) interface, which interacts with the user. The SotonOne peer has only two required functions; to be able to make search queries and to display replies in some visual manner. The SotonOneServer peer should be able to add/remove objects from the cache storage and to define a location.

The following figures, Figure 4.3-1 and Figure 4.3-2 show the UML design of the SotonOne and SotonOneServer respectively. As specified in chapter 3, SotonOneServer is a subset of SotonOne with only the minimal functionality to add/remove cached objects and to answer remote search queries. Both user interfaces are portable enough and allow the administrator to create custom user interfaces to match the environment of the system.

Figure 4.3-1 - SotonOne UML class

Figure 4.3-2 - SotonOneServer UML class

It is important to understand that the choice was made to include both a web interface on the SotonOne peer as well as an HTTP proxy. Both give the same result, but they offer different functionality. The mini-web browser provided by the Java JEditorPanel is very limited and should not be considered a fully functional web browser. However for most basic web pages it will work without problems.

Other alternatives would be to have the min-web browser only, or only the HTTP proxy. The decision to keep both is based on these facts:

- It is easy to add or remove the mini-browser.
- The HTTP proxy is more functional.
- The end-user may use other software over the network.

In a similar way, many other parts of the user interface may be discarded. The status messages and the discovery messages are of no interest to the end-user. The discovery manager and the proxy manager are described in more detail in the following sections.

4.4 Process and Control

The heart and brain of the SotonOne project is the QueryAgent. QueryAgents are individual threads which have the logic to perform query searches, locally and remotely. The name QueryAgent is based on the fact that even though the design and implementation is a Java Class, it is in fact, in many ways, an Agent.

The term Agent is controversial at best. Researchers have conflicting ideas of what an Agent is about, and how it is defined. Figure 4.4-1 displays a classification of different agents and how they relate with each other.

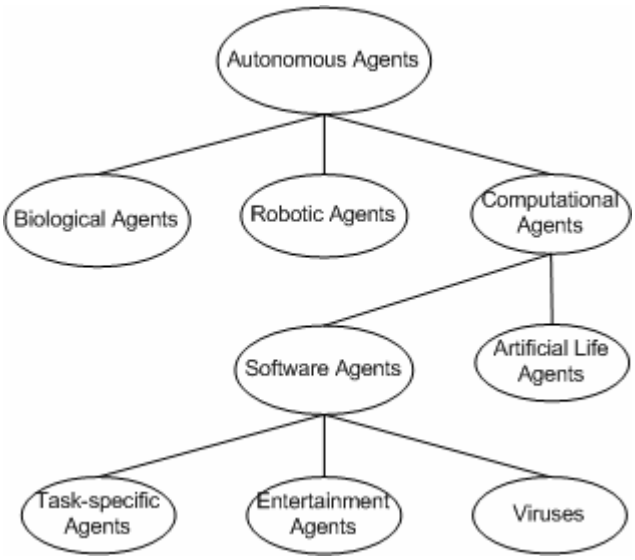

Figure 4.4-1 - Agent classification

The QueryAgent falls under the Task-specific agent category. This category is best defined by the IBM Agent definition: "Intelligent agents are software entities that carry out some set of operations on behalf of a user or another program with some

degree of independence or autonomy, and in so doing, employ some knowledge or representation of the user's goals or desires"[25].

The QueryAgent is an independent thread which spawns when a search query is received, either from the local peer or from a remote peer. Based on the rules defined in the specification from section 3.4, depending on the initiator of the search query, the QueryAgent will either perform a local search only or a local and a remote search as well. In addition, it has the ability to send newly received objects to the cache, where they may be stored by the cache manager.

The QueryAgent is independent enough to act with its own logic. Once it has been instantiated, it performs the required searches based on the search query, sends the results to the cache manager and informs the initiator of the result, as described in Figure 4.4-2.

This design allows multiple QueryAgents to be running at the same time, accessing the same local cache, communicating asynchronously and returning results without blocking other local or remote search queries.

The two most important disadvantages of this design are; the heavy resource consumption which used when instantiating new QueryAgent threads and the possibility of a remote Denial-of-Service[26] attack to the peer.

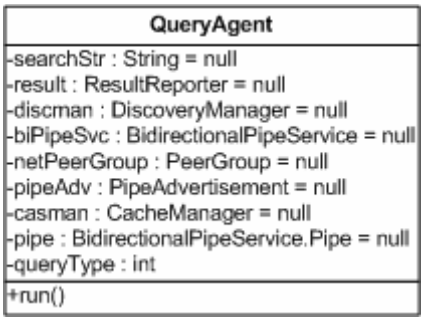

Figure 4.4-2 - QueryAgent UML class

In this case, resources are both memory and CPU[27] usage; a thread takes a lot of memory and CPU to instantiate, since threads are not only based in Java, but in the operating system itself. It is possible to minimise the heavy resource consumption of threads by implementing a thread pool. A thread pool is initialised with a finite pre-defined number of threads which are instantiated at the execution of the peer. Whenever a QueryAgent is required, an already instantiated thread will be used, thus avoiding the CPU time required to load a new thread.

DoS attacks can be minimised, but not prevented. Due to the limited and finite number of threads, a DoS attack will never instantiate too many threads, thus the system will never run out of resources. Although, once a DoS attack has instantiated

[25] IBM Agent definition: http://activist.gpl.ibm.com:81/WhitePaper/ptc2.htm
(Last access date 19-4-2003)
[26] Denial-Of-Service: DoS attacks allow an agent to render a remote system unresponsive.
[27] CPU: Central Processing Unit.

all available threads from the thread pool, the peer will stop responding to legitimate search queries. Due to the limited time schedule, a thread pool was not implemented.

4.5 Input Processing

Input processing is a very important part of the SotonOne project. A peer receives input from several different inputs, and in various formats. There are four main inputs in a peer, and they are managed by their respective manager; the Discovery Manager, the Location Manager, the Peer Input Manager and the Proxy Manager.

4.5.1 Discovery Manager

The Discovery Manager, as described in Figure 4.5-1, is a thread which initiates advertisement discoveries via the JXTA core and receives asynchronous discovery advertisements replies again from the JXTA core. Discovery advertisements are of several types, although the only one which is of interest to this project is the bidirectional pipe advertisement that each peer advertises. Advertisement definition and use are out of the scope of this document, since they are defined and explained in the JXTA v2.0 Protocol Specification[28].

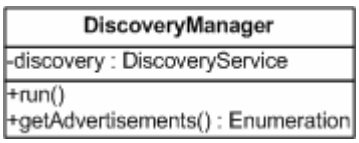

Figure 4.5-1 - DiscoveryManager UML class

Once bidirectional pipe advertisements are discovered, they are automatically stored by the JXTA core in the local advertisement cache, which is stored within a subdirectory from the peer's current directory.

These bidirectional pipe advertisements are kept in the cache until the peer makes a remote search query. Then the QueryAgent will instantiate a Peer Output Manager, which in turn requests all the available cached bidirectional pipe advertisements and uses them to contact remote peers.

4.5.2 Location Manager

The Location Manager covers all the location awareness of the SotonOne project. It is designed to achieve high flexibility, thus there are two designs for a location

[28] JXTA v2.0 Protocol Specification: http://spec.jxta.org/v1.0/docbook/JXTAProtocols.html (Last access date 19-4-2003)

definition. The Location Manager, as described in Figure 4.5-2, stores the current and previous location of the peer, and allows the other modules of the peer to query and set these locations. The locations are defined and hard-coded to the peers by the administrator of the network. It is possible in future enhancements of the project to implement dynamic location definitions, but it has been decided that for the demonstration of the basic location implementation, hard-coded locations are good enough.

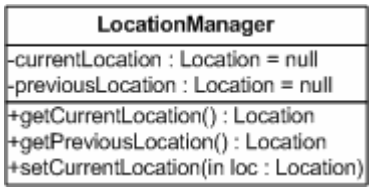

Figure 4.5-2 - LocationManager UML class

To keep up with the wide demand for location awareness and the different types of location specifications and definitions, two completely different location abstract definitions were created; a two-dimensional, Figure 4.5-3, and a three-dimensional, Figure 4.5-4, abstract definition.

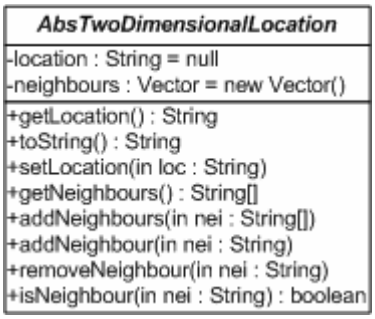

Figure 4.5-3 – AbsTwoDimensionalLocation UML class

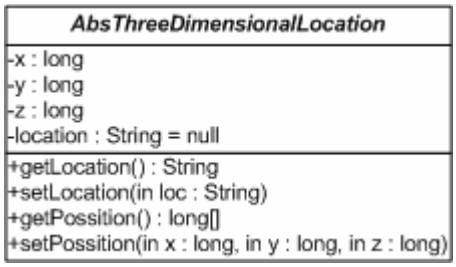

Figure 4.5-4 - AbsThreeDimensionalLocation UML class

The two dimensional location defines a two-dimensional area based on location object relationships, also called neighbours, where each location object may have one or more neighbours defined into it. As a requirement, when one location defines a neighbour, that neighbour must define the first location as its neighbour as well.

As seen in Figure 4.5-3, the two-dimensional location allows the administrator to set the location name, add and remove neighbours and to check if a location is a neighbour of another location.

The three-dimensional location defines a three-dimensional area based on an X, Y and Z Cartesian coordinate system. Each X, Y and Z position defines a single location, and all immediate positions within a difference of one point are defined as neighbours. For example, the locations 1,1,1 and 1,1,2 are neighbours, but the location 3,3,3 has higher distance than 1 on all three locations; thus it is not considered a neighbour of the other two.

The network administrator has a choice between the two location implementations, and once that choice is made, it must be kept, since a change afterwards may require a software upgrade on all the peers on the network. The administrator should then create a subclass of the chosen abstract class definition, as described in Figure 4.5-5. The implementation of the subclass is not required to have any methods in it, since it will inherit the required methods from the abstract super-class.

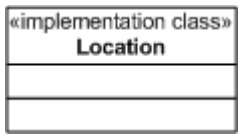

Figure 4.5-5 - Location UML class

4.5.3 Peer Input Manager

The Peer Input Manager is responsible for the incoming peer search queries, as described in Figure 4.5-6. It listens on the bidirectional pipe, which the peer advertises, and once a connection is made, it spawns a new QueryAgent thread. The Peer Input Manager has no other logic, other than to make sure it passes the right parameters to the QueryAgent.

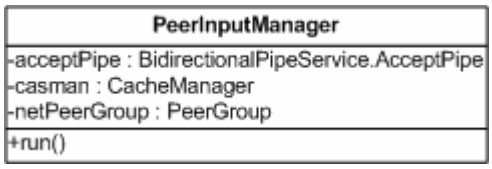

Figure 4.5-6 - PeerInputManager UML class

There are two ways to listen for connections. Either poll on the bidirectional pipe directly, or implement a message listener and receive asynchronous connection events.

Polling for incoming connections is done via the accept(int i) method of the BidirectionalPipeService.Pipe. The parameter is an integer which specifies the seconds before a timeout occurs. Since we want to keep on waiting for a connection indefinitely, the method call is within an infinite loop.

Asynchronous connections are done via the implementation of a BidirectionalPipeService.MessageListener. The method is called whenever an incoming connection is received by the JXTA code. This implementation allows the class to perform other operations as well, while waiting for incoming connections.

A choice was made to use the poll method, since the Peer Input Manager is a dedicated independent thread, used specifically to receive incoming connections, thus has no other logic and will achieve the same results as a message listener.

4.5.4 Proxy Manager

The proxy manager is a fully featured HTTP/1.1 proxy implementation based on RFC 2616, which redirects HTTP requests and replies through the SotonOne project, as described in Figure 4.5-7. The result is the maximum transparency between the end-user and the network, the user may use his web browser, or other HTTP compatible software, to retrieve objects distributed via the SotonOne peers.

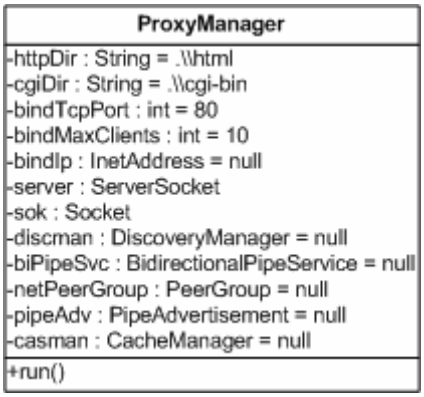

Figure 4.5-7 - ProxyManager UML class

Some of the advantages of this approach are described below:

- End-user may use his own software over the SotonOne peer network.
- The SotonOne peer does not have any requirements on the user interface.
- Minimise resource requirements for peers.
- Transparent distribution of web content is possible.

The end-user has the ability to use his own software, any HTTP compatible application which supports either the HTTP/1.0 or HTTP/1.1 protocols. This allows the user to keep on using a familiar user interface, and in some cases this method may hide the infrastructure of the network from the user.

The SotonOne peer does not have any requirements on the user interface. Either the user may use his own user interface as described above, or the administrator of the network may design a custom made user interface for specific purposes.

Minimising resource requirements for peers is achieved because a peer without a user interface has no requirements for the Java Swing API or any other visual overhead; small mobile peers are then viable.

Transparent distribution of web content is possible via the analysis of the hostname part of the URL. Web content is first loaded on the SotonOne peer network, either manually or dynamically, based on a location object which is instantiated with the hostname as a location name. The hostname is then used as a location keyword to search for web content by the peers.

4.6 Output Processing

Output processing has been implemented in the Peer Output Manager, as described in Figure 4.6-1. The Peer Output Manager is not an independent thread; instead it is an object which is instantiated whenever a QueryAgent makes a remote search query.

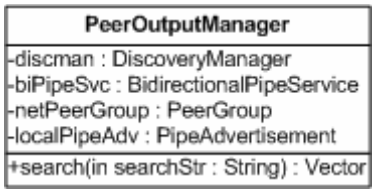

Figure 4.6-1 - PeerOutputManager UML class

The Peer Output Manager first retrieves all available bidirectional pipe advertisements, then generates a JXTA Message object and finally makes a connection and sends the Message object to all the advertised pipes.

The communication can be done in two ways; either via multiple asynchronous parallel connections, or via a single rotating connection.

Multiple asynchronous parallel connections are implemented via message events; after the pipe advertisements have been collected, a single communication is opened for each pipe on separate threads. Once each thread completes, it calls a return method on the Peer Output Manager. On a successful search query an object is returned and the Peer Output Manager closes all open threads and returns the object to the initiator of the search query.

A single rotating connection is implemented via a single loop. The loop goes through all the available pipes, one after the other, initiating a connection to each pipe. If the search query fails, the loop moves to the next pipe advertisement. Once a search query returns an object, the loop terminates and the Peer Output Manager returns the object to the initiator of the search query.

A decision was made to use single rotating connections. Although this method is slower and lowers performance dramatically, it has a single advantage which is of importance to this project; it gives the ability to demonstrate the architecture of the project implementation very clearly, and allows for easy debugging.

Peers directed towards a real-life network should use the multiple asynchronous parallel connections instead.

4.7 Local Storage

Local storage is managed by the Cache Manager as seen in Figure 4.7-1. The Cache Manager is an independent object which implements a set of methods for the storage of Java Objects.

The Cache Manager stores the objects in a Java HashMap and loads and saves the whole HashMap to disk when the peer is loaded or terminated. To simplify the implementation of the Cache Manager, the whole HashMap object is serialized and saved into a file.

This method requires that all objects stored within the HashMap must implement the serialized interface.

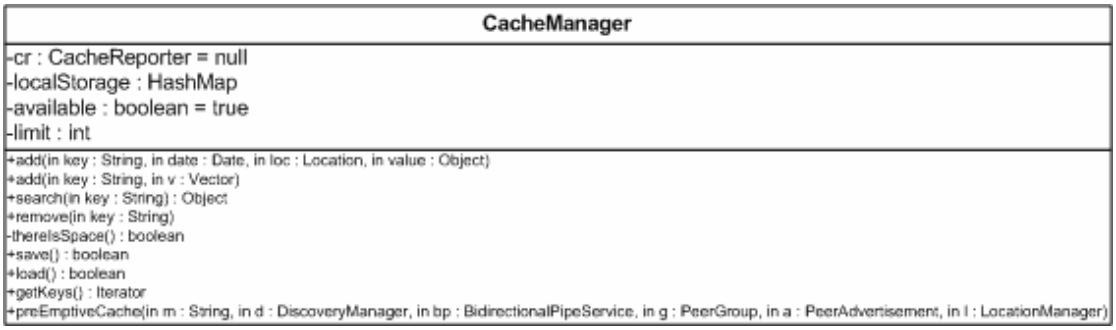

Figure 4.7-1 - CacheManager UML class

Pre-emptive caching is also implemented within the Cache Manager. Once the peer changes locations, the user interface notifies the Cache Manager, via the preEmptiveCache method. The Cache Manager is then free to implement any pre-emptive caching technique required by the administrator. This project implements a very simple pre-emptive cache; peers pre-emptively cache the 'index.html' of the new location, and all neighbours of that location.

A more complex pre-emptive cache was planned during the early stages of the project, but was never made possible due to the limited resources and time schedule.

4.8 Search and Request Objects

The communication between SotonOne peers is done via the transfer of JXTA Message objects. JXTA Message objects contain one or more JXTA MessageElement objects. There are three messages which are used by the peers, the search message, the reply message for an unsuccessful search and the reply message for a successful search.

The search message object contains only one message element as seen in Figure 4.8-1. The message element is named “Query” and contains a string of data, which represents the search query string requested by the peer.

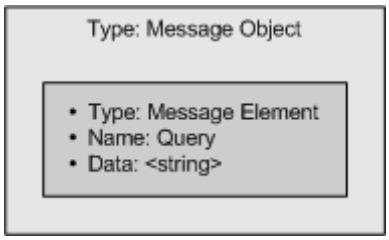

Figure 4.8-1 - Search query message

The reply message for an unsuccessful search also contains only one message element as seen in Figure 4.8-2. The message element is named “Status” and contains a single byte of the character zero. The zero represents the failure of the search query.

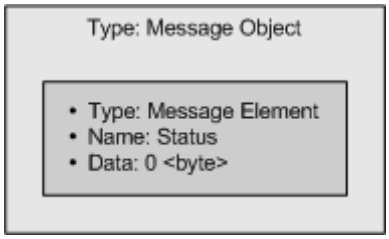

Figure 4.8-2 - Unsuccessful reply message

The reply message for a successful search contains two message elements, as seen in Figure 4.8-3. The first message element is named “Status” similar to the message reply for an unsuccessful search, although the data it contains is a single byte of the character one, which represents the success of the search query. The second message element is named “Reply” and contains a byte array of the object that matches the search query.

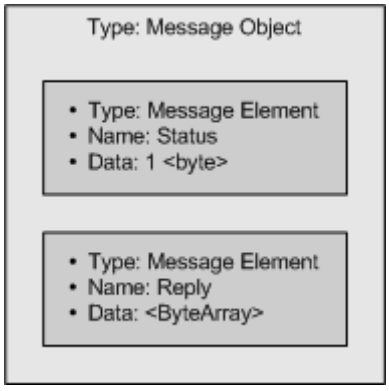

Figure 4.8-3 - Successful reply message

JXTA provides several different methods for communication; Pipe, JxtaSocket and BidirectionalPipeService. During the first design stages of the project Pipes were used and at a later time Pipes were replaced by JxtaSockets. Finally, nearing the end of the

project schedule a choice was made to use the bidirectional service instead of JxtaSockets.

The reason to move from plain Pipes to JxtaSockets was the cumbersome implementation of keeping track of multiple Pipes; due to their nature, they are single-directional and do not provide any means for a peer to reply to a request. JxtaSockets simulate the C language TCP/IP socket and provide a way for two-way communication.

With the release of the newer JXTA v2.0 core, the bidirectional service provided a two-way communication on top of the simplicity of Pipes. Therefore, it is no longer required to transmit a Message object byte-by-byte as done with JxtaSockets. Instead the bidirectional service encapsulates simple Pipes for two-way communication.

Chapter 5

Behavioural Description

This chapter describes the interaction and behaviour of the implementation. UML sequence diagrams give a graphic detail of the system and the behaviour of the different modules, components and peers. The following sections describe the two different outgoing search queries, the first does a search on the local cache first, while the second searches through remote peers directly. The third section describes the incoming search query.

5.1 Search query sequences

Search queries are made by instantiating a new thread of the QueryAgent class. The thread is given a search string and the caller waits till the thread completes the search query. Results are either be ignored, or received via the asynchronous call to reportResult() method which implements the ResultRporter interface.

Due to the nature of the QueryAgent, search queries can be performed from many different parts of the application. Thus, the CacheManager performs search queries based on the pre-emptive cache method, the ProxyManager performs search queries based on incoming HTTP web browser requests and the user interface may perform search queries based on a custom user interface model.

The SotonOne peer implementation has an embedded min-web browser, which uses the local ProxyManager to perform search queries. In a different implementation the peer may not use a web browser, instead he may use some other kind of user interface, since the interface is able to perform search queries by instantiating QueryAgent threads directly.

5.2 Outgoing search query through local cache

Outgoing search queries made via the end-user go through the local cache first. If the local cache has the requested object, it is returned to the user, otherwise the search is passed on to a PeerOutputManager object, which handles remote queries and returns a

result, which is in turn passed down to the user. The following Figure 5.2-1 illustrates the sequence of events.

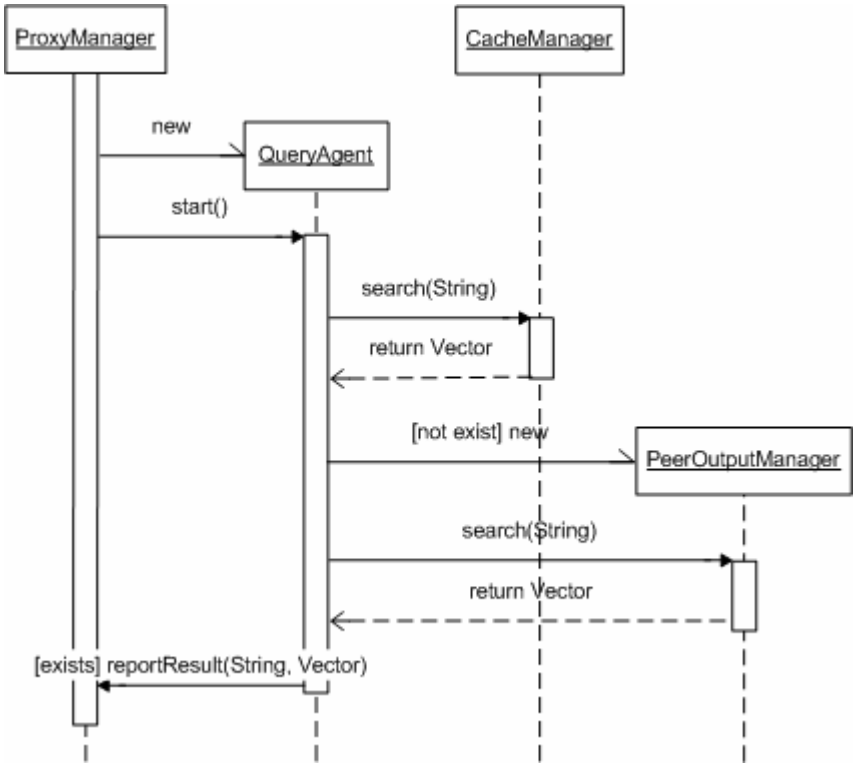

Figure 5.2-1 - Outgoing search query through local cache

When a search query is received via the ProxyManager, a QueryAgent thread is started. The QueryAgent in turn will perform a search on the local cache, and if a match isn't made then it will perform a remote search. Remote searches are initiated by instantiating a PeerOutputManager. The result is then passed back to the caller, in this case the ProxyManager, via the reportResult method. The caller is required to implement the ResultReporter interface.

5.3 Outgoing search query without local cache

Outgoing search queries made by the pre-emptive cache do not require a search via the local cache, thus they are directly used for remote searches. A PeerOutputManager is instantiated as in section 5.2, and the result is added to the local cache. The following Figure 5.3-1 illustrates the sequence of events from the CacheManager's pre-emptive cache.

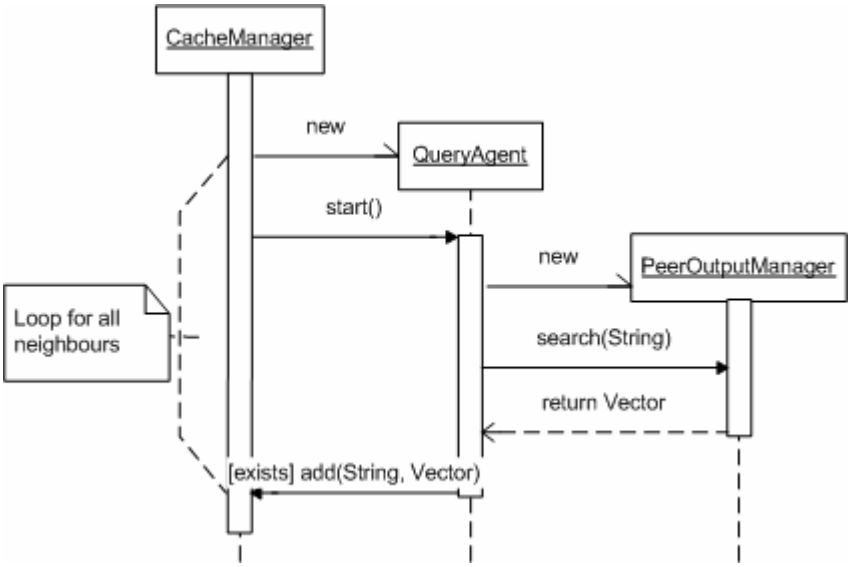

Figure 5.3-1 - Outgoing search query without local cache

Once the preEmptiveCache method is called, the CacheManager instantiates and starts a QueryAgent to search the index.html for the current location. The preEmptiveCache method will instantiate and start additional QueryAgents for each neighbour of the current location.

The search queries initiated by the CacheManager perform remote searches and any successful results are added to the local cache.

5.4 Incoming search query

Incoming search queries are received by the PeerInputManager which polls on the BidirectionalPipeService.AcceptPipe for incoming connections. Once a connection is made, the PeerInputManager instantiates and starts a QueryAgent thread, and continues to listen for new connections. The following Figure 5.4-1 illustrates the sequence of events initiated by incoming connections from the PeerInputManager.

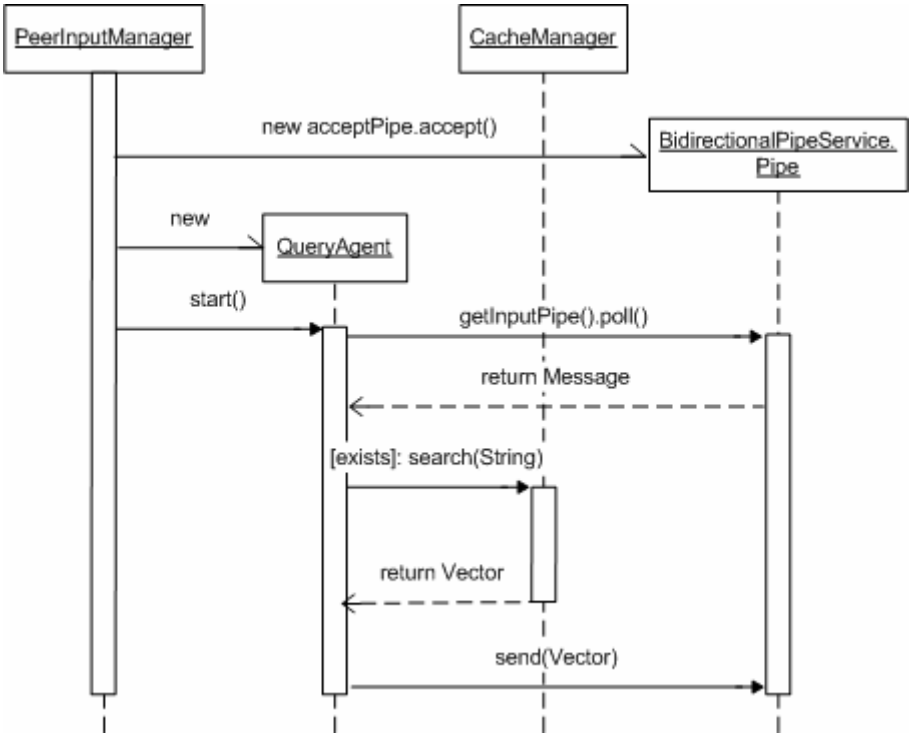

Figure 5.4-1 - Incoming search query

Once an incoming connection is made, the acceptPipe returns a reference to a bidirectional Pipe. The PeerInputManager instantiates and starts a QueryAgent thread for this connection and returns to listening for new connections. The QueryAgent thread reads the Message from the Pipe, initiates a search on the local cache. The result generates a new Message object, which is returned to the initiator of the connection, via the Pipe object.

Chapter 6

Testing

This chapter describes the analysis of a simulation, which shows the effects of network distance and peers numbers on the probability to have or not have connectivity. In addition, black-box testing demonstrates if the implementation performs as designed.

6.1 Description of the Simulation Test

The SotonOne peers implement two important features, which provide connectivity on decentralised ad-hoc networks. The first is packet propagation and the second is caching. Packet propagation is maintained by the JXTA core and allows a remote peer without direct connectivity to a SotonOneServer peer to communicate through other peers which cover the area in between. Caching stores recently accessed objects and can offer them to other peers within direct connectivity area, or via propagation.

One of the most important problems is the distribution of initial content to the peers, because when a network of peers is first setup, all caches are empty and there is no content to distribute among peers. The only available content is stored at the SotonOneServer peers. This test is a simulation of such a network, were all content is currently stored within the SotonOneServer peers.

The simulation takes a random number of SotonOne peers, from 100 to 2000, randomly distributed within a 1km by 1km area. The four SotonOneServer peers are specifically placed to simulate the distances of four buildings at the University of Southampton campus; the Electronics & Computer Science, the Social Sciences, the Management and the Mathematics buildings.

The first tests use a 25 meter coverage for SotonOne peers, and 100 meter coverage for SotonOneServer peers, and then the same tests are performed with 50 meter and 100 meter coverage.

This test aims to highlight the probability of connectivity as a function of the distance of a SotonOne peer from the closest SotonOneServer peer, which is explained by using a logistic regression model.

6.2 Logistic Regression

The logistic regression model for this problem is described by the following equation.

$$\log\left[\frac{\Pr(Connectivity)}{\Pr(No\ Connectivity)}\right] = \alpha_0 + \alpha_1 * \text{Distance}$$

Using the logistic regression model we can answer the main question of this test, what is the effect of an increase in the distance of the SotonOne peer from the closest SotonOneServer peer by 1 metre on the odds of connectivity?

In order to answer this question we need to compute $\exp(\alpha_1)$, where α_1, is the parameter of distance in the logistic model. The resulting value can be interpreted as follows: A unit increase in distance changes the odds of connectivity by:

$$(\exp(\alpha_1)\text{-}1)*100$$

We fit the logistic model using the data generated by the simulation described in section 6.1. For this application the statistical software S-plus was used.

6.3 Application of the Data on the Logistic Model

In order to simulate a real life scenario, a special connectivity simulator was designed and implemented, named SotonOneSimulation. The simulator is written in Java Swing and displays a 1km by 1km area, with the four SotonOneServer peers and a random number of SotonOne peers. The simulator displays the range of coverage by each peer as a circle. SotonOneServer peers are in red colour, SotonOne peers are in black colour when they do not have connectivity and blue when they have connectivity. Figure 6.3-1 is an example of the SotonOneSimulation application.

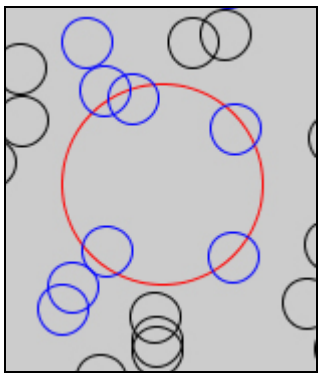

Figure 6.3-1 - SotonOneSimulation example

Peer connectivity is detected via the un-weighted shortest-path algorithm (Weiss 1999), which performs a breadth-first search.

The following section only describes, in detail, the analysis of the first dataset. The rest of the datasets are in Appendix B.

6.3.1 Fitting the logistic model in Dataset 1

This dataset was generated using the following parameters in the simulation:

Parameter	Value
Number of SotonOne peers	100
Connectivity range of SotonOne peers	25
Connectivity range of SotonOneServer peers	100

Table 6.3-1 - Simulation parameters

The following table summarises the results of the logistic regression.

	Coefficient	Std.Error	t-value
Intercept (α_0)	4.10	2.30	1.78
Distance (α_1)	-0.054	0.024	-2.23

Table 6.3-2 - Logistic regression results

We first check if distance is a significant variable in explaining the probability of connectivity. This can be done using the following hypothesis test

H0: Distance is not a significant variable i.e. $\alpha_1 = 0$
H1: Distance is a significant variable i.e. $\alpha_1 \neq 0$

If $|t-value| > t_{1-a/2,n-1}$ we reject H0, which means that distance is significant in explaining the probability of connectivity. The value $t_{1-a/2,n-1}$ is calculated from statistical tables of the $t-$distribution. Thus, with $a = 0.05$ and with a large number of samples, $t_{1-a/2,n-1} \simeq 2$.

Therefore for dataset 1, distance is a significant variable in explaining the probability of connectivity since $|-2.23| > 2$.

The logistic regression for this application is now given by the following expression

$$\log\left[\frac{\Pr(Connectivity)}{\Pr(No\ Connectivity)}\right] = 4.10 - 0.054 * Distance$$

Interpreting α_1 we conclude that a unit increase in distance changes the odds of connectivity by:

$$(\exp(-0.054)-1)*100 = (0.9474-1)*100 = -5.25\%$$

Thus, if distance increases by 1 meter, the odds of connectivity decrease by 5.25%.

6.4 Results of the Simulation Test

Based on the simulation and the analysis of the resulting data, we reach several conclusions:

- The number of SotonOne peers plays an important role in connectivity when the range of connectivity is relatively small (around 25 meters).
- The number of SotonOne peers has little or no effect in connectivity when the range of connectivity is relatively large (around 50 meters).
- At small number of SotonOne peers (<=500), the distance plays a major role in connectivity.
- At a large number of SotonOne peers (>500), the distance has little or no effect in connectivity.

These results give us clear indication how a network of peers should be setup in order to avoid connectivity problems. It is important to keep track of the number of peers based on their range of connectivity and the area the network has to cover.

Therefore an area with long range connectivity is as good as an area with short range connectivity and large number of peers. It is clear that in the construction of a network, expensive long range devices are capable of eliminating any requirements on the number of peers.

6.5 Black-box Testing

Black-box testing is based on the functional requirements and specifications of the system. Black-box testing is not a complete testing solution; rather it is a complementary approach to more detailed tests. Thus black-box testing complements the SotonOneSimulation tests, and tries to uncover errors in the implementation of the software.

The environment used to perform the tests is two systems running Linux connected via Ethernet. Each system runs four independent peers, two are SotonOne peers and the other two are SotonOneServer peers. Network connectivity is simulated by connecting and disconnecting the network from the two systems. Cached objects are then distributed within a system when there is no connectivity.

The following table shows tests based on JXTA core communication and their results.

Test	SotonOne peer	SotonOneServer peer
Load JXTA and join netPeerGroup	Success	Success
Generate new advertisements	Success	Success
Load existing advertisements	Success	Success
Create bidirectional pipe service	Success	Success
Publish advertisements	Success	Success
Remote peer discovery	Partial	Partial

Table 6.5-1 - JXTA core tests

The following table shows tests based on search queries and their results.

Test	SotonOne peer	SotonOneServer peer
Perform local search query	Success	Success
Perform remote search query	Success	
HTTP/1.0 connections	Success	
HTTP/1.1 connections	Success	
Pre-emptive cache query	Success	
Interface search query	Success	Success
Propagate search query	Partial	Partial

Table 6.5-2 - Search query tests

The following table shows tests based on the user interface and their results.

Test	SotonOne peer	SotonOneServer peer
Mini-web browser support	Success	
Local cache management	Success	Success
Status reports	Success	Success
Discovery reports	Success	

Table 6.5-3 - User interface tests

Each test was performed by recording the expected result, performing the action and recording the actual result.

There are two problems with the current implementation. Propagation of search queries is limited when using multicast packets on a single interface; in order for a multiple peers to work on a single interface with multicast packets, the interface must be connected to a remote end. Remote peer discovery may stop working at random, due to congestion in the JXTA core. Both problems are related to the JXTA core.

Chapter 7

Conclusion

This report is about the specifications, the design and the implementation of a JXTA Service, which offers decentralised object distribution in a location aware environment. Overall, the target to implement a working prototype of such a service has been successful.

The report considered all the available P2P technologies and it was decided to use JXTA as the underlying communication protocol. A RAD process model was adopted as a software engineering process throughout the project.

The specifications are complete and detailed enough to provide a clear understanding of the system, and the design is closely defined based on the specifications providing all the required information to understand the structure of the system.

The JXTA Service was implemented as specified into the SotonOne and SotonOneServer peers. A peer offers object viewing, object distribution and connectivity propagation. Peers are designed into five sections; users interface processing, input processing, process and control, output processing and local storage.

The system provides several modules to perform each task independently and in an asynchronous manner; a discovery manager identifies remote peers, a location manager handles the information of the current location its neighbours, an input manager handles remote incoming connections, an output manager handles remote search queries and finally a proxy manager which handles web browser requests.

Search queries are performed in three ways. Outgoing search query which goes through the local cache first, outgoing search query which by-passes the local cache, and remote incoming search query.

The system was tested in two ways; the first test was a theoretical test, based on the behaviour of the system and the second test was a black-box test of predefined inputs and expected outputs. The theoretical test shows that during the initial setup of the network, the administrator has to take into consideration several factors; the number of peers, the range of coverage and the network distances. Depending on the type of network a different factor may have more effect in the overall performance. Black-box testing shows if the implementation performs as specified. The results show that there are still some minor problems that need to be addressed.

7.1 The Road Ahead

Even though the implementation of the peers has been successful, there are many areas left for improvement. The initial requirement to build a portable JXTA Service has been met but the strict time schedule did not allow for an actual implementation of a peer user interface under J2ME.

The performance of search queries is not optimised for speed; instead, search queries have been optimised for debugging and testing purposes. The user interface is also based on a sample design, which may not match the requirements of a custom network implementation. Future improvements should optimise search queries for speed, and implement a several different user interfaces to match different user requirements.

Location awareness is still in its infancy; the module does not support hardware automated location information, therefore the user has to manually specify his location via the user interface. Future improvements should support at least a single hardware device for automated location information.

The cache manager lacks some of the specified LRU properties, like automated object removal. The pre-emptive cache is also very limited; it will pre-emptively cache the index.html pages only. Future improvements should add more functionality in the cache, and improve the complexity logic.

This report makes no privacy considerations. Privacy issues arise when users are able to share their own content on the network. This project is build based on the requirements of a university campus and the distribution of university pages only. If the project is used over an unmanaged network, privacy concerns arise. Future improvements should address privacy issues with location information, as well as issues with concerned with sharing of copyrighted material.

The project has been published under an open source license on Sun's JXTA web site and future work will take place at the project's web page. For more information please visit: http://www.jxta.org or http://www.jxta.org/universities/southampton.html.

References

Charette R. N. (1989). “Software Engineering Risk Analysis and Management”. McGraw-Hill/Intertext.

Chess D. et al (1994). “Mobile Agents: Are they a good idea?”. IBM Research Report, RC 19887(88465), T.J. Watson Research Center, Yorktown Heights.

Flenner R. et al (2003). “Java P2P Unleashed”. Indianapolis USA: Sams Publishing.

Fowler M. and Scott K (2000). “UML Distilled, a brief guide to the Standard Object Modeling Language, second edition”. Upper Saddle River, NJ: Addison-Wesley.

Hornby A. S. et al (1952). “The advanced learner’s dictionary of current English”. London: Oxford University Press.

Kerr J. and Hunter R. (1994). “Inside RAD”. McGraw-Hill.

LSI Logic Corporation (2002). “MegaRAID Controller Configuration”. Document DB15-000255-00, First Edition.

Pfleeger S. L (1998). “Software engineering, theory and practice” International edition. Upper Saddle River, NJ: Prentice-Hall International Inc.

Pressman S. R. (1997). “Software Engineering, a practitioner’s approach, 4th Ed”. European Adaptation: McGraw-Hill.

Shaw M. and Garlan D. (1996). “Software Architecture: Perspectives on an Emerging Discipline”. Upper Saddle River, NJ: Prentice Hall.

Wasserman A. I. (1996). “Toward a discipline of software engineering”. IEEE Software 13(6): 23-31. Los Alamitos, CA: IEEE Computer Society Press.

Weiss Allen Mark (1999). “Data Structures & Algorithm Analysis in Java”. Addison Wesley Longman, Inc.

Bibliography

Aguayo Daniel, Couto De. Douglas S.J., Wen Chiang Lin, Hu Imm Lee, Jinyang Li. "Grid: Building a Robust Ad Hoc Network. Parallel and Distributed Operating Systems Group", M.I.T. Laboratory for Computer Science.

Bakker Arno, Maarten van Steen, Tanenbaum S. Andrew. "From Remote Objects to Physically Distributed Objects". Vrije Universiteit Amsterdam. Department of Computer Science. Amsterdam, The Netherlands.

Brookshier Daniel, Govoni Darren, Navaneeth Krishnan (2002). "JXTA: Java P2P Programming". Indianapolis, Indiana: SAMS.

Cheverst Keith, Davies Nigel and Mitchell Keith. "A Reflective Study of the GUIDE system". Distributed Multimedia Research Group, Department of Computing, Lancaster University.

Coulouris George, Dollimore Jean, Kindberg Tim (2002). "Distributed Systems Concepts and Design". Pearson Education Edinburgh, Addison-Wesley.

Deitel H. M., Deitel P. J. (1999). "Java, How to program". New Jersey: Prentice Hall.

Gidari Albert (2000). "Location Privacy, Fair Location Information Practices for Mobile Commerce". Location Decisions 2000 – Application of Location Technology for the International Commercial Environment. Chicago, Illinois.

Long, S., Kooper, R., Abowd, G.D., Atkeson, C.G (1996). "Rapid Prototyping of Mobile Context-Aware Applications: The Cyberguide Case Study". Proc. 2nd ACM International Conference on Mobile Computing, Rye, New York, U.S., ACM Press.

Peterson Larry L., Davie Bruce S. (2000). "Computer Networks a Systems Approach". San Francisco: Morgan Kaufmann.

Robert A. Wilson and Frank C. Keil. "The MIT Encyclopedia of the Cognitive Sciences". A Bradford Book, The MIT Press, Cambridge, Massachusetts, London, England. Massachusetts Institute of Technology.

Sun Microsystems, Inc. Project JXTA: JavaTM Programmer's Guide.
URL: http://www.jxta.org (Last access date: 19-4-2003).

Vollset Einar (2002). "Extending an enterprise messaging system to support mobile devices". MSc SDIA Thesis.

Appendix A

SotonOne Project

User's Manual

Table of Contents

1 INTRODUCTION 46
2 REQUIREMENTS 46
3 RUNNING THE PROJECT 46
4 JXTA CONFIGURATION 46
5 SOTONONE PEER 47
6 SOTONONESERVER PEER 50

Table of Figures

Figure A.1 - SotonOne peer on RedHat Linux 9 47
Figure A.2 - Object browser 48
Figure A.3 - Cache storage 48
Figure A.4 - Status monitor.....49
Figure A.5 - Discovery monitor.....49
Figure A.6 - Web browser example.....50

1 Introduction

SotonOne Project is a distributed Peer-to-Peer service written in Java, which is based on the JXTA Java protocol libraries. SotonOne Project contains two applications, the SotonOne peer and the SotonOneServer peer. The first is the application distributed to end-users, and the second is the peer for administration purposes.

2 Requirements

The minimum system requirements to run SotonOne Project are dependant on the Java and TCP/IP environment.

- Java Run-time for the target platform (v1.4.x or newer)
- TCP/IP network connectivity

In order to use the location information correctly, the administrator has to pre-define location objects for each area physical network area. Location awareness is not a requirement and can be disabled.

3 Running the project

In order to run the peers the following command has to be issued:

java –jar SotonOne.jar (for SotonOne peers)
java –jar SotonOneServer.jar (for SotonOneServer peers)

The java command must be in the command path of the system otherwise the full path must be given, for example:

/usr/java/j2sdk1.4.2/bin/java –jar SotonOne.jar (for SotonOne peers)

The lib subdirectory must exist with the JXTA jar libraries. SotonOne Project has been implemented and compiled with the JXTA v2.0 release libraries under the Java JDK version 1.4.1.

4 JXTA Configuration

When a peer is started for the first time the JXTA configuration will be initiated. It comprises of four main pages of settings. Their configuration and meaning are out of the scope of this document, as these settings are explained in detail by the JXTA documentation.

In short, each peer listens on all available TCP/IP interfaces and on the default port 9701, unless specified otherwise. In the case of running multiple peers on the same machine, it is important to specify a different port for each peer.

Rendezvous and Relays are not used on a single class network, unless there are firewalls involved. In the same of multiple network coverage the administrator of the network must define some or all of the SotonOneServer peers as Rendezvous/Relays.

JXTA peers may use multicast packets to achieve better performance; multicast is enabled from the JXTA TCP/IP configuration page. It is important to note that multicast may not work on a single interface without network connectivity for testing purposes.

5 SotonOne Peer

SotonOne peers are split into a four parts. The first part is the mini-web browser, the second part is the cache storage, the third part is the status monitor and the fourth part is the discovery monitor. Figure A.1 shows how a SotonOne peer looks like while running under RedHat Linux 9.0 with Java 1.4.2.

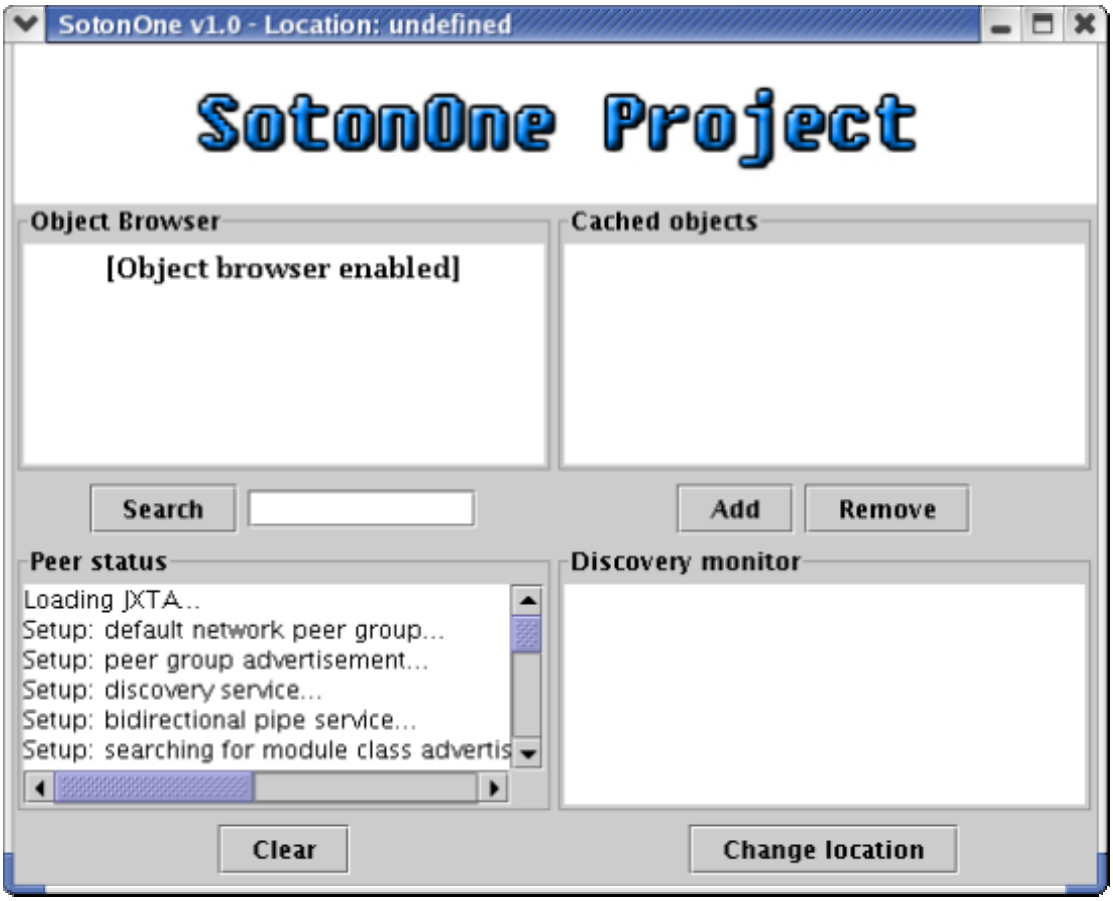

Figure A.1 - SotonOne peer on RedHat Linux 9

The mini-web browser is an example of how a peer user interface may look like. In this case it provides only a basic search facility as seen in Figure A.2. The user types an http address in the entry field and the mini-browser will connect to the proxy service of the peer to request the object. Results are then displayed on the mini-web browser. Links may also be followed through this interface like in a normal browser.

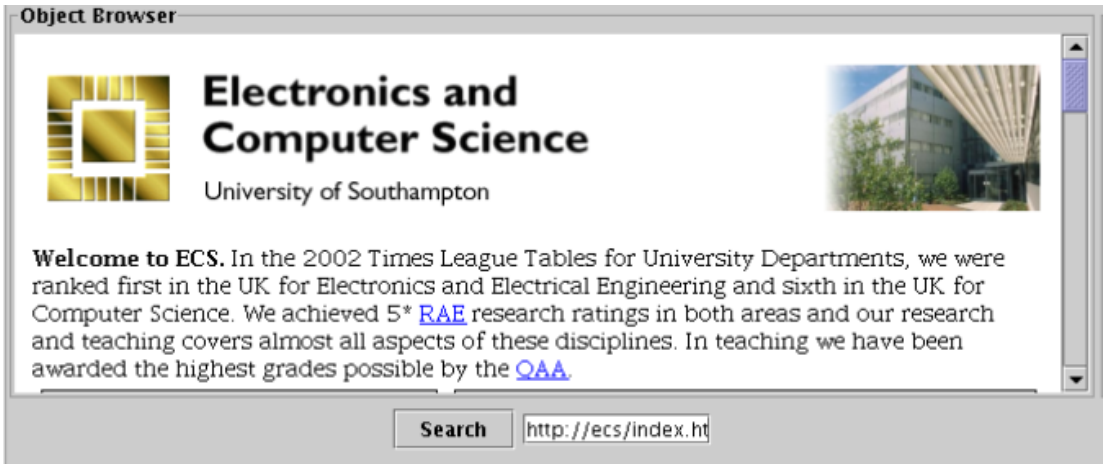

Figure A.2 - Object browser

The cache storage displays the currently stored objects, and provides the user with the option to add or remove objects, as seen in Figure A.3. Adding or removing objects is optional and may be disabled in some cases based on the administrator's discretion. Objects appear in the format //<location/<object url>, where location is a keyword from a location object.

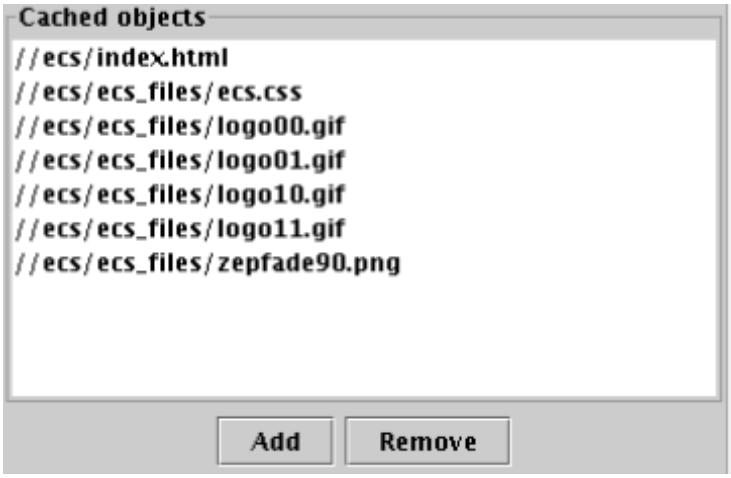

Figure A.3 - Cache storage

The status monitor displays important information about the initialisation process of the JXTA protocol and the loading of the SotonOne modules as seen in Figure A.4. Some messages are not fatal, just warnings, for example the error messages about advertisements not found means the peer is loading for the first time and hasn't generated an advertisement for it self.

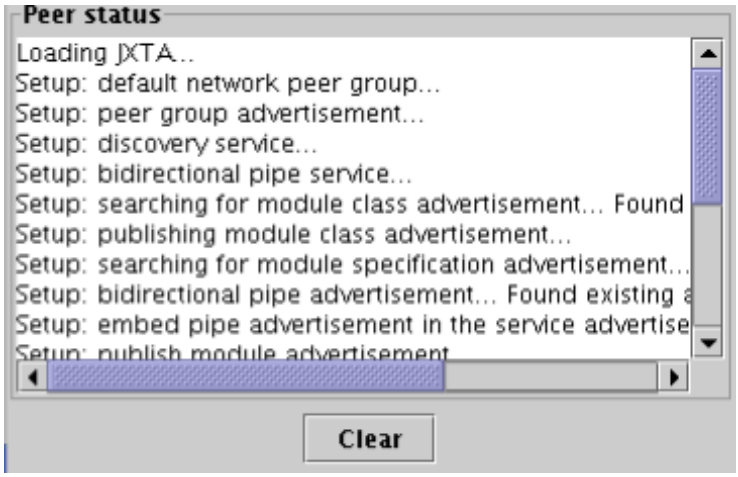

Figure A.4 - Status monitor

Finally, the discovery monitor shows detailed information about the types of advertisements the peer has discovered, as seen in Figure A.5. It is important to watch the discovery results when first deploying the system in order to detect network connectivity problems.

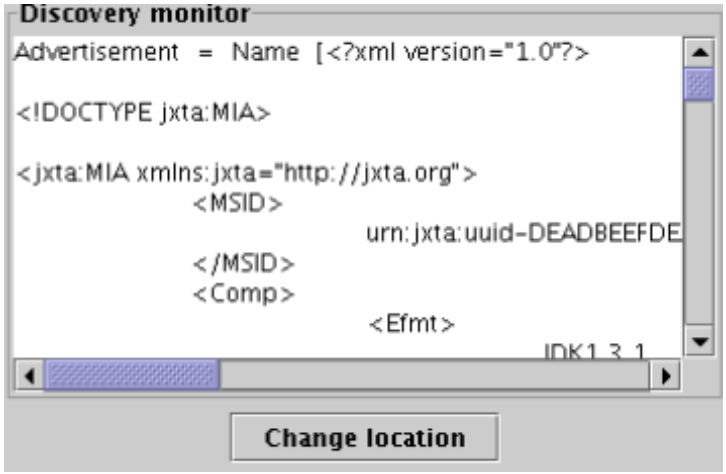

Figure A.5 - Discovery monitor

SotonOne peers also provide an HTTP proxy module. Once a peer has loaded, it binds into the first available TCP/IP port, starting from port 80. Any web browser or application with HTTP proxy support will be able to connect via HTTP/1.0 or HTTP/1.1 and retrieve data as seen in Figure A.6. Queries are not necessarily based on hostnames, but on location keywords. For example http://ecs/index.html will retrieve the index.html file from the ECS location. Location keywords can be hostnames, so http://www.ecs.soton.ac.uk/index.html will work without problems if the "www.ecs.soton.ac.uk" keyboard is properly setup.

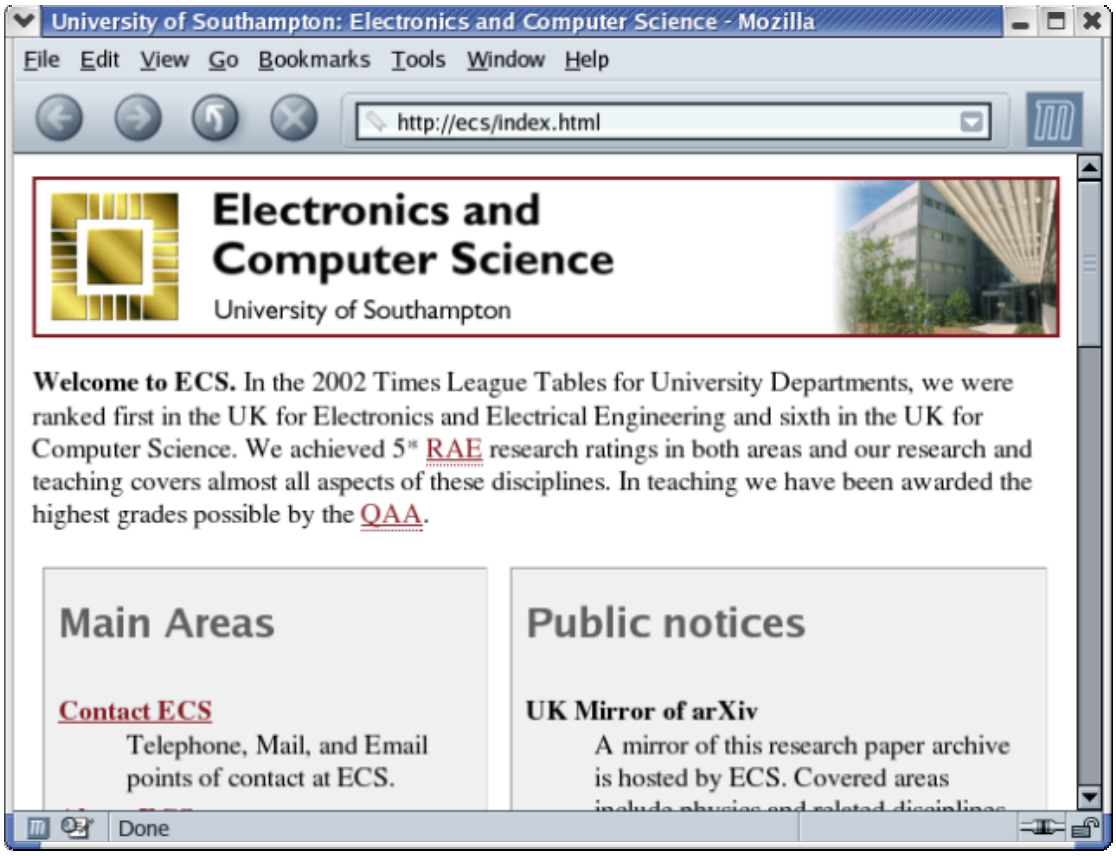

Figure A.6 - Web browser example

6 SotonOneServer Peer

SotonOneServer peers are subset peers. They provide only a minimal function, and that is to provide content based on their location only. They make no search queries and they do not provide an HTTP proxy module.

Their role is to provide content only based on their location. They have to be installed and configured with their content before the system coverage goes online, otherwise peers may not have content, or peers may be required to provide their own content.

The SotonOneServer peer window has with two parts. The first is the peer status monitor which provides the same functionality as the SotonOne peer, with the addition of the ability to define the location of the peer. The second part is the cache storage, which provides the same functionality as the SotonOne peer without any differences.

SotonOneServer peers provide no search or HTTP proxy facilities because their purpose is to provide SotonOne peers with content only.

Appendix B

Fitting the logistic model in Dataset 1

This dataset was generated using the following parameters in the simulation:

Parameter	Value
Number of SotonOne peers	100
Connectivity range of SotonOne peers	25
Connectivity range of SotonOneServer peers	100

The following table summarises the results of the logistic regression.

	Coefficient	Std.Error	t-value
Intercept (α_0)	4.10	2.30	1.78
Distance (α_1)	-0.054	0.024	-2.23

We first check if distance is a significant variable in explaining the probability of connectivity. For dataset 1, distance is a significant variable in explaining the probability of connectivity since $|-2.23| > 2$.

The logistic regression for this application is now given by the following expression

$$\log\left[\frac{\Pr(Connectivity)}{\Pr(No\ Connectivity)}\right] = 4.10 - 0.054 * Distance$$

Interpreting α_1 we conclude that a unit increase in distance changes the odds of connectivity by:

(exp(-0.054)-1)*100 = (0.9474-1)*100 = -5.25%

Thus, if distance increases by 1 meter, the odds of connectivity decrease by 5.25%.

Fitting the logistic model in Dataset 2

This dataset was generated using the following parameters in the simulation:

Parameter	Value
Number of SotonOne peers	500
Connectivity range of SotonOne peers	25
Connectivity range of SotonOneServer peers	100

The following table summarises the results of the logistic regression.

	Coefficient	Std.Error	t-value
Intercept (α_0)	1.88	0.61	3.089
Distance (α_1)	-0.033	0.0062	-5.31

We first check if distance is a significant variable in explaining the probability of connectivity. For dataset 2, distance is a significant variable in explaining the probability of connectivity since $|-5.31| > 2$.

The logistic regression for this application is now given by the following expression

$$\log\left[\frac{\Pr(Connectivity)}{\Pr(No\ Connectivity)}\right] = 1.88 - 0.033 * Distance$$

Interpreting α_1 we conclude that a unit increase in distance changes the odds of connectivity by:

(exp(-0.033)-1)*100 = -3.24%

Thus, if distance increases by 1 meter, the odds of connectivity decrease by 3.24%.

Fitting the logistic model in Dataset 3

This dataset was generated using the following parameters in the simulation:

Parameter	Value
Number of SotonOne peers	1000
Connectivity range of SotonOne peers	25
Connectivity range of SotonOneServer peers	100

The following table summarises the results of the logistic regression.

	Coefficient	Std.Error	t-value
Intercept (α_0)	1.53	0.26	5.69
Distance (α_1)	-0.017	0.0018	-9.71

We first check if distance is a significant variable in explaining the probability of connectivity. For dataset 3, distance is a significant variable in explaining the probability of connectivity since $|-9.71| > 2$.

The logistic regression for this application is now given by the following expression

$$\log\left[\frac{\Pr(Connectivity)}{\Pr(No\ Connectivity)}\right] = 1.53 - 0.017 * Distance$$

Interpreting α_1 we conclude that a unit increase in distance changes the odds of connectivity by:

$$(\exp(-0.017)-1)*100 = -1.68\%$$

Thus, if distance increases by 1 meter, the odds of connectivity decrease by 1.68%.

Fitting the logistic model in Dataset 4

This dataset was generated using the following parameters in the simulation:

Parameter	Value
Number of SotonOne peers	1500
Connectivity range of SotonOne peers	25
Connectivity range of SotonOneServer peers	100

The following table summarises the results of the logistic regression.

	Coefficient	Std.Error	t-value
Intercept (α_0)	1.52	0.13	11.25
Distance (α_1)	-0.0079	0.00045	-17.506

We first check if distance is a significant variable in explaining the probability of connectivity. For dataset 4, distance is a significant variable in explaining the probability of connectivity since $|-17.506| > 2$.

The logistic regression for this application is now given by the following expression

$$\log\left[\frac{\Pr(Connectivity)}{\Pr(No\ Connectivity)}\right] = 1.52 - 0.0079 * Distance$$

Interpreting α_1 we conclude that a unit increase in distance changes the odds of connectivity by:

$$(\exp(-0.0079)-1)*100 = -0.78\%$$

Thus, if distance increases by 1 meter, the odds of connectivity decrease by 0.78%.

Fitting the logistic model in Dataset 5

This dataset was generated using the following parameters in the simulation:

Parameter	Value
Number of SotonOne peers	2000
Connectivity range of SotonOne peers	25
Connectivity range of SotonOneServer peers	100

The following table summarises the results of the logistic regression.

	Coefficient	Std.Error	t-value
Intercept (α_0)	2.99	0.15	18.77
Distance (α_1)	-0.0024	0.0029	-8.24

We first check if distance is a significant variable in explaining the probability of connectivity. For dataset 5, distance is a significant variable in explaining the probability of connectivity since $|-8.24| > 2$.

The logistic regression for this application is now given by the following expression

$$\log\left[\frac{\Pr(Connectivity)}{\Pr(No\ Connectivity)}\right] = 2.99 - 0.0024 * Distance$$

Interpreting α_1 we conclude that a unit increase in distance changes the odds of connectivity by:

$$(\exp(-0.0024)-1)*100 = -0.23\%$$

Thus, if distance increases by 1 meter, the odds of connectivity decrease by 0.23%.

Fitting the logistic model in Dataset 6

This dataset was generated using the following parameters in the simulation:

Parameter	Value
Number of SotonOne peers	100
Connectivity range of SotonOne peers	50
Connectivity range of SotonOneServer peers	200

The following table summarises the results of the logistic regression.

	Coefficient	Std.Error	t-value

Intercept (α_0)	3.11	0.9004	3.45
Distance (α_1)	-0.02006	0.0049	-4.064

We first check if distance is a significant variable in explaining the probability of connectivity. For dataset 6, distance is a significant variable in explaining the probability of connectivity since $|-4.064| > 2$.

The logistic regression for this application is now given by the following expression

$$\log\left[\frac{\Pr(Connectivity)}{\Pr(No\ Connectivity)}\right] = 3.11 - 0.02006 * Distance$$

Interpreting α_1 we conclude that a unit increase in distance changes the odds of connectivity by:

$$(\exp(-0.02006)-1)*100 = -1.98\%$$

Thus, if distance increases by 1 meter, the odds of connectivity decrease by 1.98%.

Fitting the logistic model in Dataset 7

This dataset was generated using the following parameters in the simulation:

Parameter	Value
Number of SotonOne peers	500
Connectivity range of SotonOne peers	50
Connectivity range of SotonOneServer peers	200

The following table summarises the results of the logistic regression.

	Coefficient	Std.Error	t-value
Intercept (α_0)	2.87	0.26	10.72
Distance (α_1)	-0.0052	0.00053	-9.74

We first check if distance is a significant variable in explaining the probability of connectivity. For dataset 7, distance is a significant variable in explaining the probability of connectivity since $|-9.74| > 2$.

The logistic regression for this application is now given by the following expression

$$\log\left[\frac{\Pr(Connectivity)}{\Pr(No\ Connectivity)}\right] = 2.87 - 0.0052 * Distance$$

Interpreting α_1 we conclude that a unit increase in distance changes the odds of connectivity by:

$$(\exp(-0.0052)-1)*100 = -0.51\%$$

Thus, if distance increases by 1 meter, the odds of connectivity decrease by 0.51%.

Appendix C

The CD in the back sleeve of this report contains the following:

- SotonOne Project source code
- SotonOne pre-compiled JAR files
- SotonOneSimulation application and source code
- Dataset simulation results in CSV format
- JXTA core files
- JXTA documentation
- J2SE 1.4.2 for Linux

The contents of this CD are also provided via the author's web site:

http://www.michelinakis.gr/Dimitris