Helsinki University of Technology
Department of Electrical and Communications Engineering
Networking Laboratory

Juuso Aleksi Lehtinen

Design and Implementation of Mobile Peer-to-Peer Application

Master's Thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Technology.

January 16, 2006

Juuso Lehtinen

Supervisor: Professor Raimo Kantola

Instructor: Nicklas Beijar, Lic. Sc. (Tech.)

HELSINKI UNIVERSITY OF TECHNOLOGY

ABSTRACT OF THE MASTER'S THESIS

Author:	Juuso Aleksi Lehtinen	
Name of the Thesis:	Design and Implementation of Mobile Peer-to-Peer Application	
Date:	January 16, 2006	**Number of pages:** xiii + 102
Department:	Department of Electrical and Communications Engineering	**Professorship:** S-38
Supervisor:	Professor Raimo Kantola	
Instructor:	Nicklas Beijar, Lic. Sc. (Tech.)	

Since the beginning of the decade, a dominant part of Internet traffic has been generated by various Peer-to-Peer (P2P) applications. According to some studies, even 80% of Internet traffic in the subscriber networks is peer-to-peer. Indeed, the peer-to-peer networking has established itself as the killer application of the decade.

As the mobile phones are getting more network bandwidth, processing power, and storage capacity, the mobile phone users are starting to lust after the services that have been traditionally available only in the fixed networks; one of these services is the peer-to-peer file sharing.

The main objective of this thesis was to find out how a mobile peer-to-peer file sharing application could be implemented using the Session Initiation Protocol (SIP) as the underlying signaling protocol. The feasibility of the concept was evaluated by measuring the message sizes, signaling delays and transmission bandwidth in the network.

The thesis is divided into two parts. First, in the literature part we review the work done on the peer-to-peer networking, and discuss special requirements posed by the mobile networks. Then, in the second part we present our implementation of a mobile peer-to-peer software on Nokia Series 60 platform and measure its performance.

The key findings of the study are that the current mobile terminals and networks are ready for peer-to-peer applications. The performance of the software is more than sufficient in the user perspective; there are no noticeable delays in the software use and the download speeds are adequate for downloads of a few megabytes.

Keywords: mobile peer-to-peer, file sharing, session initiation protocol, Symbian

TEKNILLINEN KORKEAKOULU

DIPLOMITYÖN TIIVISTELMÄ

Tekijä:	Juuso Aleksi Lehtinen		
Työn nimi:	Mobiilivertaisverkkosovelluksen suunnittelu ja toteutus		
Päivämäärä:	16.1.2006	**Sivuja:**	xiii + 102
Osasto:	Sähkö- ja tietoliikennetekniikan osasto	**Professuuri:**	S-38
Työn valvoja:	Professori Raimo Kantola		
Työn ohjaaja:	Nicklas Beijar, Tekn. Lis.		

Vuosikymmenen alusta saakka suurin osa Internet liikenteestä on ollut vertaisverkkojen aiheuttamaa. Joidenkin tutkimusten mukaan jopa 80% kaikesta tilaajaverkkojen Internet liikenteestä on vertaisverkkoliikennettä. Voidaankin sanoa, että vertaisverkko-ohjelmistot ovat tämän vuosikymmenen kuumimpia Internet-sovelluksia.

Sitä mukaa kun matkapuhelimet saavat lisää verkkokapasiteettia, laskentatehoa ja tallennustilaa, alkavat matkapuhelimien käyttäjät haluta puhelimiinsa palveluita, jotka ovat olleet perinteisesti saatavilla vain kiinteässä verkossa. Yksi näistä palveluista on vertaisverkkoihin perustuva tiedostonjako.

Tämän työn päätavoite oli selvittää kuinka tiedostonjaon mahdollistava mobiilivertaisverkkosovellus voitaisiin toteuttaa käyttäen Session Initiation Protocol (SIP) protokollaa ohjelman signalointiprotokollana. Idean toimivuutta testattiin tekemällä mittauksia signalointiviestien koosta, siirtoviiveistä ja puhelinverkon siirtokapasiteetista.

Tämä työ on jaettu kahteen osaan. Kirjallisuusosassa käymme läpi yleisen vertaisverkoista tehdyn tutkimuksen ja käsittelemme mobiiliverkkojen asettamia erityisvaatimuksia. Toisessa osassa implementoimme vertaisverkkotiedostonjako-ohjelman Nokia Series 60 -alustalle ja mittaamme tämän ohjelman suorituskykyä.

Tutkimuksen päälöydökset ovat seuraavat: Nykyiset mobiiliverkot ja kännykät ovat valmiita vertaisverkko-ohjelmistoille. Luodun ohjelmiston suorituskyky on käyttäjän näkökulmasta enemmän kuin riittävä. Ohjelman käytössä ei ole huomattavia viiveitä, ja tiedostojen latausnopeudet ovat riittäviä muutaman megatavun tiedostokokoon asti.

Avainsanat: mobiilit vertaisverkot, tiedostonjako, session initiation protocol, Symbian

(8) It is more complicated than you think.
RFC 1925, The Twelve Networking Rules

Acknowledgements

This master's thesis has been done for Helsinki University of Technology Networking Laboratory in June – December 2006. The thesis has been done as part of a larger project, which concentrates on mobile peer-to-peer research.

I want to thank my supervisor for the thesis, Professor Raimo Kantola, for support during my thesis and for giving me valuable insight into peer-to-peer networking, and into its potential applications in mobile networks.

I would also like to thank Nicklas Beijar, who was the instructor for my thesis and lead for the mobile peer-to-peer project. Nicklas gave many ideas for my thesis and was invaluable help during the writing process.

My gratitude also goes to Tuomo Hyyrylä and Marcin Matuszewski who have been part of the mobile peer-to-peer project, and given good ideas for my thesis. I want to especially thank Tuomo for being my personal Symbian helpdesk during the project.

Finally, I would like to thank my family and all of my friends for supporting me during all my studies.

January 16, 2006

Juuso Lehtinen

Contents

Abbreviations

3G	Third Generation
3GPP	Third Generation Partnership Project
AOR	Address-of-Record
ARPANET	Advanced Research Projects Agency Network
AS	Application Server
B2BUA	Back-to-Back User Agent
BGCF	Breakout Gateway Control Function
BGP	Border Gateway Protocol
CAMEL	Customized Applications for Mobile network Enhanced Logic
CSCF	Call Session Control Function
DHT	Distributed Hash Table
DNS	Domain Name System
DSL	Digital Subscriber Line
DTD	Document Type Definition
EDGE	Enhanced Data Rates For Global Evolution
EPA	Event Publication Agent

ESC	Event State Compositor
FQDN	Fully Qualified Domain Name
FTP	File Transfer Protocol
GGSN	Gateway GPRS Support Node
GPRS	General Packet Radio Service
GUI	Graphical User Interface
HSS	Home Subscriber Service
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IETF	Internet Engineering Task Force
IMP	Interface Messaging Processor
IMS	IP Multimedia Subsystem
IM-SSF	IP Multimedia Service Switching Function
IP	Internet Protocol
IPv6	Internet Protocol version 6
IRC	Internet Relay Chat
I-CSCF	Interrogating-CSCF
LEN	Low Entry Networking
MGW	Media Gateway
MGCF	Media Gateway Control Function
MMS	Multimedia Messaging Service
MMUSIC	Multiparty Multimedia Session Control

MPAA	Movie Picture Association of America
MRFC	Media Resource Function Controller
MRFP	Media Resource Function Processor
MRF	Media Resource Function
MT	Mobile Terminal
MTP	Message Transfer Part
MTU	Maximum Transmission Unit
NAT	Network Address Translation
NCP	Network Control Processor
NGN	Next Generation Networking
OSA-SCS	Open Service Access – Service Capability Server
OSPF	Open Shortest Path First
P2P	Peer-to-Peer
PCM	Pulse Code Modulation
POP	Post Office Protocol
PSTN	Public Switched Telephone Network
P-CSCF	Proxy-CSCF
QoS	Quality of Service
RFC	Request for Comments
RIAA	Recording Industry Association of America
RIP	Routing Information Protocol
RTP	Real-Time Protocol

SCTP	Stream Control Transmission Protocol
SDP	Session Description Protocol
SGSN	Serving GPRS Support Node
SGW	Signaling Gateway
SIMPLE	SIP for Instant Messaging and Presence Leveraging Extensions
SIP	Session Initiation Protocol
SIPPING	Session Initiation Proposal Investigation
SLF	Subscriber Location Function
SMTP	Simple Mail Transfer Protocol
SNA	Systems Network Architecture
SP	Super-Peer
S-CSCF	Serving-CSCF
TCP	Transmission Control Protocol
TTL	Time-to-Live
TLS	Transaction Layer Security
UA	User Agent
UAC	User Agent Client
UAS	User Agent Server
UDP	User Datagram Protocol
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
VoIP	Voice Over Internet Protocol

VTAM	Virtual Telecommunication Access Method
WCDMA	Wideband Code Division Multiple Access
WWW	World Wide Web
XML	Extensible Markup Language

Chapter 1

Introduction

Usage patterns of the Internet have radically changed during the past few years. Peer-to-Peer (P2P) networking might be the next big thing after the World Wide Web (WWW) boom. After the release of the first peer-to-peer music swapping software, Napster, in 1999, a mass of Internet traffic has been created between ordinary end nodes, i.e. between millions of home and work PCs. Some studies suggest that even 80% of Internet traffic in subscriber networks is peer-to-peer traffic [5]. Measurements in the Internet backbone suggest numbers as high as 60% of all Internet traffic being peer-to-peer [24]. No longer is the WWW the dominant traffic generator in the Internet, and no longer is the traffic load concentrated around a bunch of centralized servers.

At the same time, mobile devices — especially the mobile phones — have greatly developed. The phones are getting more memory, faster processors, and larger color screens. Also the mobile phone networks are moving towards the Internet architecture. Voice and signaling traffic is being transmitted over the Transmission Control Protocol / Internet Protocol (TCP/IP) family protocols. Future Third Generation (3G) networks will employ the Session Initiation Protocol (SIP), a protocol developed originally for Voice Over Internet Protocol (VoIP) signaling in the Internet, as their signaling protocol.

Internet connectivity starts to be a standard feature in today's mobile phones, and users are accessing services, like WWW and E-mail with their cell phones regu-

larly. As mobile phones are becoming capable of playing music and video clips, there is also greater interest to get multimedia content on the mobile phones regardless of place or time. Knowing the popularity of the peer-to-peer networking among the Internet users, and the trend of ever evolving mobile platforms, it is only a matter of time before the users start asking for peer-to-peer applications for their handsets.

1.1 The Problem

As mobile phones are getting more feature rich, the phone users want to share content on their mobile phones with their friends, and to acquire multimedia content available on the other phones in the mobile network. This content is mostly self produced (e.g. pictures and videos) but maybe also professionally created (e.g. movie trailers, ring tones, and application programs). Users are posed with the question of how to share this content with their friends and other users conveniently.

In the existing mobile infrastructure, sharing information, for example pictures, is only possible by uploading the content to a centralized server, e.g. a web server, where the other people can fetch the content. Because the uploading has to be done manually, there is no guarantee that the web server has the most recently taken pictures or other recently created content available. Also, the author of the pictures has to upload the files to the centralized server even though he does not know if anyone will ever download these pictures, thus wasting his upload bandwidth. Furthermore, there exists no convenient way to search for content published by different people.

Also from the network operator's point of view there are some problems with the client-server architecture. Because all information passes through a centralized server, the operator has to spend large amounts of money and other resources to maintain this server. The server has to store a large number of files and it has to serve multiple uploads and downloads concurrently. It's also very possible that the server will be a communication bottleneck in the future — especially if the

service turns out to be more popular than predicted (the flash crowd phenomenon). The centralized server is also a single point of failure, which can bring the whole service down. Also, in some countries, the operator might be held liable for the content uploaded to the server. Also, because peer-to-peer networking seems to be a desired service, the operator is losing potential revenues by not being able to supply this service to the users.

End user problems can be summarized:

1. The content must be uploaded to be available, even though there is no knowledge if anyone will ever download it.
2. Search for the content must be done manually if the location of the content is not known in advance.

And operator problems:

1. The storage server costs money, is a single point of failure, and has hard time dealing with the flash crown phenomenon.
2. Possible revenues are lost because the operator is not able to give users what they want.

A peer-to-peer application for mobile networks seems to be a good solution for these problems. However, there is no known implementation of such product available; and because there are unique needs in the mobile networks, porting a popular peer-to-peer program to a phone would not help much. Mobile networks can not handle the traffic loads created by extensive signaling required by some existing peer-to-peer applications, and neither do these applications provide any kind of operator control.

Mobile phones are still less capable than home PCs. They also have very limited bandwidth for network connectivity. Also when services are used in the Internet, the mobile user has to pay for the traffic per byte; seldom do operators provide cheap, flat rate connections for mobile Internet users. On the other hand, for the services inside the mobile operator's network, the operator has traditionally

wanted to keep control of the service, whereas peer-to-peer networking is inherently distributed with no central point of control.

Content which is shared in mobile networks differs from the content in fixed networks. In the mobile networks content is mostly self-created, and thus many users want to limit access to this content to people they know. To enable content sharing between restricted user groups calls for authentication and grouping mechanisms to be implemented in the mobile peer-to-peer application.

1.2 Objectives and Scope

The objective of this thesis is to find out how to efficiently realize a peer-to-peer file sharing application for mobile phone networks and effectively allow users to share and acquire content which is saved in their and their friends' mobile phones. This content is both created by the users (pictures, videos) and commercial companies.

We implement a prototype of a mobile peer-to-peer application and measure its performance. Measurement data will be analyzed to draw conclusions on the feasibility of the mobile peer-to-peer content sharing.

The product has to be efficient on resource usage, both on those of the handset and the network, and use the already existing (and upcoming) network infrastructure in the mobile networks as much as possible to allow easy development and deployment. Also some kind of operator control has to be possible so that accounting and control of the shared material can be enforced.

The Session Initiation Protocol will be used as the signaling protocol for the peer-to-peer software, because SIP is a standardized protocol, which will be an integral part of the IP Multimedia Subsystem (IMS) in the future 3G networks.

The software will be used to evaluate the feasibility of the concept, considering the following points:

- Does a modern mobile phone have enough resources to run the Peer-to-Peer (P2P) application?

- Is SIP a suitable protocol for peer-to-peer signaling?
- Is the software performance satisfying in user perspective?

1.3 Structure

The structure of the thesis is following. In the second chapter we review different architectures of peer-to-peer networks and compare those to the traditional client-server architecture. We also give examples of how peer-to-peer computing can be used in different contexts. In chapter three, we study how SIP works and how it is used in upcoming mobile networks, like in the IP Multimedia Subsystem (IMS) in the Third Generation Partnership Project (3GPP) release 05 and later. Before going into the implementation details, we review requirements for mobile peer-to-peer and past work on the mobile peer-to-peer networking in chapter four, and finally, in chapter five we discuss the design and implementation of our mobile peer-to-peer application. Then, in chapter six, we present measurements on the network and software performance for our application. Last, in chapter seven, we draw conclusions on the work done and discuss which components could be further developed.

Chapter 2

Peer-to-Peer Communications

Peer-to-peer communications has been a hot topic for discussion in the field of networking during the past few years. A lot of research has been done on the subject, and many peer-to-peer applications have been developed. Many ordinary Internet users have been keen to play with these new peer-to-peer applications that provide means for distributed computing, content sharing, Internet telephony, and instant messaging.

Although seemingly a new topic, research on peer-to-peer has been done before. IBM developed a peer-to-peer resource sharing system for Systems Network Architecture (SNA) networked hosts in 1990 [50]. The system was called Low Entry Networking (LEN), and it allowed computers to share processor time with each other in the same subarea network as peers. The system architecture resembled something that is currently known as the centralized peer-to-peer architecture, as it had some centralized knowledge in the Network Control Processor (NCP) and in the Virtual Telecommunication Access Method (VTAM) nodes.

Also many original Internet protocols are peer-to-peer by design. For example Domain Name System (DNS), Usenet news, routing protocols like Open Shortest Path First (OSPF), Routing Information Protocol (RIP) and Border Gateway Protocol (BGP) are inherently peer-to-peer systems. Also usage of many other Internet protocols, like File Transfer Protocol (FTP), can be seen as peer-to-peer, when in the early days of the Internet, every node was running a server to which

any other node could connect.

As the time has passed, a lot has happened. Today, applications like *Napster*, *Kazaa*, *Bittorrent* and *Skype* are used every day by millions of end users. These applications give users access to a vast number of files, or other services, like Internet telephony. This peer-to-peer boom has affected not only the Internet as a service, but also the media industry, the network equipment suppliers and the Internet operators. The Internet is never going to be the same again.

In this chapter, we are going to examine the traditional client-server architecture and different peer-to-peer architectures and compare their properties. We will also study some proprietary peer-to-peer protocols used in the Internet. Then, we discuss various means how peer-to-peer application performance can be improved. Last, we will discuss shortly the effects of peer-to-peer applications on the Internet and on the media industry. We are limiting our focus mainly on the content sharing peer-to-peer protocols because they are the most relevant to the thesis. However, first we need a definition for what peer-to-peer is actually about. Schollmeier provides us a good one in [47]:

> A distributed network architecture may be called a Peer-to-Peer (P-to-P, P2P, ...) network, if the participants share a part of their own hardware resources (processing power, storage capacity, network link capacity, printers, ...). These shared resources are necessary to provide the service and content offered by the network (e.g. file sharing or shared workspaces for collaboration). They are accessible by other peers directly, without passing intermediary entities. The participants of such a network are thus resource (service and content) providers as well as resource (service and content) requesters (servent-concept).

The key point is that the participants of the peer-to-peer network can exchange information with each other without passing the information via some centralized, controlling entity. When many peers communicate with each other, the load inside the network is also shared evenly between the users.

2.1 Client-Server Paradigm

In the early days of the Internet or the Advanced Research Projects Agency Network (ARPANET) as the Internet was called back then, there were very few computers connected to the net. Large universities in the USA were connected to the ARPANET using Interface Messaging Processor (IMP) mini-computers, ancestors of routers. The primary use of the ARPANET was research, electronic mail, file transfers and remote logins. There were no such services as WWW and there were no such things as personal computers. Thus, every machine connected to the network communicated directly via each other. Information was fetched directly from the computer which had the needed information, not from a centralized server.

In 1990, Tim Berners-Lee wrote the *WorldWideWeb* browser (later named *Nexus*) that was able to show HyperText Markup Language (HTML) hypermedia pages [14]. This was the birth of the World Wide Web. The WWW and the concept of the HTML were based on a 50 year old idea of a global information store, *Memex*, visioned by Vannevar Bush [9].

Easy to use graphical browsers made the Internet appealing to the general public and businesses; the size of the Internet started to increase exponentially. Also, thanks to the WWW, the fundamental peer-to-peer architecture of the Internet shifted towards centralized client-server architecture (see figure 2.1). Schollmeier defines the client-server architecture as follows [47]:

> A Client/Server network is a distributed network which consists of one higher performance system, the Server, and several mostly lower performance systems, the Clients. The Server is the central registering unit as well as the only provider of content and service. A Client only requests content or the execution of services, without sharing any of its own resources.

Besides WWW many popular Internet services, like the Post Office Protocol (POP), also rely on the centralized client-server architecture. However, the protocol behind the WWW, the HyperText Transfer Protocol (HTTP), has been the one

generating the most traffic.

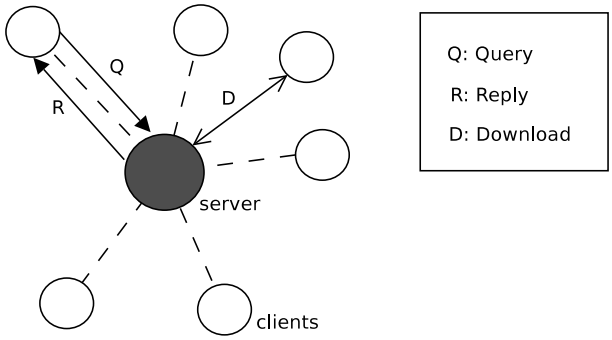

Figure 2.1: In Client-Server architecture clients communicate only with the centralized server. The server holds all the information in the network. If two clients want to exchange some information, one has to first upload the content to the server so it can be fetched from there by the other client.

2.2 Peer-to-Peer Paradigm

The peer-to-peer paradigm can be divided into structured and unstructured categories. Unstructured peer-to-peer networks can be further divided into three different architectures:

- Centralized architecture,
- Decentralized architecture,
- Hybrid architecture.

Peers communicate directly with each other in all of these architectures. Differences between the architectures are in logical network topologies and how the information is searched inside the network. Figure 2.2 shows how these architectures relate.

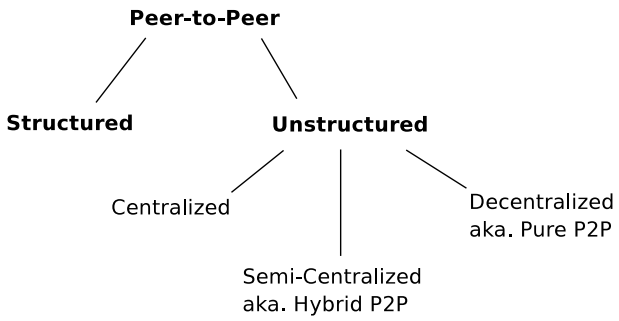

Figure 2.2: Peer-to-peer hierarchy tree

Naming of different P2P architectures is a little vague, and for example [29] divides the centralized architecture into two sub-architectures; to one where the centralized server works as the centralized database and peers communicate directly with each other, and to another where peers do not communicate with each other at all, but only with the centralized server. This latter paradigm is used e.g. in distributed computing applications where the end nodes do most of the work (e.g. SETI@home), and where the centralized server merely distributes tasks to the end nodes. We are not discussing about this latter architecture in this thesis because it is identical to the client-server paradigm in case of file sharing.

2.2.1 Centralized Architecture

The centralized peer-to-peer architecture (see figure 2.3), also often called the Napster architecture, is an architecture where a centralized server, a Super-Peer (SP), holds information about all the content available in the network. Ordinary peers register with this server and upload information about the files they have to the SP. When a node searches for files, it queries the central server, the server checks its internal database, and replies the query with information about the peers having the requested resource. The querying node then connects directly to a peer that the server just specified. All further communications is directly between these two peers; the server is only vital for searching resources. The centralized peer-to-peer architecture resembles the client-server architecture; the centralized server is

needed to hold information, but now it only holds meta information, i.e. information about information that other peers have. All resources are transferred from peer to peer, without SP involvement.

The centralized peer-to-peer architecture is most suitable for situations where some authority wants to have control on the system. For example, a telephone operator maintaining a peer-to-peer service can control the files distributed by the system if it has control on the centralized server.

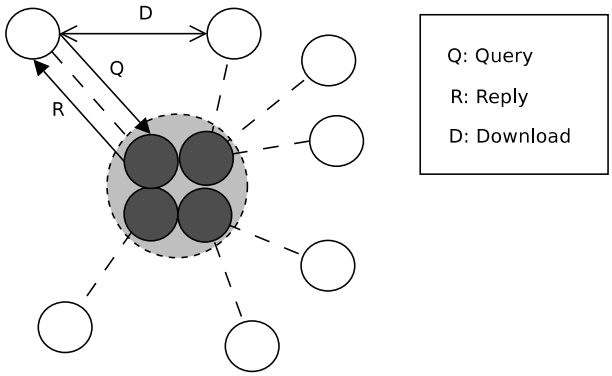

Figure 2.3: Centralized peer-to-peer architecture [46]

2.2.2 Decentralized Architecture

The decentralized architecture, also called the pure peer-to-peer architecture, has no centralized node, i.e. no SP. All nodes belonging to the peer-to-peer network are equal in this type of architecture. When a node boots up, it has to locate another member of the network by some means. Because there are no static centralized servers in the network, this is a non-trivial task. A node can for example hold a database of connected nodes when it is running. This database is saved after every session. Then the node can try to connect to several of those nodes stated in the database when starting a new session in hope that someone of the nodes listed in the database is still online in the peer-to-peer network. After finding a peer, the node contacts this peer and becomes part of the peer-to-peer network. Addresses of other peers can be learned via the first peer.

Searches in pure peer-to-peer networks are done by forwarding queries from node to node, using a flooded request algorithm (e.g. in Gnutella 0.4), which effectively broadcasts the search with limited scope; or by some more intelligent routing method (e.g. Freenet), where the query is routed towards the host most likely having the requested file. In both search types, propagation of queries is limited by the Time-to-Live (TTL) field in the queries. If some node has the requested file, it replies to the original querying peer. The reply is usually sent back to the original requester via the same path the request was routed initially. Further communications are directly between the peers or via intermediate nodes, depending whether there is a need for anonymity in the system or not.

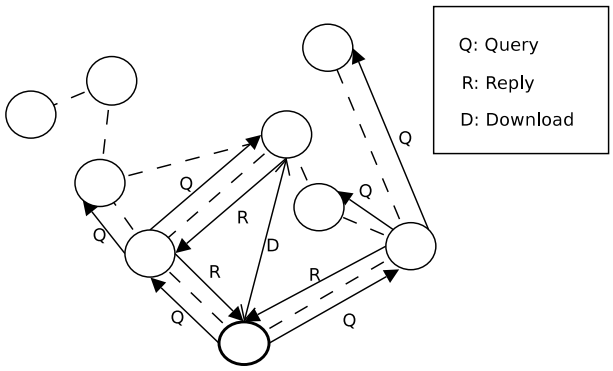

Figure 2.4: Decentralized peer-to-peer architecture

2.2.3 Hybrid Architecture

The hybrid architecture is a combination of the centralized and the pure peer-to-peer architectures. The hybrid networks have several SPs and the end users connect to these SPs as in the centralized architecture. However, the SPs themselves form a decentralized peer-to-peer network. The SPs hold information about the resources that nodes connected to them have. Every SP usually knows only about the nodes connected directly to it, not about the nodes connected to the other SPs. The hybrid architecture is used by many popular peer-to-peer applications, such as *Gnutella 0.6* and *Kazaa*. This architecture is also used by *Skype*, a massively popular peer-to-peer Internet telephony application, developed by the

Kazaa team [7].

Queries are forwarded from end nodes to an SP. The SPs query each other if any of their ordinary nodes have the searched resource. If there is a match, peers exchange all further information directly without SP involvement.

From the perspective of an ordinary client, the hybrid architecture is just like the centralized architecture. The client speaks to one super-peer and is not aware of the other super-peers, or about the pure peer-to-peer network between those super-peers. However, in some architectures the client can be connected to multiple super-peers at the same time, so if one super-peer disconnects from the network, the client can still reach the peer-to-peer network via other super-peers.

Some hybrid peer-to-peer networks allow the node to function as a normal end node or as a super-peer, depending on the conditions, like the available bandwidth and processing power. This is totally different from the centralized peer-to-peer architecture where the functionality of the super-peer and the client is always clearly separated and not built into the same application.

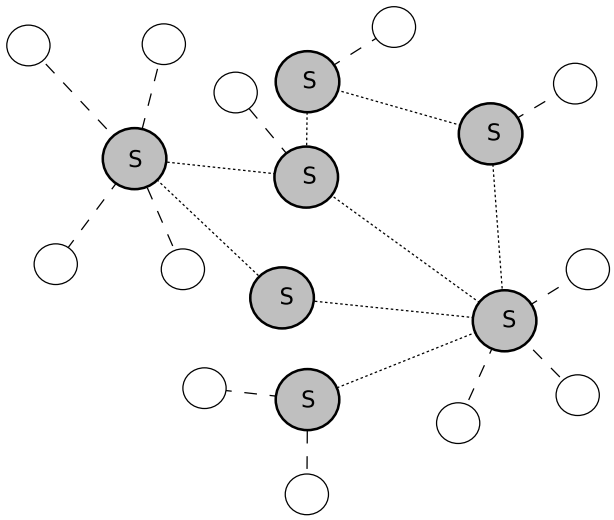

Figure 2.5: Hybrid peer-to-peer architecture

2.2.4 Structured Peer-to-Peer Architectures

In addition to the architectures described above, there are structured peer-to-peer architectures. These architectures use the Distributed Hash Table (DHT) algorithm to locate content in the network. In these architectures every file is given a unique ID or hash. When a new file is added to the network, a link to the file is stored to the node responsible for the respective part of the hash space. This link points to the actual file location.

The major drawback of using DHT networks for file sharing is that the name of the file or the hash of the file has to be known completely to fetch the file from the network — no wildcard searches are possible in DHT networks.

2.2.5 Architecture Summary

As stated above, different peer-to-peer architectures have different properties. The scalability, resiliency, search efficiency and ability to control the network depend on the architectural choice of the network.

For example, the pure peer-to-peer architecture does not scale well to a large number of nodes. However, it is the most resilient against node failures, whereas a failure of the centralized server eliminates the whole centralized peer-to-peer network. On the other hand, the centralized server may become a capacity bottleneck in the centralized architecture and thus render a large peer-to-peer network unusable.

The hybrid architecture is located between the pure and the centralized peer-to-peer architectures. It scales quite well and has good resiliency as long as the failed nodes are not functioning as super-peers. The hybrid architecture is also situated somewhere between the other two architectures when considering the search coverage in the network. When a node sends a search out, the request is forwarded to other super-peers which then forward the request onwards. The search coverage is limited by the TTL of the request message, but the coverage is a lot larger in the hybrid architecture than in the pure peer-to-peer architecture, thanks to the request only traversing super-peers.

Because both, the hybrid and the centralized peer-to-peer architectures have super-peers built into the architecture, some kind of operator control is possible in these networks via the super-peer control.

Comparison between different peer-to-peer architectures is presented in table 2.1.

Table 2.1: Architectural comparison

Architecture	**Pure**	**Hybrid**	**Centralized**
Scalability	Low	Very high	Medium
Signaling overhead in super-peer	–	High	Very high
Signaling overhead in ordinary peer	High	Low	Low
Resiliency	Very high	Medium	Low
Operator control	Low	High	Very high
Search coverage	Medium	High	Very high

When compared to the client-server architecture, all peer-to-peer architectures have one major down side: Once the peer with a certain file leaves the network, the file is no longer available — unless several peers are sharing the same file.

2.3 Popular Peer-to-Peer Protocols

Peer-to-peer networking has many uses; however, modern peer-to-peer has its roots in anti-censorship movement. Peer-to-peer networking can be used to provide far greater amount of anonymity than conventional Internet routing does. In many countries, the Internet traffic is monitored by the government or some other, potentially hostile, entity, like the employer or a jealous partner. Many of the first peer-to-peer protocols have been developed primarily with anonymous communication and information sharing in mind (e.g. Freenet).

Shortly after appearance of various anonymity-oriented peer-to-peer applications, it was noticed that peer-to-peer was a great architecture for distributing other con-

tent as well, like music and movies. Nowadays, file sharing programs are probably the most used type of peer-to-peer applications. Distributed computing is also one use of peer-to-peer systems, where a centralized server distributes computation tasks to end nodes periodically. These end nodes compute tasks assigned to them, and return results to the central node which uses the results to combine some larger answer.

Also, Internet telephony is a major user of the peer-to-peer paradigm. The SIP protocol, which is used for Internet telephony signaling, is fundamentally a peer-to-peer protocol.

As noticed earlier, many peer-to-peer protocols exist in the Internet. We will review a few of those shortly. Protocols reviewed here are not the most popular, nor the most advanced; however, they represent how different peer-to-peer paradigms are used in applications and give a good, general picture about peer-to-peer in the Internet. For last, we discuss some issues between the proprietary and the standardized peer-to-peer protocols.

2.3.1 Napster

Napster was the program which brought peer-to-peer networking to the masses. The original Napster, released in 1999, was a centralized peer-to-peer system which allowed users to share music files on their hard drives, and download files from the other users logged on to the service.

Napster does not exist in its original form anymore because of legal actions against the company. Nowadays, Napster is a subscription based, legal music service, where users can download songs by paying a monthly fee. The file-transfer paradigm used by the current Napster is no more peer-to-peer; it is traditional client-server instead. The original Napster architecture is described in [20], and compared against the Gnutella architecture.

In the original architecture, a centralized database was used to store information about which files every node had, and this database was used for file searches. When a user found an interesting file, the centralized server passed information

about the peer having that file, so that peers could directly transfer that song.

The idea behind the Napster was intelligent, the company did not have to deal with the huge traffic loads generated by the actual file transfer sessions; it only needed to handle the signaling traffic coming from and going to the centralized database.

2.3.2 Gnutella

Gnutella system is a hybrid peer-to-peer content sharing system, designed originally by Nullsoft. Unlike Napster, Gnutella allows users to share any kinds of files, not just music. The Gnutella system tries to fight legal threats by not having centralized servers which can be shut down.

The original Gnutella protocol was a pure peer-to-peer system without any central nodes [20]. Due to scalability issues, the concept of *ultrapeers* was introduced in the Gnutella version 0.6 [51]. Some nodes in the Gnutella network are assigned as ultrapeers, this assignment is based on the node resources: the network bandwidth, the firewall/Network Address Translation (NAT) status and the uptime the node is having. Many end nodes connect to these ultrapeers like ordinary nodes connected to the Napster servers. The ultrapeers form a pure peer-to-peer network among themselves; thus, they work as proxies to the Gnutella network for the less capable nodes. In the new architecture, the less capable nodes do not need to bother with large amount of signaling traffic, whereas the more capable nodes function as super-peers and are responsible for propagating search messages inside the network.

2.3.3 Freenet

Freenet is a pure peer-to-peer system designed by Ian Clarke. Freenet's main aim is to provide anonymity for its users. It allows users to publish and fetch files anonymously in the network. Freenet provides privacy via strong encryption. Content is distributed over the network, and users of the network are not able to deduce what information is passed via their computers, nor what files the Freenet system has stored on their computers. The system replicates files in the network

automatically, so a computer can store files that the user has never requested.

Instead of sending simple flooded requests as searches, Freenet builds a dynamic routing table containing mappings between the addresses of other nodes, and the content those nodes are assumed to be holding. However, files are always routed via multiple nodes, so neither the sender nor the receiver of the file knows who has the file or who is requesting it. [13]

2.3.4 Proprietary versus Standardized Protocols

Usually, the peer-to-peer file sharing protocols have not been standardized by any official standardization body. The protocols have been proprietary protocols created by the authors of the different peer-to-peer applications. Some of these protocols have become de-facto standards, i.e. there are many programs available using the same protocols, and thus able to access the same sets of files. For example the *FastTrack* protocol, originally used by the Kazaa application, has become a kind of de-facto standard, and is now used by many applications, such as *Morpheus, Grokster, and Apollon.*

It is difficult to say why there are no official peer-to-peer file sharing protocol standards. Maybe it is because of the lack of interest from the standardization bodies, or maybe there just has not been a need for a standard peer-to-peer file sharing protocol.

Nonetheless, some peer-to-peer protocols have been standardized; e.g. the Session Initiation Protocol (SIP). SIP is not designed to be a file sharing protocol but a signaling protocol. However, nothing prevents using SIP for peer-to-peer signaling.

There are some advantages in using standardized protocols for peer-to-peer file sharing. For example, having a standardized protocol helps network administrators identifying peer-to-peer traffic, and possibly imposing some restrictions on that traffic. Also sometimes, using a standardized protocol allows peer-to-peer applications to be integrated more closely with the existing network. For example, the SIP protocol is the signaling protocol for future mobile phone networks.

Building a file sharing application on top of the SIP protocol allows the application to be integrated closely to the network, and enables the operator to implement for example the charging functions easily.

2.4 Peer-to-Peer Performance Improvements

Performance of peer-to-peer systems can be increased by various means. If mere upload/download bandwidth is wanted, usage of the centralized architecture minimizes the link usage for signaling and thus maximizes the link use for actual content transfers. The same argument goes for the hybrid peer-to-peer networks, where a small number of more capable super-peers can handle larger signaling overhead.

Another way to improve performance is to cache popular content on the fast nodes, where it can be downloaded more efficiently. For example, if some slow node with a modem connection is sharing an interesting piece of information, it would be useful to cache that content to some faster node to move part of the load to a faster connection, and thus improve the overall quality of service. [26]

Multi source downloads are yet another way to improve performance perceived by a single node. They do not improve the overall network capacity, but they allow a node with a broadband Internet connectivity to use multiple sources when downloading a file, thus accelerating the download process.

2.5 Effect of P2P on the Internet

As it was mentioned in the introduction, peer-to-peer applications eat up a large portion of the Internet bandwidth. Claims between 60% – 80% of all Internet traffic being peer-to-peer have been published [5, 24].

Not just the amount of the Internet traffic has been changed because of peer-to-peer; also location of this traffic in the topology has changed. Earlier communication was just between ordinary clients and a few centralized servers; now the

traffic load is shared evenly across millions of end nodes. This poses large requirements on the network infrastructure that was originally engineered with the client-server paradigm in mind. Popular asymmetric cable modem and Digital Subscriber Line (DSL) connections are not the best means to connect to peer-to-peer networks. These connections usually have high downstream bandwidths, but their upstream capacity is limited. This asymmetric design is not most suitable for peer-to-peer, because the total flow out of the network (download bandwidth) cannot exceed the total flow into the network (upload bandwidth). Thus, asymmetric connections virtually restrict the download bandwidth to the average upload bandwidth, considering that the same amount of time is used uploading and downloading information.

This asymmetry would not be a problem if the peer-to-peer users spent only a fraction of the total time they are running the peer-to-peer applications downloading content. Unfortunately, most users run the peer-to-peer applications only when they are downloading content from the network and shut down these applications as soon as the downloads have finished. These users — often called freeriders — effectively share content only when they are downloading something from the network. [37]

2.6 Economical and Legal Issues

The music and movie industry has had issues with the peer-to-peer since Napster. Legal actions against Napster were the reason for the original service shutting down. Also many other peer-to-peer developers and users have been harassed by the media industry.

In the US, Recording Industry Association of America (RIAA) and Movie Picture Association of America (MPAA) have been strongly fighting against peer-to-peer networks, and especially distribution of pirated material in these networks [3]. There have been legislative changes which allow suing of peer-to-peer developers based on copyrighted material distributed in the peer-to-peer networks accessible through their programs. Legal issues faced by the peer-to-peer networks and users

are analyzed in [32].

There still does not seem to be any studies available, except those done by RIAA and MPAA, which show if the peer-to-peer networking has actually had any impact on the music or movie industry. Some studies indeed point that the content creation industry could benefit from peer-to-peer by adjusting their business models and using the peer-to-peer networks as a content distribution channel [27].

Chapter 3

Session Initiation Protocol

The Session Initiation Protocol (SIP) [45] is a general Internet signaling protocol for creating, modifying and terminating multimedia sessions between two or more participants. SIP can be used as a signaling protocol for Internet Protocol (IP) telephony, instant messaging, multimedia conferences, and similar applications. [49]

SIP was drafted by the Internet Engineering Task Force (IETF) Multiparty Multimedia Session Control (MMUSIC) working group in 1997 as the result of merging two different signaling protocol proposals: the Session Invitation Protocol (SIP) by Mark Handley and Eve Schooler, and the Simple Conference Invitation Protocol (SCIP) by Henning Schulzrinne. In 1999 the SIP working group was established, and later Session Initiation Proposal Investigation (SIPPING) and SIP for Instant Messaging and Presence Leveraging Extensions (SIMPLE) working groups were set up for investigating further applications of SIP and defining instant messaging extensions for it. [23]

SIP is an end-to-end protocol; SIP messages are routed via SIP proxies from the originator to the target user. SIP entities have a peer-to-peer relationship between each other, thus any entity can send the initial request, and any entity is capable receiving requests. During a single transaction, the entities are in a client-server relationship, where the request sender functions as the client, and the party who sends the reply, as the server.

The SIP will be the signaling protocol for the IP Multimedia Subsystem (IMS) in

the 3G networks. IMS will be part of the 3G networks from the 3GPP Release 5 onwards.

In this chapter we will review the SIP protocol and some extensions to it. We will start the chapter by reviewing how the protocol functions, and what kind of signaling messages are included in the protocol. Then, We will also discuss how SIP is used in the future Third Generation (3G) IP Multimedia Subsystem (IMS) mobile networks. A reader familiar with the SIP protocol should skip to section 3.2.

3.1 SIP in General

SIP is a text based signaling protocol; it is based on the HyperText Transfer Protocol (HTTP) and the Simple Mail Transfer Protocol (SMTP). SIP uses the same request/response transaction model and status codes as HTTP [17]. From the SMTP [25] protocol SIP takes the text encoding rules and header style, it uses many same header fields with the SMTP, e.g. *To*, *From* and *Subject* fields.

SIP is specified in RFC 3261 [45]. This Request for Comments (RFC) document specifies the protocol, and necessary components of the SIP signaling framework. The SIP architecture provides means for resource location and location independent routing of signaling messages. SIP only provides signaling for negotiating session characteristics; the protocol provides no means to transfer actual communication data between session participants; thus other protocols have to be used in addition to SIP to create any meaningful services. Figure 3.1 illustrates how SIP fits into the protocol stack.

RFC 3261 specifies five aspects of multimedia session establishing, and terminating that SIP provides:

1. User location - where to route signaling?
2. User availability - is the requested user available?
3. User capabilities - what are the media capabilities of the callee?

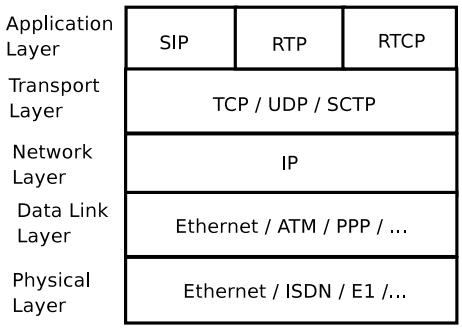

Figure 3.1: SIP is an application layer signaling protocol. It's used in conjunction with other application layer protocols to create meaningful services.

4. Session setup - establishment of the session parameters.

5. Session management - transferring, modifying, terminating the session and invoking services.

Compared to another major internet signaling protocol, H.323 [22], SIP is a lightweight protocol. H.323 is an umbrella recommendation for multimedia communications, which includes many protocols, from signaling to actual voice transmission, whereas SIP only defines the signaling part, and leaves other functionality to separate protocols; e.g. telephony applications use the Real-Time Protocol (RTP) to transmit the audio. A comparison between SIP and H.323 can be found in [48].

SIP is not tied to one transfer protocol. SIP is able to run on the Transmission Control Protocol (TCP), the User Datagram Protocol (UDP), the Transaction Layer Security (TLS) as defined in RFC 3261 [45], and the Stream Control Transmission Protocol (SCTP) as defined in an Internet draft [44].

3.1.1 SIP Infrastructure and Terminology

Before going into details of how SIP functions, it is good to sit back and define some key concepts.

Address-of-Record (AOR) is a SIP or SIPS Uniform Resource Identifier (URI) which points to a domain with a location service. The AOR is the public address

of the user, which is usually communicated in the Request URI and To: header field when contacting the user; e.g. alice@detroit.com.

Dialog is a peer-to-peer relationship between two UAs defined by the To tag, From tag, and Call-ID header field.

Session is a collection of participants, and media streams between them. Session consists of one or more SIP dialogs.

SIP Transaction consists of a single request and one or more replies to that request.

Home Domain is the domain where the user's AOR is allocated; e.g. the home domain for a user having AOR charly@detroit.com would be detroit.com. This is also the domain where the registrar for the user resides. Also the SIP messages to the user are initially routed via this domain.

User Agent Client (UAC) is the logical entity that creates the SIP request.

User Agent Server (UAS) is the logical entity that receives the SIP requests sent by the UAC and generates replies to them.

User Agent (UA) is the logical entity that can act as a UAC and UAS.

Back-to-Back User Agent (B2BUA) is the logical entity that works as a UAC and UAS at the same time, so it can receive SIP requests, modify them, and send them out as new requests. The B2BUA can be used for example to implement an anonymizer service, which removes all tags from the SIP request which could be used to identify the original message sender.

Request is a SIP message that the UAC sends to the UAS. The first line in the request specifies the SIP method.

Reply is a SIP message that the UAS sends to the UAC as a reply to the request sent by the UAC. The first line in the reply specifies the status code.

Method is the SIP request type, specified in the first line of the SIP request message, e.g. *INVITE*, *OPTIONS*, *BYE*, etc. The method specifies what functionality the request is supposed to invoke on a server.

Contact URI is an IP-address or Fully Qualified Domain Name (FQDN) where

the user for some AOR can be reached at the moment.

Proxy is an element which forwards the SIP requests sent to it. The proxy does not issue requests, nor does it have media capabilities or parse message bodies. The proxy server can be seen as application layer router. The proxy uses information from the DNS and location service to route messages. A proxy can be stateless or stateful, depending on whether it keeps state information about SIP dialogs passing through it.

Registrar Server is a server that accepts REGISTER requests and saves the information contained in these messages to the location service of the domain; i.e. AOR to Contact URI mappings.

Location Service is a logical component that stores AOR to Contact URI mappings for users of the domain. The registrar inserts these mappings to the location service. The proxies use the information in the location service to route messages to users.

SIP Gateway is an entity which can convert the SIP signaling to another signaling protocol format. A gateway can, for example, be used to establish calls between SIP and H.323 enabled devices, SIP gateways can be also placed between traditional Public Switched Telephone Network (PSTN) and VoIP enabled networks.

General SIP infrastructure is pictured in figure 3.2.

3.1.2 Resource Location

SIP infrastructure provides resource location of location-independent names, i.e. for AORs. The main components needed for the resource location are the registration and the message routing functionalities.

3.1.2.1 Registration

When a SIP enabled UA starts up, it has to register to the Registrar of its home domain. The registration is performed by sending a REGISTER message (see Section 3.1.4) to the registrar. The REGISTER message includes user's current

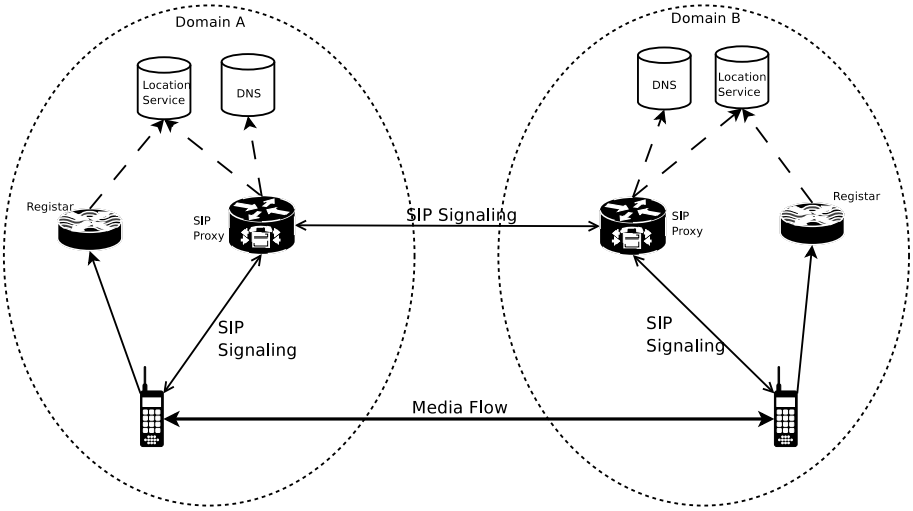

Figure 3.2: The SIP Trapezoid. The figure shows signaling and media paths for SIP session between users in different domains. It also shows the key components in the SIP infrastructure.

address in the Contact field, and the user's AOR in the To field and METHOD line. The registrar will update UA's current location to the location service, i.e. create a mapping between the AOR and the terminal address the user is using, and send 200 OK message back to the UAC to inform that the registration succeeded.

Example registration is shown in figure 3.3.

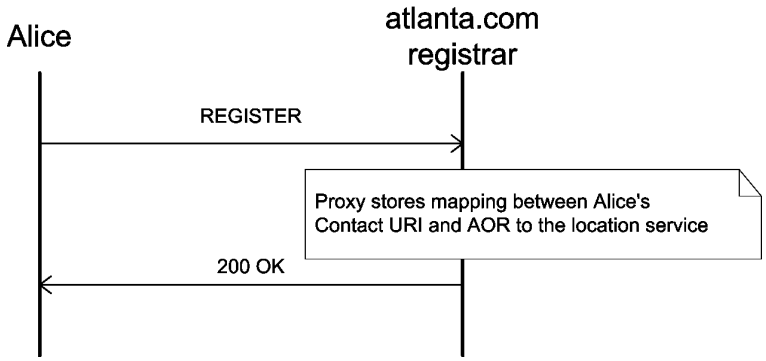

Figure 3.3: User updates her information to location service with REGISTER message

3.1.2.2 Message Routing

When a user wants to contact another user, he usually does not know the current location of the another party. This means that the user can not contact him directly by using his Fully Qualified Domain Name (FQDN) or IP-address. However, the SIP infrastructure provides message routing that enables the originating user to send the SIP message to the known AOR of another user. User A sends the message first to the preconfigured outbound SIP proxy in his home domain, or alternatively to a inbound SIP proxy in foreign user's domain, whose address can be attained using DNS [43].

If the message was forwarded first to the outbound proxy in the home domain, this proxy will resolve the correct inbound proxy in the foreign domain, and send the message there. The proxy in the foreign domain will contact the location service of the domain to get information about the current location of user B. The proxy then uses the location information to route the message to the final recipient. The final recipient will send the reply via the same proxies the request came from. In addition to message routing, these proxies may be also used to implement charging and application layer firewall functions.

After the communicating partners have located each other via proxies, they may start communicating the SIP messages directly between each other without proxies if the intermediate proxies have not required to stay on the signaling path by inserting a Record-Route field in the initial SIP messages.

Figure 3.4 shows an example of routing INVITE message between two users in different domains.

3.1.3 SIP Message Format

SIP is a text based protocol and it does not have a static message format. Instead the SIP messages consist of separate lines of text. A line may be a Request-Line, a Status-Line, a Message Header-Line or it can be part of the Message-Body.

The Message-Body can be used to convey arbitrary information. Usually information in the body has no meaning for the signaling, but it is processed by the

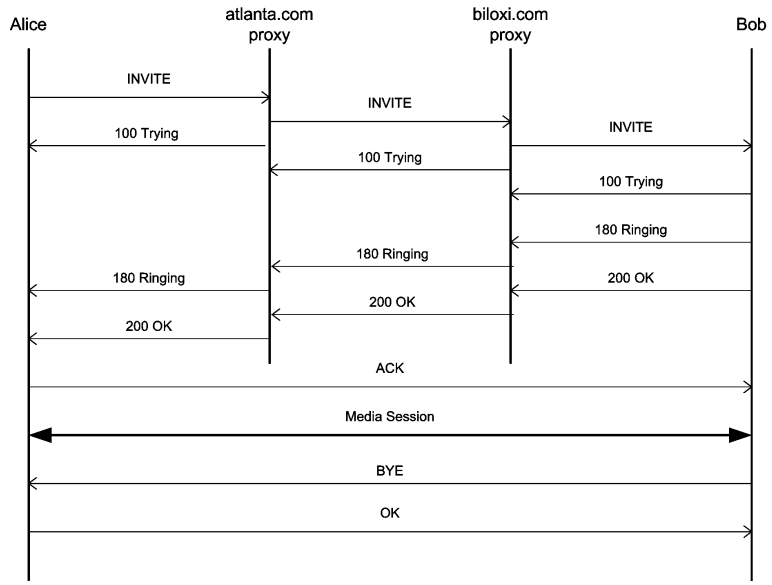

Figure 3.4: The figure shows how a session is created between two users in different domains. Alice sends INVITE to her default proxy, which forwards it to Bob's domain. Bob's proxy will further forward the message to Bob. [45]

application using the SIP protocol; e.g. two clients can negotiate their media options using the Session Description Protocol (SDP) carried in the body of SIP messages when establishing a new session. This message-body can also be used for example to convey peer-to-peer application specific information if the application uses SIP as its signaling protocol.

The message format for requests and replies has minor differences. Shown below are the high level descriptions of the message formats.

Format for SIP Request

```
Request-Line (Method, Request-URI and SIP-Version)
Message Headers
Optional Message-Body
```

A valid SIP request must contain at least the Request-Line; and the To, From, CSeq, Call-ID, Max-Forwards and Via headers, whereas a valid SIP reply must

contain at least the Status-Line and the same headers as requests.

Format for SIP Reply

```
Status-Line (SIP-Version, Status-Code and Reason-Phrase)
Message Headers
Optional Message-Body
```

3.1.4 SIP Requests

The basic SIP RFC [45] defines the REGISTER, INVITE, BYE, CANCEL and OPTIONS methods for establishing, modifying and terminating sessions. There are additional methods specified in later RFCs, like INFO [15], MESSAGE [11], PUBLISH [31], UPDATE [39] SUBSCRIBE and NOTIFY [40], REFER [53] and PRACK [42].

Methods are in request/response style, so there is one final reply per one request. There can also be several provisional replies before the final one. The INVITE request is an exception, it is a three-way message, meaning there is a request-reply-request (INVITE – 200 OK – ACK) pattern.

Proxies do not have to understand the method to pass messages forward, they only check where the message is heading and treat unknown requests like the OPTIONS method. If the receiving UA is not aware of the method it receives, it has to discard the message and reply with an error code.

REGISTER The REGISTER request is needed to push user's current Contact URI (an FQDN or an IP-address) to the location service in its home domain. The location service creates a mapping between this URI and the user's AOR in its database. Informing location service about the user's current location enables user mobility, because proxies in the home domain can now route messages sent to the user to the correct host. The REGISTER request can be also used to refresh, query and remove mappings in the location service for the AOR. The REGISTER request is sent from the UAC to the Registrar.

INVITE The INVITE request is used to establish sessions between UAs. The INVITE request is different from the other requests because it is a three-way request; whereas there is usually one request and one reply, for INVITE there will be the initial INVITE request, zero or more provisional 1xx-responses, and a final reply. After the final reply, the UAC will send an ACK request to the UAS to confirm reception of the final reply. There is no reply for ACK. This different behavior is because the INVITE requests can take a long time to complete, since they need user confirmation. Often there are provisional responses for the INVITE requests. The INVITE request often contains a SDP [18] payload, that describes what protocols and codecs are used on the media path. An offer/answer model is used when negotiating session parameters using SDP [41]. INVITE requests can be sent inside a session to update session or media parameters, these mid-session requests are called re-INVITEs.

BYE The BYE request is used to terminate session established with the INVITE request. When a session participant sends a BYE request to the other end, the request is replied with 200 OK, and the sessions is closed.

CANCEL The CANCEL request is used to cancel pending requests. Request can be only cancelled if it has not been fully processed yet, completed requests can not be cancelled. This means that practically only INVITE requests can be cancelled, because their processing can take a longer time, whereas other messages are usually processed immediately. INVITE requests can be cancelled when the initial INVITE is sent, but there has been no final reply yet.

OPTIONS The OPTIONS request is different from other methods defined in RFC 3261 [45]. It is a request that does not establish, modify or terminate sessions, and it can be sent even if there exists no session between the endpoints, i.e. outside a session. The request is used to query the UA or the proxy about the SIP capabilities it supports and to discover its current availability. Response to a OPTIONS request includes capabilities the UA or server supports.

INFO The INFO request, defined in RFC 2976 [15], is used to convey call control information during a existing session. This information does not change the state of the SIP session parameters, or media characteristics of the call, it merely allows mid-session signaling for the application in both ends.

MESSAGE RFC 3428 [11] specifies the MESSAGE method, which can be used to transfer user readable messages between terminals over the SIP signaling path. The MESSAGE requests can be used for instant messaging and services alike. MESSAGE requests do not initiate SIP dialogs, they can be sent within an existing dialog, or outside any dialog. The request is different from the other SIP requests because it carries actual media rather than signaling. It is more convenient to use the signaling path to send short messages, than to create a short-lived session for every message transaction.

SUBSCRIBE and NOTIFY The SUBSCRIBE and NOTIFY requests are defined in RFC 3680 [38]. These methods are used for subscribing to and notifying of events related to the SIP system. The SUBSCRIBE request is sent to an element that can compose NOTIFY requests whenever some predefined event happens. Subscribing to the UA registration status is an example where this framework can be used [40]. An entity can SUBSCRIBE to registration status of some other subject, and every time the registration status for this subject changes, the service will notify the subscribing entity with a NOTIFY request. This scheme is more efficient than polling the registration status every now and then.

PUBLISH RFC 3903 [31] describes the PUBLISH method. This method can be used for publication of event state information from UAC's Event Publication Agent (EPA) to an Event State Compositor (ESC). This method is largely intervened with the SUBSCRIBE and NOTIFY methods, and is used for example to publish presence information for presence service.

UPDATE The UPDATE request is defined in RFC 3311 [39]. The UPDATE request is used to modify the state of a pending session. If a session is in in-

termediate state, i.e. the initial INVITE is sent but no final reply is received, a re-INVITE can not be sent to modify session parameters, however an UPDATE request can be used to modify session state when the session is being established; e.g. the UPDATE request can be used to put a media stream on hold.

PRACK RFC 3262 [42] defines the PRACK method, a method used to provide reliable provisional responses. Usually provisional 1xx replies to INVITE requests are not guaranteed to be received reliably because there are no replies for them, whereas the final response is always responded with ACK. The PRACK request is sent on reception of a provisional message if both UAs support this option. Because the PRACK method is handled as a usual request, there will be a 2xx-reply for every PRACK.

REFER The REFER request is specified in RFC 3515 [53]. The REFER request is used to ask another UA to access a resource specified by the Uniform Resource Identifier (URI) or Uniform Resource Locator (URL) in the REFER request. It can be used, for example, for call forwarding or referring the user to a webpage. The request can be sent within or without an existing dialog.

3.1.5 SIP Responses

The SIP response messages or replies are identified by the reply codes. The reply codes are divided into six response classes, shown in table 3.1. Because of a large number of individual response codes, and their irrelevance to this thesis, only a selected few are reviewed.

100 Trying The `100 Trying` is a special type of response which is always sent hop-by-hop. It is never forwarded by proxies, although UAs as well proxies can generate these responses.

200 OK The `200 OK` response is sent to accept a session invitation and to indicate successful completion of a non-INVITE request.

Table 3.1: Classes for Response Codes

Class	Description	Action
1xx	Provisional	Indicate status of the session prior to completion. Are also called provisional replies.
2xx	Success	Request has succeeded. Retransmission of messages is stopped. For an INVITE, send ACK.
3xx	Redirection	The UAS or an intermediate proxy has returned possible locations for the AOR we are trying to reach.
4xx	Client error	The request has failed due to an error in the UAC.
5xx	Server failure	The request has failed due to an error in the UAS.
6xx	Global failure	The request has failed. It cannot be fulfilled by any server.

300 Multiple Choices The `300 Multiple Choices` is a redirection message. It contains multiple `Contact` header fields, which specify the URIs where the user, for the AOR specified in the original request, can be found.

400 Bad Request The `400 Bad Request` indicates that the server did not understand the request. The server sends this message if the request was malformed or did not contain all necessary headers.

501 Not Implemented The `501 Not Implemented` response indicates that the UAS does not support the method in the request.

3.2 SIP in IP Multimedia Subsystem

The IP Multimedia Subsystem (IMS) will be the IP-based core of the future packet switched 3G networks. It is a collaborative effort of the Internet Engineering Task Force (IETF) and the Third Generation Partnership Project (3GPP) to bring the cellular networks to a new era of communications. The idea behind IMS is to

provide Internet services anywhere and anytime for the mobile users and to create a common platform for various multimedia services.

IMS uses popular Internet protocols, like SIP, RTP, RTCP, COPS, H.248, Diameter, and DNS with minor modifications, to enable multimedia communications between the mobile users all around the world. The Internet Protocol version 6 (IPv6) will be used as a network protocol for the IMS networks, as specified in RFC 3316 [4]. The requirements for the IMS networks are described in [10]:

1. Support for establishing IP Multimedia Sessions.
2. Support for Quality of Service (QoS) negotiation.
3. Support for interworking with Internet and circuit-switched networks.
4. Support for strong operator control.
5. Support for rapid service creation.

Sipilä [52] describes briefly what changes and additions are needed to be done to Internet protocols for IMS use. The 3GPP TS 22.228 Release 5 [1] contains the whole IMS specification.

3.2.1 3G/IMS Network Architecture

Figure 3.5 shows the basic components in the IMS network. Some of these components have evolved from their General Packet Radio Service (GPRS) counterparts, while some of them are completely new.

Home Subscriber Service (HSS) is a central database which holds all user related data, like location, security and user profile information.

Subscriber Location Function (SLF) is needed only if there are more than one HSSs in the network. The SLF maps every user to a correct HSS.

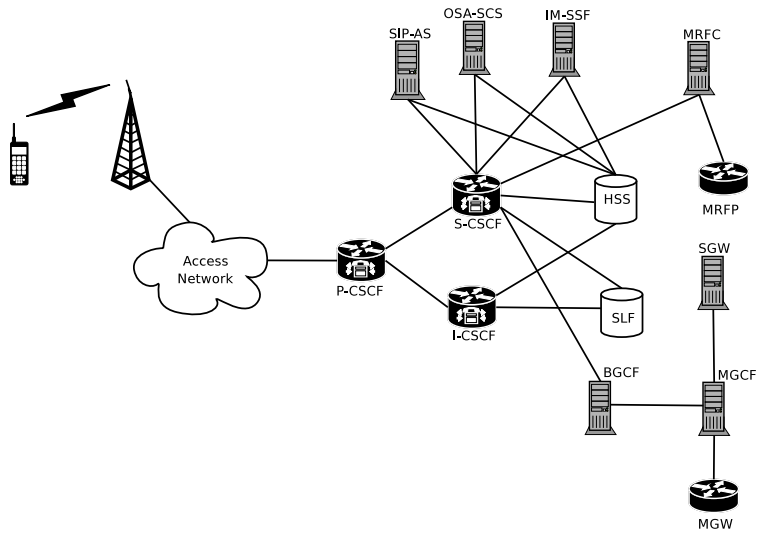

Figure 3.5: IMS network architecture.

Call Session Control Function (CSCF) is a SIP proxy server. There are three types of CSCFs existent in the IMS network, Proxy-CSCF (P-CSCF), Interrogating-CSCF (I-CSCF), and Serving-CSCF (S-CSCF). P-CSCF is the outbound SIP proxy for a mobile user. It is the first point of contact between the IMS network and the user terminal. P-CSCF includes functions like compression and decompression of the SIP messages between the user terminal and the IMS network. It also includes security features, like authentication, and checking of SIP messages for correctness.

I-CSCF is a SIP proxy used as a gateway to operator's network from other operators' networks. The SIP messages are routed between operators via I-CSCFs. Proxies use DNS to solve addresses of I-CSCFs, like described in section 3.1.2.2.

S-CSCF is a central node for SIP signaling, all the SIP signaling sent by the user terminal is routed via S-CSCF. S-CSCF works as a SIP proxy; it performs session control and acts as a SIP registrar; it also routes messages to application servers when needed.

Application Servers are divided into three different groups. There are OSA-SCS (Open Service Access – Service Capability Server) and IM-SSF (IP Multimedia Service Switching Function) application servers for older applications, originally developed for GSM networks. They provide interfaces to the OSA framework Application Servers and to the CAMEL (Customized Applications for Mobile network Enhanced Logic) services. The SIP AS is the native application server for IMS. Future applications will be purely hosted on the SIP ASs. An application server can act as a SIP proxy, UAS, UAC or B2BUA.

Media Resource Function (MRF) is a kind of a media server. It can generate, mix, and transcode between different media streams in the network. It can for example play announcement to the user when the service the user is trying to reach is not available. The MRF is divided into two parts, to the Media Resource Function Controller (MRFC) which performs the SIP signaling, and to the Media Resource Function Processor (MRFP) which performs the actual media-related functions, like the mixing of multiple media streams.

Breakout Gateway Control Function (BGCF) is a SIP proxy which routes calls destined to the circuit-switched network.

Circuit-Switched Network Gateway functionality is provided by the Signaling Gateway (SGW), Media Gateway Control Function (MGCF) and Media Gateway (MGW). The SGW converts the lower level protocols like Message Transfer Part (MTP) to SCTP, whereas the MGCF converts the actual circuit-switched domain signaling protocol to SIP. The MGW converts between RTP and Pulse Code Modulation (PCM) and performs the transcoding if necessary.

3.2.2 GPRS Access Network

Typically a General Packet Radio Service (GPRS) or some GPRS compliant access network is used to access the IMS network. The connection to the IMS network is established via the Serving GPRS Support Node (SGSN) and Gateway

GPRS Support Node (GGSN) nodes of the operator. The SGSN is the node nearest to the mobile node and it is selected based on the handset location. The GGSN will be the same no matter where the user is located. The SGSN relays user data to the GGSN and the GGSN provides a direct connection to the IMS network.

3.3 SIP in Next Generation Networking

IMS is only beginning for the Next Generation Networking (NGN). In the future Next Generation Networking (NGN) the same core network is shared between the fixed and the mobile networks, and the same network carries all traffic, be that ordinary voice, video or data. NGN will facilitate SIP as the underlying signaling protocol, and thus will enable the same services to be accessed through the fixed and mobile networks. This also means that the same peer-to-peer application can be used from the fixed and wireless terminals if the application is built on the SIP protocol. Thus, when using service like this, users will be able to access all the same information wherever they are.

Chapter 4

Requirements for Mobile Peer-to-Peer

Mobile peer-to-peer has specific requirements compared to fixed peer-to-peer networks. Terminal capabilities are more limited than in the fixed networks where the terminals have more than enough processing power and storage capacity. Also, in mobile networks the shared content is mostly self created, whereas in fixed networks the content is mostly professionally created.

Although most of the studies concerning peer-to-peer networking have been made with fixed, high capacity networks in mind, research on mobile peer-to-peer is slowly emerging. The special requirements and constraints of the mobile platform prevent us using the peer-to-peer protocols developed for the fixed networks in the cellular domain; thus extra work has to be done to enable mobile peer-to-peer communication.

We will start this chapter by studying those special constraints and requirements of the mobile platform. Then, we will discuss user requirements for mobile peer-to-peer application. Next, we will discuss what would be the optimal architecture for a mobile peer-to-peer application. Later, we will compare mobile peer-to-peer file sharing to other mobile content sharing methods and compare it also to fixed peer-to-peer file sharing. Last, we will review past research done on the subject.

4.1 Technical Constraints of Mobile Platform

The mobile platform has special constraints which have to be taken into account when designing a peer-to-peer application for mobile terminals.

4.1.1 Memory Size

It is important to minimize the memory usage of the application, because modern mobile phones usually have many programs running concurrently, and these programs have to share the limited memory. Although the memory capacity is slowly increasing, it is still one of the scarcest resources on the mobile phone.

Memory-use minimization means that we generally have to minimize state information in the application. For example, holding information for hundreds or thousands of peers (like in Freenet) is not a good idea in a mobile application.

4.1.2 CPU Performance

As with the memory, the same restrictions go with the CPU cycles, thus usage of complex algorithms should be avoided if possible. Although, thanks to Moore's Law, the CPU speed is a less of an issue with the current mobile phones than it used to be. The CPU usage also directly correlates with the battery usage.

4.1.3 Access Network Parameters

Another big constraint which has to be accounted for when designing a peer-to-peer application is the bandwidth usage of the program. The air interface used to access the mobile network has very limited bandwidth, and this bandwidth has to be shared between multiple users in the cell. This is why it is vital to minimize bandwidth usage of the peer-to-peer protocol. Using distributed peer-to-peer protocols, which flood requests to many nodes, is inherently a bad idea bandwidth wise.

Compared to fixed networks, wireless networks are more prone to errors, have lower bandwidth and higher delay and packet-loss; these matters arise from the nature of the wireless data path, and from the greater protocol overhead and complex error correction schemes used in the air interface. Wireless access networks are also more prone to communication interruptions of undetermined duration. These issues pose additional constrains to the application design.

4.1.4 Screen and Keyboard Size

Small screen size of the mobile terminal restricts the user interface design for the application. The peer-to-peer application needs to have various information visible to the user, such as the files being downloaded and uploaded, the files in the queue, current bandwidth usage, and the search dialog. Although screen sizes of the mobile phones are increasing and the screens are getting higher resolutions, we are still far away from an ordinary PC screen.

Also small numeric keyboards are not the best ones for inputting extensive textual searches. Some phones have full QWERTY-keyboards, but these are a minority. Thus, UI design has to take the limited features of the input device also into consideration.

4.1.5 Battery Capacity

Faster processors and larger screens consume the battery faster than before. Current trend of ever decreasing physical size of the mobile terminal is inevitably shrinking the battery too. The battery industry tries to keep up with increased power demands, but best results can be achieved together with the use of modern batteries and low-power CPUs and screens. Also, limiting transmission over the wireless link helps with the energy efficiency.

4.2 Special Needs of Mobile Environment

In addition to the restrictions of the terminal, the mobile environment has special needs posed by various access methods, operator and operational requirements [35].

4.2.1 Support for Various Access Networks

The mobile peer-to-peer application has to take into account different kinds of access methods to the mobile network and have good performance in all of them; such as, GPRS, EDGE, WCDMA, HSDPA, Wi-Fi and Bluetooth [12]. These access methods have differences in error rates, bandwidths and delay characteristics; indeed, the speed of the access link and end-to-end delay characteristics can differ in orders of magnitude, as seen in table 4.1.

Table 4.1: Theoretical maximum bandwidths and minimum round trip times to the fixed network for various access networks

GSM Data	9.6 kbps	< 300 ms
HSCSD	57.6 kbps	< 300 ms
GPRS	115 kbps	< 1000 ms
EDGE	384 kbps	< 800 ms
WCDMA	2 Mbps DL[1] / 384 kbps UL	< 300 ms
HSDPA	3.6Mbps DL[2] / 384kbps UL	< 100 ms
HSUPA	14 Mbps DL / 5.76 Mbps UL	< 100 ms
WLAN	54 Mbps	< 10 ms

Another problem is roaming between these various access network types. The application has to be able to work in a changing environment without introducing

[1] Current devices are typically capable of 384 kbps.

[2] This is for the first phase HSDPA terminals, the second phase terminals will offer 14 Mbps downlink.

errors in the data or other malfunctions. If the network layer does not incorporate a seamless integration of different access networks, like providing handovers between them, the application has to have a way to handle effects of these changes, such as, support for changing IP-address, built in.

Holistic connectivity management described in [19] is a method for handling multiple underlying network connections and changing between them efficiently without user or application noticing this.

4.2.2 Operator Control

Traditionally the network operator has wanted to have control on all the services in its network. Because peer-to-peer is inherently distributed, implementing some kind of operator control is a non-trivial issue. However, centralized and hybrid peer-to-peer networks make use of super-peers; if the super-peer is under operator control, the operator has virtually full control on the content available in the peer-to-peer network. The actual content is still transmitted between the end nodes, but the super-peer acts as a broker for client communications. More control can be added by facilitating special media gateways that inspect the actual content sent between the end nodes.

4.2.3 Feasible Bandwidth Pricing

At the moment, the file transfer prices are still quite high for end-users, although a trend can be seen that the operators are starting to provide flat rate connections, meaning that the user pays a fixed amount of money to the operator every month regardless of how much the user actually transfers data. Peer-to-peer can become successful in the mobile domain only if the file transfer costs are bearable to the user.

Current data transfer pricing does not encourage users to share files, because the users have to pay for both received and sent bytes. There is little reason for any user to pay to let other people to download files from his phone. Also, the user generally has no control how much other people download from him, so with the

current pricing, the mobile peer-to-peer application can quickly become a major money eater. To create a successful mobile peer-to-peer service, uploading should be free of charge, or give extra credits to the user that can be later used for downloading. The more files the network has, the more valuable users see this network.

4.2.4 Economical and Legal Issues

The economical and legal issues are the same for mobile networks as in the fixed network peer-to-peer communications; these issues are reviewed in chapter 2. In the mobile networks, there is also the risk that the mobile operator is seen as responsible for the material exchanged through its peer-to-peer service. This is because the operator has traditionally had total control on its services, and thus control on the content distributed through its services.

4.3 User Requirements

The user expects a mobile peer-to-peer application to provide him the same capabilities that the applications for fixed networks provide. These expectations include quick response times for searches, rapid downloads and easy-to-use user interface. However, the user will probably tolerate somewhat inferior performance compared to the fixed peer-to-peer networks because the mobile network is available everywhere.

Mobile peer-to-peer network should include user group management features, which allow users to create closed groups where files can be swapped. This is important because a lot of content is self-created and most users want to restrict access to this material only to people they know. Groups can be formed among families, friends, or any other relevant set of people.

4.4 Optimal Architecture

It can be argued that the centralized and the hybrid peer-to-peer architectures are most suitable for mobile peer-to-peer networking. If the super-peer is located in a fixed network, mobile nodes do not have to handle huge signaling loads during searches. All a mobile terminal has to do, is to send the search to the super-peer it is connected to. The super-peer consults its own database (and other super-peers in case of the hybrid architecture) and sends a reply to the mobile terminal.

The centralized and the hybrid architectures can also provide the mobile operator some control over the shared files. As long as the operator controls the super-peer, it can enforce sharing policies and ban unwanted files. Of course, some users consider this as a bad feature. Operator control also inherently kills the possibility for any kind of anonymity in the network.

The choice between the hybrid and the centralized architecture is dictated by the overall network design. If the idea is that the network operator controls the super-peer, and that the users of different operators are able to share files, the hybrid architecture is the most suitable. In this architecture every operator can run its own super-peer which connects to the other super-peers ran by the other network operators.

It is also possible that people want to run their own super-peers, e.g. a family or sports club might want to run its own super-peer which is accessible only for its members. If no connection to outside world is needed, the centralized architecture is fit; otherwise, the hybrid architecture can be used.

From the viewpoint of the mobile application the centralized and the hybrid architecture look the same. The mobile terminal is still connected to one super-peer; it does not care if the answer is combined from the results of multiple super-peers connected in peer-to-peer fashion, or from the database of a single super-peer. Because of this, the same application can be used for connecting to centralized and hybrid peer-to-peer networks.

More arguments for choosing the hybrid or the centralized architecture are presented in section 4.7.5.

As it was discussed in chapter 2, the unstructured architectures are not generally suitable for file sharing applications when the search functionality is needed. Thus they were left out altogether from this evaluation.

4.5 Comparison with Other Mobile Content Sharing Methods

Today, the only way to share files in the mobile domain is to send the files to a centralized sever where they can be fetched by anyone, e-mail the files to some specific person, or to use some third party application that handles the image uploading to a centralized server and the user group management. However, there is no simple way to make information available for a larger group of users without first uploading this information to a public server.

Mobile peer-to-peer architecture allows us to share content without initially uploading it all to a centralized server. This is useful when the user is not sure if anyone ever wants to access the content, and thus prevents unnecessary transfers. The files are kept on the user's handset, and sent over the air interface only when someone actually requests the information. However, this also means that if a shared file is popular, it will be transmitted several times over the limited-capacity air interface; thus, complementing peer-to-peer network with some kind of caching mechanism can improve the performance greatly for popular downloads.

Mobile peer-to-peer implements an integrated search function which looks up information through the network, whereas searching files from the WWW is much more complex, because different users use different web servers, and thus the information is scattered all over the Internet.

4.6 Comparison to Fixed Peer-to-Peer

Mobile peer-to-peer networking can be seen more as an addition to fixed peer-to-peer networking than as a substitute for it. It is likely that mobile peer-to-peer is

used to access material when on move, and when a fixed network is not available.

It is probable that a lot of content is self-created in the mobile peer-to-peer networks, e.g. pictures and videos shot with the mobile's camera. This content is mostly shared among friends and families, because the content has no relevance to the general public. Piracy is also a less of an issue in the mobile networks — at least initially. It is safe to predict that as the network bandwidth increases and the terminals get more storage capacity, swapping of music and movie clips becomes more popular in the mobile domain too.

The hybrid and the centralized peer-to-peer architectures are most suitable for mobile peer-to-peer networks, whereas any architecture can be used in the fixed network. This is because the mobile networks are not capable of handling the extensive signaling generated by large pure peer-to-peer networks. In the hybrid architecture the super-peer nodes can be located in the fixed network, and thus the heaviest signaling load is moved away from the mobile domain. Existence of the super-peers can also give the operators the control over the peer-to-peer service. However, the pure peer-to-peer architecture might be feasible if the network size is small enough, e.g. for family use.

Table 4.2 contrasts key features of the fixed and mobile peer-to-peer networks.

4.7 Past Work

Peer-to-peer research in mobile networks has been done for, both, real-time and non-real-time communications. The real-time peer-to-peer communication research is mostly about the Internet telephony, and the non-real-time is centered on the file sharing networks. We will focus on the non-real-time research because it is most relevant to our application.

Discussion about using different peer-to-peer protocols on the mobile platform is presented in [19]. The paper argues that the hybrid peer-to-peer architecture is the most suitable architecture for mobile networks because many functions can be moved from mobiles to the super-peers. The paper also proposes a framework called the *Plug-and-Play Application Platform (PnPAP)* that allows mobile appli-

Table 4.2: Key features of fixed and mobile peer-to-peer

Fixed peer-to-peer	**Mobile peer-to-peer**
Available at fixed locations	Available anywhere
Plenty of bandwidth	Limited Bandwidth
Terminals have nearly unlimited storage capacity	Storage capacity is limited
Content is mostly professionally created	Content is mostly self-created
Piracy is a major problem	Little piracy
Open for all	Group centric
Pure architecture feasible in small and medium size networks	Pure architecture feasible only in small networks
Centralized and hybrid architectures feasible in small and large networks	Centralized and hybrid architectures feasible in small and large networks

cations to use many underlying peer-to-peer and session management protocols dynamically.

4.7.1 Gnutella in Mobile Environment

Another paper [55] argues that the usual peer-to-peer file sharing networks, such as Gnutella, are not suitable for the mobile environment due to their bandwidth consuming broadcast nature. Instead, a modified architecture for Gnutella network is proposed where a mobile agent in the fixed network works on behalf of the mobile device. The mobile agent is part of the Gnutella network, where it acts as a normal Gnutella peer, and has vital information like the file-list of the mobile device. The mobile device and the agent communicate using a light-weight protocol. Thus, the mobile agent can handle most of the signaling traffic, such as searches, and direct only download requests to the mobile device. The mobile device can perform the actual file transfer directly with the other end node or,

alternatively, the mobile agent can do this on behalf of the mobile device.

The mobile agent also abstracts away mobility of the mobile device, because the agent has a fixed IP address, even though the phone has a rapidly changing IP address. The mobile device just has to update its IP address to the mobile agent whenever it changes.

4.7.2 Mobile eDonkey

Oberender et al. [35] have developed a mobile peer-to-peer file transfer architecture based on the eDonkey protocol. Some modifications to the original architecture have been done. This architecture includes modifications to the Index Server, a Crawling Peer and a Cache Peer. The index server keeps track of the popularity of the files in the network, and exports this popularity data to the cache peer. The cache peer stores these popular files, and the crawling peers support the index server by linking it to other index peers in the Internet. The resulting architecture is something between centralized and hybrid peer-to-peer.

The benefit of the cache peer is that the popular files can now be stored in the core network, and they do not have to be transferred multiple times over the air interface. When a file is traditionally transferred from one mobile to another, the data goes over the air interface twice. Reducing this traffic to half is the main benefit of the cache peer.

Modifications to the index server, which is a kind of a super-peer, include logging features and redirections of searches to the cache peer, when the cache has the requested content.

The crawling peer is the node which creates a connection between the index server of the mobile network, and the other index servers in the public Internet. Usually in the eDonkey architecture, the client connects to multiple index servers if the local index server does not have enough/requested files listed. Because mobile nodes have lesser capabilities, it is rational for them to have only one connection to the index server, and let the crawling peer handle the traffic to the other index servers; thus, enabling the mobile node to acquire all the content available in the

mobile network and in the Internet combined.

Another paper [36] by the same authors proposes content replication aka. caching as a general mobile peer-to-peer design feature to improve performance and to reduce traffic over the air-interface. It recognizes that because mobile networks have a strictly hierarchical architecture, i.e. the traffic between mobile phones has always to go through nodes in the core network, even when the mobiles are in the same cell, it is most useful to cache content near these core network nodes in economical sense.

4.7.3 JXTA for J2MR

Some peer-to-peer protocols have been designed with mobile platforms in mind, like JXTA community's JXME (JXTA for J2ME). Unfortunately JXTA is not available for Symbian C++, which is the programming language of many modern phones.

4.7.4 Other possibilities

SIP can be used as a signaling protocol for future IMS games. Reference [2] describes a gaming platform which sits between the game and the 3G network, and uses services provided by IMS for the game's needs; e.g. peer-to-peer connectivity, instant messaging, and QoS. The game server can be implemented as a application server, and the gaming clients running on user terminals communicate with the server using normal SIP messages.

4.7.5 Mobile Peer-to-Peer Architectures

Some work has been done evaluating suitability of different peer-to-peer architectures for mobile use. The general knowledge seems to be, that some kind of centralized or hybrid architecture is best for mobile terminals, whereas pure peer-to-peer is infeasible with the current bandwidths (e.g. [19, 55]).

Performance of different Gnutella topologies in wireless networks is evaluated in [6]. This paper notices that a semi-random mesh is the best topology for networks where all nodes are of the same kind. On the other hand, connected stars topology is found to be better for networks filled with heterogeneous devices, with different capabilities. The paper recognizes:

> A remarkable aspect of this [connected stars] topology is that the load is unequally divided: the star nodes are heavily loaded while the leaf nodes carry much less traffic. This can be useful in cases where the network consists of heterogeneous devices, such as mobile phones as leaf nodes and personal computers as star nodes.

Even though the paper is an evaluation of the old Gnutella protocol without super-peers (version 0.4), this idea directly applies to the concept of super-peers. A super-peer can carry most of the signaling traffic, while the mobile node only carries file transfers of its own and signaling messages to the super-peer.

Another paper [54] suggests the hybrid peer-to-peer architecture for mobile proximity applications, that is, for applications which are used between devices communicating over short distances. The paper notices that the hybrid architecture is well adapted to changing environments and in addition, the applications are easily managed, thanks to partial centralization (i.e. the super-peers).

Chapter 5

Developing Mobile Peer-to-Peer Client

The aim of the thesis was to design and implement a mobile peer-to-peer file transfer client which uses the Session Initiation Protocol as the underlying signaling protocol, and to test feasibility of the mobile peer-to-peer architecture on modern cellular networks. The motivation behind using SIP as the signaling protocol was to enable the application to be used in the future IMS networks that have native SIP support. Because SIP is inherently a peer-to-peer protocol it seems to be a suitable platform for signaling needs of a peer-to-peer application.

The SIP was chosen as the underlying protocol instead of some proprietary protocol because SIP is an integral part of the Third Generation IP Multimedia Subsystem networks. Using SIP as the underlying protocols, enables the network to be aware of the peer-to-peer application, and allows operators to implement functions such as charging for the peer-to-peer application.

To the author's knowledge, this is the first implementation of a peer-to-peer application on a mobile platform which uses SIP as the signaling protocol. Using SIP as the peer-to-peer signaling protocol is not totally unheard of; e.g. SIP is used as the signaling protocol for EarthLink's *SIPshare*, a file sharing program based on the pure peer-to-peer paradigm; however, *SIPshare* is not available for any mobile platform. [16].

This chapter will start with the review of the application's high-level design. Then, the software architecture and the software implementation is discussed. Later in the chapter, there is short review about the super-peer and the TCP-relay which were implemented concurrently with the client application. This chapter ends with a use case study which shows how the peer-to-peer application searches for a file and finally downloads that file from another peer.

5.1 High-Level Design

The client was designed to be modular and easy to use. The idea is that the client provides a simple search dialog where the user can input information about the content he is looking for. The user can initiate the search by specifying the name, type, size, hash of the file he wants to find. Searches using multiple parameters are also possible.

The user can configure the software so that the media he creates on his phone is automatically shared, e.g. user taken pictures and video clips can be shared immediately after they are saved in the phone.

The modularity means that the application will be easy to extend in the future. Ideas like peer-to-peer media streaming and chatting are considered to be implemented in the future. The modular design also allows some parts of the software to run in the background without the user interface running, so the mobile phone can serve files to other users even though the owner of the phone does not himself want to obtain any content from the peer-to-peer network.

Hybrid peer-to-peer was chosen as the underlying paradigm to cut down the traffic on the air interface. In this architecture, every mobile client connects to one super-peer, and these super-peers, located in the fixed network, create a pure peer-to-peer network between themselves. The hybrid architecture also allows the operator to have control of the peer-to-peer service by controlling the super-peer. The architecture is pictured in figure 5.1.

The hybrid architecture was chosen instead of the centralized architecture because it allows different mobile operators to run super-peers of their own. Also from

the client's perspective, there is no difference between the hybrid and centralized architectures. The choice only affects the implementation of the super-peer.

The super-peer is being created concurrently with the client. Overview of the super-peer functionality is presented in section 5.3.

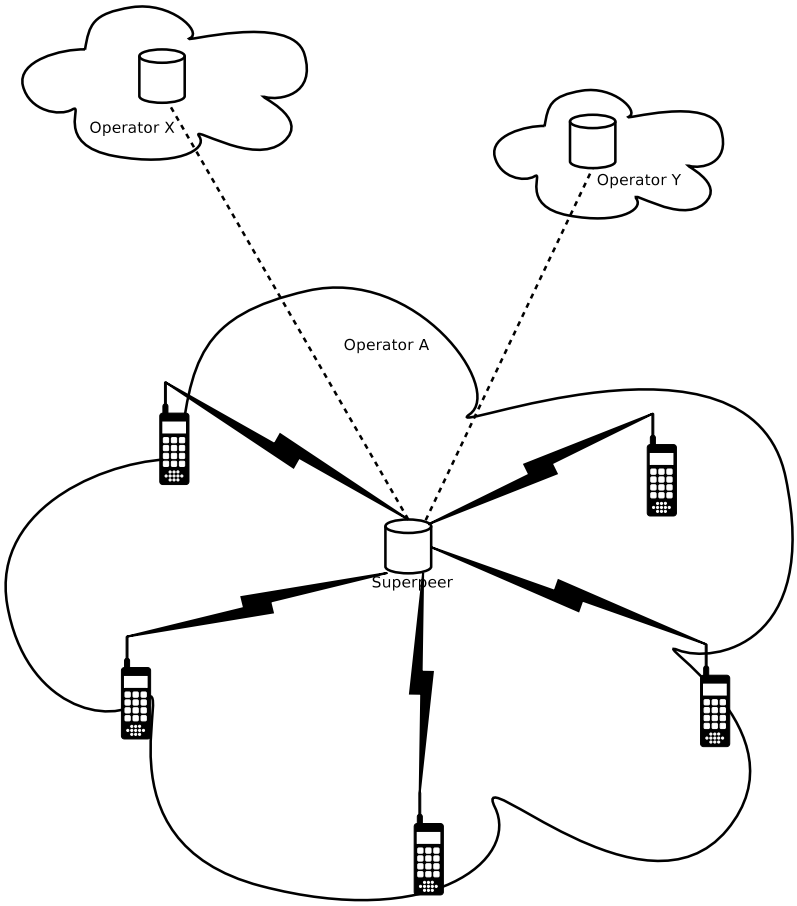

Figure 5.1: Mobile phones are connected to a centralized server located in the core network.

Our architecture is close to the architecture proposed in [55] (see section 4.7 for details); however, in our architecture the mobile agent is replaced with the super-peer; and one super-peer is used by many mobile devices. The same basic assumptions still apply; like that the super-peers hold file lists of the mobile devices, and their location information (i.e. the SIP URIs) — just like the mobile agents in the

other architecture do. Super-peers also handle large parts of the signaling traffic between themselves, and do not flood the mobile devices with unnecessary signaling traffic.

In our architecture mobile phones function as ordinary peers. These peers connect to a super-peer located in the fixed network. The access network is assumed to provide packet access to the super-peer and to other ordinary peers. This access can be provided for example via 3G/WCDMA (Wideband Code Division Multiple Access), GPRS (General Packet Radio Service), EDGE (Enhanced Data Rates For Global Evolution) or WLAN connection.

5.1.1 Software Architecture

The software architecture described here is originally presented in a paper by the project team [8].

The mobile peer-to-peer client consists of three separate processes. The overall architecture of the client is illustrated in figure 5.2. The basic functionality is divided into four modules. These modules are the registrar, the finder, the transfer, and the graphical user interface module. The registrar and the finder modules are implemented in one process, called the core process. The transfer module and the GUI module have their own separate processes. The motivation of the process separation is to allow parts of the software to be running independently, e.g. the user can shut down the GUI without stopping the ongoing file transfers.

The software uses file hashes as unique file identifiers. All search replies from the super-peer and download initializations messages between ordinary peers contain file hashes.

5.1.1.1 The Core Process

The core process consists of the registrar module and the finder module. This process contains core functionality which is general enough that it can be used with different kinds of modules, like with the file transfer and chatting modules. The registrar part of the process is responsible for providing information to the

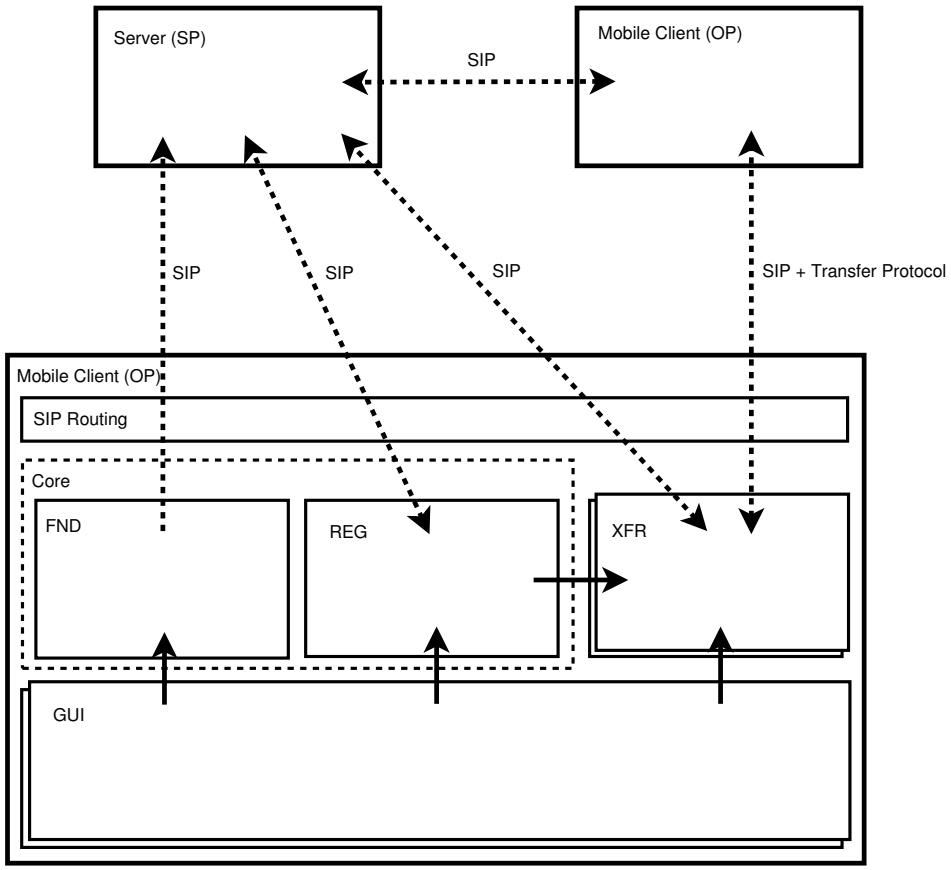

Figure 5.2: Client software is divided in three processes: GUI, Transfer and Core.

super-peer about the current state of the client, i.e. what services are running at any moment. The super-peer subscribes to this information from the registrar, and the registrar then notifies the SP whenever some services are started or stopped. While we are supporting only file transfer at the moment, the subscription of this state information is not yet implemented.

Initially the registrar was also meant to send SIP REGISTER messages; however, this functionality was left out because the Series 60 SIP infrastructure provides a built-in application, the SIP Profile Manager, for SIP registrations. Now SIP Registrar functions as a mere gateway for controlling the SIP Profile Manager.

Another half of the Core process is the Finder module. The finder is the component generating SIP requests used for file searches. The GUI passes the search

information to the finder, which then creates an appropriate SIP INVITE message, containing the search information in XML format as the payload. When the super-peer replies to the search with 606 Not Acceptable message, the search results are extracted from the reply and passed back to the GUI.

5.1.1.2 The Transfer Module

The transfer module is responsible for transmitting the requested file between the two peers, and updating the client's file list to the super-peer, so that the super-peer has info about added and removed files. The module features a simple TCP file transfer protocol. Multi-source downloads and file transfers over MMS messages might be implemented in the future.

The transfer module starts file transfers by sending a SIP INVITE message to another peer. The INVITE message contains SDP as the payload. The SDP message defines the IP address and the port number combination to use for the file transfer. It also defines the hash for the requested file in the session name field. The peer, who receives the INVITE message, can recognize the requested file by checking the hash. It replies with a 200 OK message, waits for an ACK, and starts the file transfer to the address specified in the initial INVITE request.

The transfer module communicates with the super-peer about the files the client is holding. Whenever the peer-to-peer application is started, the transfer module will upload the whole file list to the super-peer. Later, when files are added to or removed from the shares, the transfer module will send update deltas to the super-peer. These updates are sent inside MESSAGE requests.

5.1.1.3 The Graphical User Interface

The graphical user interface is the process which passes information from the user, via other modules, to the network and back. It also functions as the component which ties the core and the transfer processes together because those processes do not have a direct signaling connection between them. For example, when searching for a file, the user inputs the file information in the search dialog and the

Graphical User Interface passes the search in XML to the finder. When the finder receives a reply to the search from the super-peer, it passes the reply in XML back to the GUI. The search results are then shown to the user. There can be zero, one or more hits for the search. The user can next specify the file he wants to receive and the GUI passes the information about the selection to the transfer module. Finally, the transfer module performs the file transfer.

The GUI can also configure other modules and query statistics from them, e.g. it can ask the transfer module how much of a file has been transferred or how many transfers there are waiting in the queue. It can also activate and deactivate other modules. In case we had streaming and chatting modules installed in addition to the file transfer module, the GUI could activate these separate modules and ask the registrar to notify the super-peer about the new situation.

5.1.2 Functionality Provided by SIP

SIP was chosen as the underlying protocol because it is integral part of the IMS. As SIP is guaranteed to work in the IMS networks there is a quarantee that our peer-to-peer application will work in these networks too. When the NAT and firewall traversal functions are provided by SIP, we do not have to deal with these issues in our application at all. Also, using standard protocol also allows more rapid application development compared to inventing a signaling protocol of our own.

SIP allows the operator to implement control functions on the peer-to-peer service and necessary charging functions. Implementation of these functions is easy because all operator equipment are SIP aware.

In addition to the things mentioned above, the features of SIP are used in two additional ways. First, the IMS/SIP framework has a concept of Application Server (AS). The Application Server for the peer-to-peer service is used to store information about content available on the clients in the network; it functions as super-peer. The super-peer will answer to the content queries sent by the peer-to-peer clients.

Second, the SIP infrastructure is used to locate the user who has the requested content, and to forward session establishment requests to that user. User agents communicate with each other using the SIP URIs, so the end nodes can contact each other, without knowing each others' current IP addresses. The SIP URI of the other end is all that is needed to contact the node, the SIP infrastructure takes care of the message routing to the right terminal.

5.1.3 SIP Messages

The application uses four different SIP requests: (1) the REGISTER request is used for registering to the SIP network, (2) the INVITE request is used for searches and to start actual file downloads, (3) the MESSAGE request is used to upload information about client's shares to the super-peer.

Example messaging flow diagrams and message contents can be seen in appendix A.

5.1.3.1 SIP Register

Before sending any other SIP requests, the application has to register itself to the SIP registrar. Registration creates a mapping between the current contact URI and user's public SIP URI.

This part is executed by the SIP Profile Manager; our application merely issues a call to the profile manager to register a predefined SIP profile.

5.1.3.2 File List Update

Mobile terminals use the MESSAGE[1] request to upload file lists to the super-peer. Terminals can add or remove separate files in super-peer's database, and they have an option to clear all files marked for them in the super-peer's file list in case of a client shutdown or in need for a clean startup. Whenever the mobile

[1]The original idea was to use the PUBLISH request to upload information about new files to the super-peer; however, due to problems with Nokia SIP proxy rejecting PUBLISH requests we used MESSAGE requests instead. Changing back to the PUBLISH request may be considered in the future.

terminal starts, it has to upload the full file list to the super-peer, so the other nodes can search for the files of the user. Whenever new files are added to or removed from the file system, the mobile terminal automatically sends MESSAGE requests to the super-peer. These requests contain deltas about the recent file changes. The file list update messages are split into several 1400 byte requests to prevent message fragmentation in the network.

File list updates include the list of the files the node has in the Extensible Markup Language (XML) format. Appendix B presents the Document Type Definition (DTD) for the file list update XMLs.

5.1.3.3 Content Search

The client is using the INVITE request to search files. The INVITE request containing the search information is sent to the super-peer. The super-peer will process the search and reply with information about the nodes that have the requested content. However, we do not want to create long lasting sessions between the mobile terminal and the super-peer, so the super-peer returns the search results in a global error message, 606 Not Acceptable, which terminates the session establishment between the mobile client and super-peer. An error message was chosen for the reply because we did not want to establish a session with the super-peer during the search. Thus, we had the option of generating a redirect or an error message to prevent session establishment, a global error message, 606 Not Acceptable, was chosen in lack of a better message. Figure 5.3 has an example search message sent from a client to the super-peer.

Usage of the INVITE request for searches can be argued, e.g. the MESSAGE request could have been used instead; however, the MESSAGE request is usually used for carrying actual user data, not signaling data, and in the view of the peer-to-peer application, the search is inherently signaling. It seems that SIP is actually missing a general request-reply type of a signaling message that could be sent outside a dialog and that would not create a dialog of its own. OPTIONS request fits into this description, but it is not general enough for request-reply use. The OPTIONS requests are used to query peers about the SIP options they support.

```
INVITE sip:sp@realm SIP/2.0
Route: <sip:130.233.154.17;lr>
Via: SIP/2.0/UDP 130.233.154.32:5060
;branch=z9hG4bKO0PR9jA9aKyRG
From: sip:player2@realm;tag=lwDR9oR_n-
To: sip:sp@realm
Contact: sip:player2@130.233.154.32
Supported: sec-agree
CSeq: 344 INVITE
Call-ID: Hk7R9hhoou0OICbnAnO5n1Mnf5mjwI
Max-Forwards: 70
Content-Type: text/xml
Content-Length: 81

<?xml version="1.0" standalone = "yes" ?>
<request>
<name>Meeting*</name>
</request>
```

Figure 5.3: Search message sent from a mobile client to the super-peer

Also, it can be thought that we INVITE to the file we are searching for, so even though in SIP's perspective the session we are trying to establish between the mobile terminal and the super-peer, and then between the two mobile terminals for the file transfer, are different, we could see this as one session on the peer-to-peer level, a session to get some specific file.

Currently the size of the reply message is bounded to 1400 bytes to prevent packet fragmentation in the mobile network. In the future more results can be received by sending multiple search messages each specifying an offset which tells the super-peer which search results we want to receive.

Appendix B presents the DTDs for the file search and reply XMLs.

5.1.3.4 Content Download

The INVITE message is a natural choice for starting a download session between two mobiles. The mobiles create a session and communicate the session parame-

ters, like the hash of the file to be transferred via SDP inside the INVITE message. An example INVITE for the file transfer is shown in figure 5.4.

```
INVITE sip:player1@realm SIP/2.0
Route: <sip:130.233.154.17;lr>
Via: SIP/2.0/UDP 130.233.154.32:5060
;branch=z9hG4bKiY7R9umiJuSy4
From: sip:player2@realm;tag=LIjR9gJ6lL
To: sip:player1@realm
Contact: sip:player2@130.233.154.32
Supported: sec-agree
CSeq: 345 INVITE
Call-ID: xw7R9l7AMeVg__Kh8YA8n28JLxNZMy
Max-Forwards: 70
Content-Type: application/sdp
Content-Length: 159

v=0
o=player2 5771190683978352867
5771190683978352868 IN IP4 130.233.154.32
s=281dcbfd8a2a1b72
c=IN IP4 130.233.154.32
t=0 0
m=application 1234 TCP MP2P
```

Figure 5.4: INVITE message initiating file transfer between two peers

5.1.3.5 Motivation Behind Chosen SIP Methods

The SIP protocol provides more methods than used in this application. Motivation for choosing the specific methods used in this program were discussed in previous sections. Tables 5.1 and 5.2 provide motivation why the other SIP messages were not used instead.

Table 5.1: Suitability of different SIP methods to different uses

Method	**Suitability for peer-to-peer use**
BYE	This method can be used only for terminating sessions already established with the INVITE request. This request is not suitable to be sent alone without sending an INVITE request first. The BYE request can be used to terminate a download session established with an INVITE request.
CANCEL	This method is used to cancel pending requests (practically only INVITE requests). This method is not suitable to be used in our peer-to-peer application.
OPTIONS	Because this method does not establish a session, and because it can be sent outside of any existing session, this request would be a good general request-reply method if it was not semantically defined to be used only for requesting information about the SIP capabilities the other end supports.
INFO	The INFO request can be used only inside an existing session, thus it is not suitable for peer-to-peer signaling.
MESSAGE	This request can be sent outside of any existing session. This method is usually used to transmit actual user communication, for example text messages. It should not be used for application signaling. For example, in IMS the user might be charged for all MESSAGE requests because they are thought to be user readable messages, and of course we want the application signaling to be free-of-charge.

Table 5.2: Suitability of different SIP methods to different uses

Method	**Suitability for peer-to-peer use**
PUBLISH	Would have been a good message for publishing file lists to the super-peer. However, due to the problems with the current SIP-proxy, this message could not be used.
UPDATE	This method is used to modify the state of a pending session. Because we want to send all our messages outside of any existing session, this method is not suitable for our uses. In the future, this method may be used to request network resources for file downloads or music/video streams during the session establishment.
SUBSCRIBE and NOTIFY	Usage of these methods creates state to the endpoint where we send the SUBSRIBE request. These messages could have been used for subscribing the file list from the mobile terminal, so that the mobile terminal would always send a NOTIFY request to the super-peer when there are changes in the file list. However, the MESSAGE request was used instead because it allowed more rapid initial implementation.
REFER	This method is used to refer the message recipient to some other location (i.e. URI). Usage of this method would be justified if search results or actual downloadable content were available in some other location, e.g. in the World Wide Web. However, in our architecture this is not the case, so this request is not used.
PRACK	This method is used to provide reliable provisional responses. It is not suitable for peer-to-peer application signaling.

5.2 Implementation

The software was implemented on Nokia's Series 60 platform, which uses the Symbian operating system underneath. This is a platform used on many popular smart phones, and provides vast number of different APIs (Application Programming Interfaces) for performing different tasks. The Series 60 platform also has a SIP stack available for it, so with its large user base and good software support it was chosen as the implementation platform.

5.2.1 Programming Language and SIP Stack

Applications for the Series 60 platform can be programmed in C++, Java and now also in Python. The consideration was done between C++ and Java, because Python was not seen as a good language for larger software projects. C++ was chosen as the implementation language because it provides more complete interfaces for mobile programming and poses less restrictions than Java. It's also more efficient performance-wise than Java. Also, no phone on the market provided access to the SIP stack via Java at the time of the implementation.

Nokia SIP stack was the only free SIP stack for Series 60 platform available when the implementation of the software started, and because we did not want to spend time porting some open source SIP stack to the Series 60 platform, the Nokia stack was chosen. Nokia SIP plugin was installed on Nokia 6680 to enable SIP functionality on the real phone.

5.2.2 Implementation Tools

We used *Microsoft Visual C++ .NET of Microsoft Visual Development Environment 2003* [28] with *Series 60 2nd Edition SDK for Symbian OS Supporting Feature Pack 2* [33] which integrated the mobile development tools with the MS Development Environment. The tools also contained Series 60 phone emulators on which much of the initial testing was done.

Nokia SIP Plug-in 3.0 and 4.0 for Series 60 [34] were used for providing SIP

functionality to the programming environment.

We needed to use an external XML parser, because the Series 60 platform did not include any of its own. Open source project, *SyExpat* [56], was chosen as the parser.

Nokia SIP Server Emulator [34] provided simple SIP Proxy and Register functionality. It was used during the software testing to enable SIP communication between the emulators and between the real phones.

partysip SIP proxy server [30] was used instead of the Nokia SIP Server Emulator during the measurements.

5.2.3 Implementation Details

5.2.3.1 SyExpat

SyExpat is a stream-oriented XML parser, i.e. it reads the document sequentially, tag by tag, from the beginning to the end. The user has to implement handlers which are called when the parser discovers associated structures in the parsed document.

SyExpat does not validate XML documents against DTDs, so there is no guarantee that the received XML is structured right application wise.

5.2.3.2 SIP Profile Manager

Registration to the SIP registrar is handled by the SIP profile manager application. This application comes with the Series 60 SIP stack and provides an API, via which other applications can ask the profile manager to register to predefined SIP registrars. Because of the SIP profile manager, the application programmer does not have to create any registration logic.

5.2.3.3 SIP Routing

SIP routing uses the SIP ECOM plugin to redirect incoming SIP requests to the correct application. It senses differences between different content types and media (m=) fields in the SDP headers.

5.2.3.4 Process Communications

The three processes communicate with each other using the client-server architecture of Symbian. Transfer and core processes provide interface classes which the GUI can link to and ask those processes to perform functions.

5.2.3.5 Implementation Issues

SIP Stack There were some issues with the Nokia SIP Stack. When an instance of the stack is created in C++, the programmer has to specify the application UID of the program using the stack. In our architecture the stack is used by the core and transfer processes. However, for some reason, the SIP Stack cannot be bound to executables which run in the background, thus it cannot be initialized with the UIDs of the core and transfer processes. To overcome this limitation we had to use UIDs of some other applications installed into the system. The downside of this is that the applications whose UIDs we are using, can not use the SIP stack concurrently with the peer-to-peer application. However, using application UIDs of non-SIP-aware programs covers our fix.

The problem has been confirmed with Nokia. We hope this minor glitch will be fixed in the future. For now we have to live with our hack in the code.

XML Parser SyExpat XML parser supports only stream oriented parsing. Parsing would have been easier to implement if the parser supported hierarchical search and access to the XML document. The XML documents used by the application contain a set of predefined fields. Thus, it would be easier to search for these fields and extract needed information than to parse the whole document in

the stream format and to maintain extra state information about where we are inside the document. However, SyExpat was the only free XML parser available for Symbian so its usage — despite these minor obstacles — was justified.

5.3 Peer-to-Peer Server

The super-peer software is being implemented concurrently with the mobile peer-to-peer client. The super-peer will be implemented as an Application Server to provide full integration with IMS networks. While the super-peer software is being implemented we use a simple Python script to act as the super-peer. The Python script does not form pure peer-to-peer networks with other super-peers, but this is not a problem because the peer-to-peer client does not see any difference between the hybrid and centralized architectures; the super-peer abstracts all differences away.

The super-peer will maintain information about shared files; this information includes file names, hashes, and sizes. Also, support for clustering will be implemented in the future. Clustering will allow separating different files to separate clusters. Access to individual clusters can be limited or open to all. Clustering allows different groups of people to share content between each other without having to make the information accessible for all.

The super-peer interacts with the clients during searches and file list updates. File searches are initiated by some client sending an INVITE request including the search string to the super-peer. The super-peer replies with a 606 Not Acceptable message. This reply has the search results that contain information about the matching files and the peers having those files.

5.4 TCP-Relay

A simple TCP relay was used to transfer files between two mobile terminals since direct connections between mobile terminals were impossible because of a mobile operator firewall. The SIP messages were routed via the SIP proxy, so there were

no problems for them.

5.5 Use Case

To use the peer-to-peer file transfer application one needs first to open GPRS/UMTS connection and to create a PDP context to an access point which provides connectivity to the super-peer. Next, one needs to perform the SIP registration, which is followed by uploading the file list to the super-peer. Now, the user is able to search and download files from the other peers. The file search and download signaling is shown in figure 5.5.

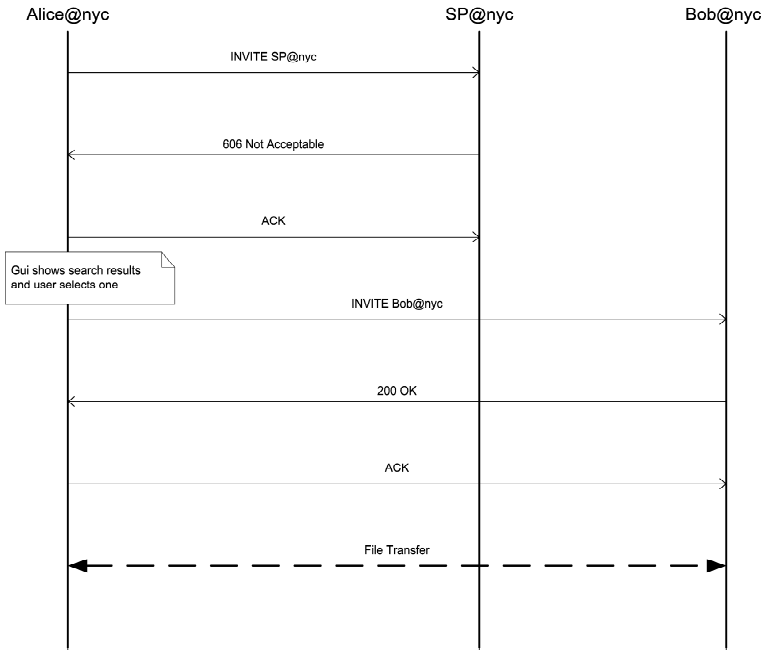

Figure 5.5: Signaling flow for a file search and download

Chapter 6

Measurements

Overall performance of the mobile peer-to-peer application depends mainly on the network performance. The performance of the SIP protocol and the TCP based content transfer protocol is influenced by the packet loss and the delay in the network. Mobile networks usually have varying delays and some packet loss on the air interface. This is due to dynamically changing signal quality [21].

This chapter presents performance measurements done on the application. Signaling delays and download bandwidths were measured, as well as the average signaling message sizes. Next, we discuss shortly the actual software performance, like the memory consumption of the application and the SIP stack. For last, we present discussion how the user perceives the software performance. The results attained in this chapter are later used to provide answers to the initial research question about the feasibility of the SIP protocol to mobile peer-to-peer use.

6.1 Measurement Setup and Restrictions

All measurements were performed using Nokia 6680 mobile phones. The phones were connected to a 3G/WCDMA network during all tests. The network provided 384 kbit/s downstream and 128 kbit/s upstream bandwidth to the mobile phones. It must be noted that we had no tools to monitor the signal quality and other

link parameters. Thus, these measurements should be only used as guide when evaluating the total performance of the application.

Unfortunately, at the time of the writing this thesis, none of Finnish mobile operators provided direct IP-level connectivity between mobile phones. Because of this we had to route all mobile to mobile traffic via a node located in the fixed network. This node was running the Linux operating system, and it was connected to the Internet via an ADSL modem. The node was running a SIP proxy/register, the super-peer and the TCP-relay. The TCP-relay enabled file transfers between the mobile nodes. Instead of opening a TCP-connection directly between each other, the mobile terminals connected to the TCP-relay for file transfers.

During measurements, traffic was captured using *tcpdump* and *Ethereal* packet capturing software. Delay measurements were done with the help of a standard *ping* utility. The results were extracted from the logs produced by the packet capturing programs. Bandwidth measurements were done using the mobile peer-to-peer application for content transfers and measuring the bandwidth in the TCP-relay.

6.2 Network Performance

Simple delay and bandwidth measurements were done to find out how the mobile peer-to-peer application handles searches and content transfers. Also the SIP message sizes were measured from the network captures. Average sizes of different messages can be seen in table 6.1. Measured message sizes include the size of the UDP datagram.

6.2.1 Message Sizes

The message sizes were extracted from the packet captures. It must be noted that because the SIP is a text based protocol and because the messages include many variable length parameters, sizes for the requests and replies vary. The message sizes shown in table 6.1 are averages from the captured messages and are rounded

to nearest ten-byte-boundary.

Table 6.1: SIP message sizes in bytes

Action	**Request**	**Size**	**Reply**	**Size**
Register	REGISTER	390	200 OK	320
Search	INVITE	450–500[1]	100 Trying	250
			606 Not Acceptable	390–1400[2]
	ACK	340		
File list update	MESSAGE	470–1400[3]	200 OK	270
Download	INVITE	560	100 Trying	250
			180 Ringing	310
			200 OK	310
	ACK	410		
De-register	REGISTER	400	200 OK	270

Size of a message consists mostly of the SIP headers. Search replies and file list updates are exceptions to this rule. These messages contain large XML payloads with file listings. If searches or file list updates are frequent it might be useful to use some other kind of encoding for these messages. For example type-length-value encoding could be used instead of XML in these messages to reduce message sizes.

6.2.2 Network Delays

Delay measurements were done using the *ping* utility program running on a PC. ICMP echo requests and replies were sent with different payload sizes and one-way delays for these messages were measured. The mobile phone was attached to the PC with a USB cable during the measurements and it served as a modem.

[1] Search size varies with the number search arguments
[2] Reply size varies with the number of search hits
[3] File list update size varies with the number of added/removed files

One way delays were measured by pinging PC's second network interface, which was connected to the ADSL modem, with the mobile interface. This way it was possible to log the sending time of the ICMP request on the mobile interface and the receiving time on the ADSL interface with microsecond precision on the same computer.

Delays measured from the mobile terminal to the super-peer can be seen in figure 6.1 and delays from the super-peer to the mobile terminal in figure 6.2. Packet loss was zero for all delay measurements.

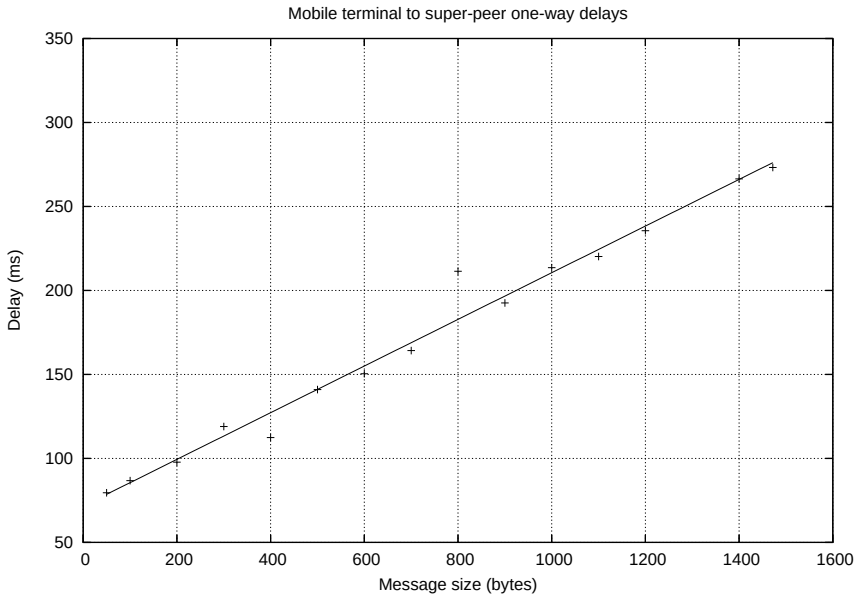

Figure 6.1: One-way delays from the mobile terminal to the super-peer

Figure 6.3 shows round trip times from the ADSL modem to the mobile gateway. We can approximate the one-way delay being about half of the round trip time in both directions for the ADSL connection. Using these results we can approximate the delay introduced solely in the mobile network; e.g. for a 1000 byte packet, we can approximate the ADSL one-way delay being around 13 milliseconds, thus about 200 milliseconds of the total 215 millisecond delay on the MT->SP path is due to the mobile network.

Using these measurement results, we can calculate how long searches take. If

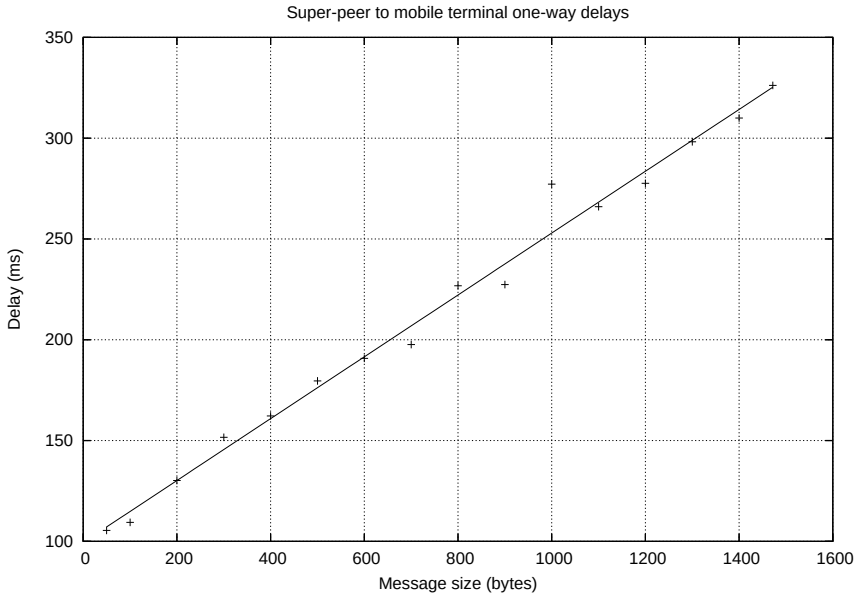

Figure 6.2: One-way delays from the super-peer to the mobile terminal

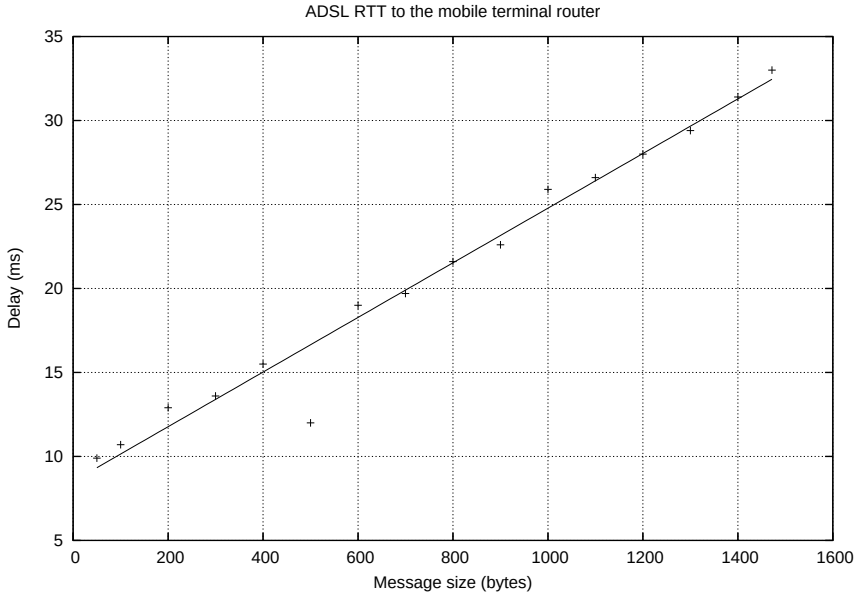

Figure 6.3: Round trip time from the ADSL modem to the router providing access to the mobile terminal (i.e. the mobile gateway)

average size of a search message is 500 bytes, it takes about 140 ms for a message of this size to get to the super-peer. The SIP proxy will reply with 250 byte Trying, which takes less than 150 ms to transfer, and finally the super-peer replies with 606 Not Acceptable message which is around 1400 bytes at maximum, resulting in 310 ms delay. Finally the mobile terminal sends an ACK to the SP, size of 340 bytes, time to transfer 120 ms. This all takes around 700 ms (in case that 100 Trying and 606 Not Acceptable do not overlap). Considering some processing delays in the super-peer and varying delays in the mobile network, the total search delay is usually below one second.

After a successful search, the user might start a file transfer. Before the actual TCP file transfer connection is opened, the download must be initialized with SIP signaling, as shown in figure 5.5. This signaling sequence includes the INVITE request with the SDP payload, then the 100 Trying from the proxy, the 180 Ringing and the 200 OK from the super-peer, and last an ACK from the client. Together these all messages should take less than 1.5 seconds to transfer, in reality the delay is usually even shorter because some of these messages are transmitted back-to-back, and thus their transmission overlaps in time; namely the 100 Trying, 180 Ringing, and 200 OK replies.

We can see that the delay measurements agree with the numbers presented initially in table 4.1. The one-way delays increase with the increasing packet sizes, so the smaller the message is, the quicker it goes through the network. We can speculate that the delays would be even smaller when using more advanced access networks such as HSDPA and WLAN.

6.2.3 Network Bandwidth

File transfer speeds were measured in an office environment. Transfer speeds were measured with varying file sizes. The results are shown in table 6.2. Times do not contain signaling delays — only actual file transfer times were measured. However, as it can be seen when comparing results to those in figures 6.2 and 6.1, for files over 50 kilobytes in size the file transfer time starts to dominate over the signaling delays. Table 6.2 also includes average download rates calculated over

the average download rates of individual tests.

Table 6.2: File transfer speeds for different file sizes

File size	Transmit time	Average download rate	Content example
1 kB	< 1 sec	10.3 kB/s	vCard
10 kB	< 1.5 sec	11.6 kB/s	E-mail
50 kB	3.5 – 6.5 sec	11.3 kB/s	Low-quality image
100 kB	7 – 8.5 sec	13.3 kB/s	Medium-quality image
500 kB	46–48 sec	10.4 kB/s	High-quality image
1 MB	102–104 sec	10.0 kB/s	MP3 / e-Book
5 MB	525–528 sec	9.7 kB/s	MP3 / Video clip

As it can be seen from the results, the download rates achieved in the 3G network are higher than those with a single channel ISDN connection. These rates are more than adequate for small file transfers. Small files download instantly, while larger ones can be downloaded in the background while the user is spending his time in other tasks. The download rate of the client is limited by the upload rate of the peer who holds the content. In this case, the upload rate in the network was limited to 128 kbit/s.

The results also suggest that some kind of phone to phone audio streaming might be feasible in the future, e.g. ability to stream MP3-music clips straight from another mobile phone.

For curiosity, transfer rates were also tested when on move, on the bus, and when walking in an urban environment. Transfer rates in vehicular use averaged to 7 kB/s; in pedestrian use, transfer rates averaged around 9.5kB/s, just below of those achieved in stationary use. Lower performance in vehicular use is probably due to several handovers during the downloads. These tests were done using one megabyte file size.

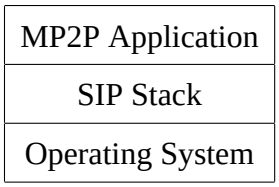

Figure 6.4: The Mobile Peer-to-Peer (MP2P) application uses the SIP stack to interface with the operating system and the network. Measuring the SIP stack performance is not possible from the MP2P application, because the SIP stack is integrated with the operating system.

6.3 Software Performance

The initial objective was also to evaluate the software performance by measuring processing delays in the system. The original idea was to measure how long it takes for the client to process a SIP message from the moment it receives the request to the moment when it sends an automatic reply. Unfortunately, measurements of the SIP processing delays could not be performed. Reason for this was that the SIP stack is integrated on top of the operating system and the network stack. Our application can only see the interface to the SIP stack it uses, not the interface between the SIP stack and the operating system. Thus, we cannot measure the performance of the SIP stack; we have no means of inserting triggers to the operating system and the SIP stack interface. This architecture is pictured in figure 6.4

Memory utilization of the MP2P application was measured with a system monitor application. The memory use of the peer-to-peer application varied between 200 and 350 kilobytes; whereas, the SIP stack and the SIP profile manager together consumed 170 kilobytes of memory.

6.4 User Perceived Performance

From this thesis's point of view, more important than numerical measurements with millisecond resolution is how the user perceives the software performance.

From the user perspective, are there noticeable delays or sluggishness in the application use?

Mainly the user is concerned how quickly he can get some interesting piece of content. This includes searching for the content and then downloading the content. The download time is, of course, affected largely by the file size and available data bearer. However, before the download can be started the search has to be done. The search and download initialization involve some SIP signaling in the background. The signaling performance was measured earlier in this chapter.

Based on measurements and subjective tests done by the project group, the performance is more than adequate. When the user has typed in the search string, the results are represented almost immediately. Considering that web pages take tens of seconds to load on mobile phones, the search speed should be more than enough.

The overall time for file downloading can be divided into five steps:

1. Application startup, PDP context activation and SIP registration delay,
2. Delay of the user entering the search string,
3. Search delay; signaling with the super-peer,
4. Delay of the user selecting the file for download,
5. Download initializing and download delay.

The first step is only needed if the application is not running yet. The delay in the second and fourth steps is solely controlled by the user, so these steps can be left out when evaluating the overall application performance. The delay perceived by the user is thus the result of the first, third, and fifth steps.

The duration of the first step depends on whether some subtask is already performed in advance. The PDP context activation usually takes longest of the three subtasks, the SIP registration takes less than two seconds and the application starts in one second. In the worst case, all these steps should be completed in less than ten seconds.

From the user's perspective, the search is instant, taking around one second at maximum, as it was noted earlier in this chapter.

The last step, which is the file download, takes the most time. The download signaling takes one or two seconds, but the actual content transfer takes time depending on the requested file size. Looking at the measured transmission rates in table 6.2, it can be noted that the duration of the download process starts to dominate when the file size is larger than 100 kilobytes. Below this, the signaling and user performance dictate the total performance.

6.5 Conclusions on Measurements

It must be noted that these measurements only give an indication of what the performance might be. As noted earlier, there was no way to monitor the signal strength, or the quality of the link. Delays also depend on the load level and queuing in the network, and on the load of the super-peer. However, these measurements indicate clearly that the application performance is high enough for a satisfying user experience. Delays perceived by the user are minimal for searches, and the file transfer rates are around ten kilobytes per seconds.

Chapter 7

Conclusions

In this chapter we will revisit the initial objectives and present the results of the study based on the measurements presented in chapter 6. Last, we will discuss how this application could be further developed in the future.

7.1 Objectives Revisited

As a reminder, we revisit the objectives for the mobile peer-to-peer system as stated in the first chapter.

The mobile peer-to-peer application will be used to evaluate the feasibility of the peer-to-peer concept in the mobile domain, considering the following points:

- Does a modern mobile phone have enough resources to run a peer-to-peer application?
- Is the SIP a suitable protocol for peer-to-peer signaling?
- Is the software performance satisfying in user perspective?

By implementing a peer-to-peer client for Series 60 platform, we studied how these objectives could be met.

7.2 Results

The working prototype of the peer-to-peer application proves that the peer-to-peer file sharing can be implemented on a mobile platform, and that performance of the mobile peer-to-peer application is good. The peer-to-peer application ran without problems even when several other applications were started in the background and the delays perceived by the user stayed low.

SIP seems to be a good protocol for peer-to-peer signaling with some remarks. SIP has large signaling overhead when compared to binary based protocols. Considering measurements reported in chapter 6, it can be noted that this overhead is not an issue on modern WCDMA networks. Also, being an integral part of the future mobile networks, SIP is a fine choice for mobile peer-to-peer signaling.

For the application user, the performance of the software is good enough. Searches can be performed in a few seconds, and most of search delay is induced by the user himself while typing in the search strings. Also download rates in modern mobile networks are satisfying for files up to a few megabytes.

All in all, this mobile peer-to-peer application shows that there are no major issues bringing peer-to-peer applications to the mobile domain, so that the users could get access to a vast number of files anywhere, anytime.

7.3 Further Discussion

The system is implemented as a hybrid peer-to-peer architecture, which allows the administrator of the super-peer to have full control on the content available in the network. This fulfills the mobile operators' requirements for the service. On the other hand, all users are not comfortable with the idea of the operators having ability to censor files in the network. These users might be willing to run super-peers of their own — which is also possible with the software presented here. The same users might also be willing to use decentralized mobile peer-to-peer applications — the marketplace is left open for decentralized mobile peer-to-peer applications. However, the performance of a large decentralized peer-

to-peer application is assumed to be quite low even in today's high-speed cellular networks. On the other hand, if the network is small enough (some tens of people), even the pure peer-to-peer architecture might be feasible for mobile use.

Another issue hindering the popularization of mobile peer-to-peer is the current bandwidth pricing in the mobile networks. With current pricing models users have to pay both for the sent and received bytes. This model is not good peer-to-peer-wise, because users have to pay for files they are sharing. Pricing models should be changed so that users could get compensation for the uploaded files. This would result in more shared data available in the network and thus in more downloads.

Asymmetric connections being developed for the future mobile networks are not most suitable for mobile peer-to-peer communication because increased download rates have no effect when everyone's upload bandwidth is restricted much lower. The only way to get decent performance out of these upcoming high speed technologies in mobile peer-to-peer is to build caching into the fixed network so that popular content does not have to be uploaded every time through narrowband mobile links.

7.3.1 Further Research Possibilities

There are many issues relevant to mobile peer-to-peer systems, which we did not address in this thesis; such as, trust, reputation, accountability, security, and interoperability of different peer-to-peer networks.

Also grouping and authentication mechanisms for handling different user groups must be studied more thoroughly to find out how to implement efficient file sharing among closed user groups.

Caching of the popular files into the fixed network like in the mobile eDonkey architecture is also something that has to be looked into. Caching the popular content into the fixed network can greatly reduce the traffic over the air iterface. Caching can be implemented in our architecture for example by creating separate caching nodes that cache the interesting files. Then, super-peers can only list these

cache-peers in the search replies for the popular file queries.

Development of the software will continue during year 2006. Further findings will be published in separate papers.

References

[1] 3GPP. Service requirements for the Internet Protocol (IP) multimedia core network subsystem (IMS); Stage 1. TS 28.228.

[2] Amjad Akkawi, Sibylle Schaller, Oliver Wellnitz, and Lars Wolf. A Mobile Gaming Platform for the IMS. In *SIGCOMM'04 Workshops*, pages 77–84. ACM, 2004.

[3] Peter J. Alexander. Peer-to-Peer File Sharing: The Case of the Music Recording Industry. In *Review of Industrial Organization*, volume 20, pages 151–161. Kluwer Academic Publishers, 2002.

[4] J. Arkko, G. Kuijpers, H. Soliman, J. Loughney, and J. Wiljakka. Internet Protocol Version 6 (IPv6) for Some Second and Third Generation Cellular Hosts. RFC 3316, April 2003.

[5] Nadia Ben Azzouna and Fabrice Guillemin. Experimental Analysis of the Impact of Peer-to-Peer Applications on Traffic in Commercial IP Networks. *European Transactions on Telecommunications: Special Issue on P2P Networking and P2P Services*, November – December 2004.

[6] Balázs Bakos, Gergely Csúcs, Lóránt Farkas, and Jukka K. Nurminen. Peer-to-Peer Protocol Evaluation in Topologies Resembling Wireless Networks. An Experiment with Gnutella Query Engine. In *International Conference on Networks*. IEEE, September – October 2003.

[7] Salman A. Baset and Henning Schulzrinne. An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol. September 2004.

[8] Nicklas Beijar, Marcin Matuszewski, Juuso Lehtinen, and Tuomo Hyyryläinen. Mobile Peer-to-Peer Content Sharing Services in IMS. In *The International Conference on Telecommunication Systems, Modelling and Analysis 2005, ICTSM2005*, Dallas, Texas, USA, November 2005.

[9] Vannevar Bush. As we may think. *Atlantic Monthly*, July 1945.

[10] Gonzalo Camarillo and Miguel A. García-Martín. *The 3G IP Multimedia Subsystem (IMS) - Merging the Internet and the Cellular Worlds*. John Wiley & Sons, Ltd, 2004.

[11] B. Campbell, J. Rosenberg, H. Schulzrinne, C. Huitema, and D. Gurle. Session Initiation Protocol (SIP) Extension for Instant Messaging. RFC 3428 (Proposed Standard), December 2002.

[12] Philippe Charas. Peer-to-Peer Mobile Network Architecture. In *1st International Conference on Peer-to-Peer Computing (P2P 2001)*, pages 55–61. IEEE Computer Society, August 2001.

[13] Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W. Hong. Freenet: A Distributed Anonymous Information Storage and Retrieval System. In H. Federrath, editor, *Anonymity 2000, LNCS 2009*, pages 46–66. 2001.

[14] Yonina Cooper and Hal Berghel. World Wide Web 10 Years Later (Part 1). *I+D Computación*, 1(2), November 2002.

[15] S. Donovan. The SIP INFO Method. RFC 2976 (Proposed Standard), October 2000.

[16] EarthLink. SIPshare: SIP-based P2P Content Sharing Prototype. `http://www.research.earthlink.net/p2p/`. Referenced: 8th October 2005.

[17] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. RFC 2616 (Draft Standard), June 1999. Updated by RFC 2817.

[18] M. Handley and V. Jacobson. SDP: Session Description Protocol. RFC 2327 (Proposed Standard), April 1998. Updated by RFC 3266.

[19] Erkki Harjula, Mika Ylianttila, Jussi Ala-Kurikka, Jukka Riekki, and Jaakko Sauvola. Plug-and-Play Application Platform: Towards Mobile Peer-to-Peer. In *MUM 2004*. ACM, 2004.

[20] Anthony J. Howe. Napster and Gnutella: a Comparison of two Popular Peer-to-Peer Protocols. February 2002.

[21] Douglas Howie, Mika Ylianttila, Erkki Harjula, and Jaakko Sauvola. State-of-the-Art SIP for Mobile Application Supernetworking. In *Nordic Radio Symposium 2004*, 2004.

[22] ITU-T. Packet Based Multimedia Communication Systems. Recommendation H.323 (07/03), July 2003.

[23] Alan B. Johnston. *SIP Understanding the Session Initiation Protocol*. Artech House Publishers, 2nd edition, November 2003.

[24] Thomas Karagiannis, Andre Broido, Nevil Brownlee, kc claffy, and Michalis Faloutsos. Is P2P Dying or just Hiding? In *Globecom 2004*, November – December 2004.

[25] J. Klensin. Simple Mail Transfer Protocol. RFC 2821 (Proposed Standard), April 2001.

[26] Nathaniel Leibowitz, Aviv Bergman, Roy Ben-Shaul, and Aviv Shavit. Are File Swapping Networks Cacheable? Characterizing P2P Traffic. Technical report, Expand Netwoks, Tel-Aviv, Israel, 2002.

[27] Marcin Matuszewski. Economics of Peer-to-Peer – Music Industry Case Study. In Raimo Kantola, editor, *Peer to Peer and SPAM in the Internet*, pages 104–114. 2003.

[28] Microsoft. Microsoft Visual Studio Developer Center. `http://msdn.microsoft.com/vstudio/`. Referenced: 25th December 2005.

[29] Dejan S. Milojicic, Vana Kalogeraki, Rajan Lukose, Kiran Nagaraja, Jim Pruyne, Bruno Richard, Sami Rollins, and Zhichen XU. Peer-to-Peer Computing. Technical report, HP Labs, Rutgers University and University of California at Santa Barbara, 2002.

[30] Aymeric MOIZARD. The partysip SIP proxy server. `http://www.partysip.org/`. Referenced: 25th December 2005.

[31] A. Niemi. Session Initiation Protocol (SIP) Extension for Event State Publication. RFC 3903 (Proposed Standard), October 2004.

[32] Klaus Nieminen. Legal Issues in P2P Systems. In Raimo Kantola, editor, *Peer to Peer and SPAM in the Internet*, pages 115–124. 2003.

[33] Nokia. Series 60 Platform SDK's for Symbian OS, for C++. `http://www.forum.nokia.com/main/0,6566,034-4,00.html`. Referenced: 25th December 2005.

[34] Nokia. SIP Plug-in for Series 60 SDK. `http://www.forum.nokia.com/main/0,6566,034-561,00.html`. Referenced: 25th December 2005.

[35] Jens O. Oberender, Frank-Uwe Andersen, Hermann de Meer, Ivan Dedinski, Tobias Hoßfeld, Cornelia Kappler, Andreas Mäder, and Kurt Tutschku. Enabling Mobile Peer-to-Peer Networking. In *Lecture Notes in Computer Science*, volume 3427, pages 219–234, 2005.

[36] Jens O. Oberender and Hermann de Meer. P2P Replication Revisited: Mobile Infrastructures. In *KiVS Kurzbeiträge und Workshop*, pages 211–214, 2005.

[37] Lakshmish Ramaswamy and Ling Liu. Free Riding: A New Challenge to Peer-to-Peer File Sharing Systems. In *Proceedings of the 36th Hawaii International Conference on System Sciences (HICSS'03)*. IEEE Computer Society, 2002.

[38] A. B. Roach. Session Initiation Protocol (SIP)-Specific Event Notification. RFC 3265 (Proposed Standard), June 2002.

[39] J. Rosenberg. The Session Initiation Protocol (SIP) UPDATE Method. RFC 3311 (Proposed Standard), October 2002.

[40] J. Rosenberg. A Session Initiation Protocol (SIP) Event Package for Registrations. RFC 3680 (Proposed Standard), March 2004.

[41] J. Rosenberg and H. Schulzrinne. An Offer/Answer Model with Session Description Protocol (SDP). RFC 3264 (Proposed Standard), June 2002.

[42] J. Rosenberg and H. Schulzrinne. Reliability of Provisional Responses in Session Initiation Protocol (SIP). RFC 3262 (Proposed Standard), June 2002.

[43] J. Rosenberg and H. Schulzrinne. Session Initiation Protocol (SIP): Locating SIP Servers. RFC 3263 (Proposed Standard), June 2002.

[44] J. Rosenberg, H. Schulzrinne, and G. Camarillo. The Stream Control Transmission Protocol (SCTP) as a Transport for the Session Initiation Protocol (SIP). draft-ietf-sip-sctp-06.txt, January 2005.

[45] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. SIP: Session Initiation Protocol. RFC 3261, June 2002.

[46] Stefan Saroui, P. Krishna Gummadi, and Steven D. Gribble. Measurement Study of Peer-to-Peer File Sharing Systems. Technical report, Department of Computer Science & Engineering, University of Washington, 2002.

[47] Rüdiger Schollmeier. A Definition of Peer-to-Peer Networking for the Classification of Peer-to-Peer Architectures and Applications. In *Proceedings of the First International Conference on Peer-to-Peer Computing (P2P'01)*. IEEE Computer Society, 2002.

[48] Henning Schulzrinne and J. Rosenberg. A Comparison of SIP and H.323 for Internet Telephony. In *Proceedings of International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, pages 83–86, 1998.

[49] Henning Schulzrinne and Jonathan Rosenberg. The Session Initiation Protocol: Internet-Centric Signaling. *IEEE Communications Magazine*, pages 134–141, 2000.

[50] Stephen Simon. Peer-to-Peer Network Management in an IBM SNA Network. *IEEE Network Magazine*, 1991.

[51] Anurag Singla and Christopher Rohrs. Ultrapeers: Another Step Towards Gnutella Scalability. December 2001.

[52] Tuomo Sipilä. Session Initiation Protocol in 3G. Technical report, Nokia Research Center, Helsinki, Finland, 2001.

[53] R. Sparks. The Session Initiation Protocol (SIP) Refer Method. RFC 3515 (Proposed Standard), April 2003.

[54] Marie Thilliez, Thierry Delot, Sylvain Lecomte, and Nadia Bennani. Hybrid Peer-to-Peer Model in Proximity Applications. In *Proceedings of the 17th International Conference on Advanced Information Networking and Applications (AINA'03)*. IEEE Computer Society, IEEE, 2003.

[55] Tim Hsin ting Hu, Binh Thai, and Aruna Seneviratne. Supporting Mobile Devices in Gnutella File Sharing Network with Mobile Agents. In *Proceedings of the Eight IEEE International Symposium on Computers and Communications (ISCC'03)*. IEEE Computer Society, IEEE, 2003.

[56] Toddsoftware.com. SyExpat for Symbian. `http://www.toddsoftware.com/`. Referenced: 22nd September 2005.

Appendix A

Signaling Flows

A.1 Registering to the SIP Registrar

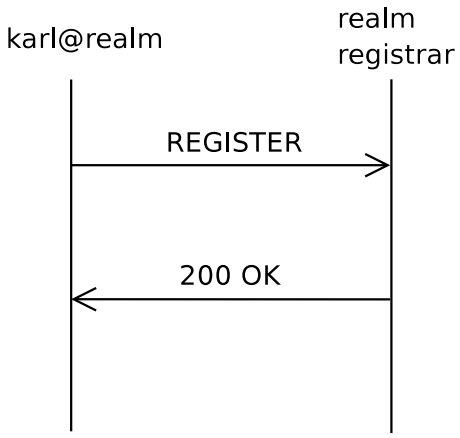

Figure A.1: Registering to the SIP registrar

```
REGISTER sip:192.168.0.2 SIP/2.0
Route: <sip:192.168.0.2;lr>
Via: SIP/2.0/UDP 62.71.210.134:5060;branch=z9hG4bK6SxyAxXNPYPVc
From: sip:karl@realm;tag=EflyAI9Gkc
To: sip:karl@realm
Contact: sip:karl@62.71.210.134;expires=3600
Supported: sec-agree
CSeq: 150646 REGISTER
```

```
Call-ID: ZrByAGG7KI_fX07jNtVH64UIRfxls2
Max-Forwards: 70
Content-Length: 0


SIP/2.0 200 OK
Via: SIP/2.0/UDP 62.71.210.134:5060;branch=z9hG4bK6SxyAxXNPYPVc
From: <sip:karl@realm>;tag=EflyAI9Gkc
To: <sip:karl@realm>;tag=1710802886
Call-ID: ZrByAGG7KI_fX07jNtVH64UIRfxls2
CSeq: 150646 REGISTER
Contact: <sip:karl@62.71.210.134>;expires=3600
Content-Length: 0
```

A.2 File List Update

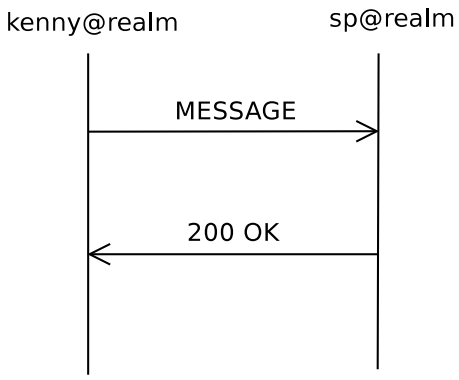

Figure A.2: Updating file list to the superpeer

```
MESSAGE sip:sp@realm SIP/2.0
Route: <sip:192.168.0.2;lr>
Via: SIP/2.0/UDP 62.71.212.186:5060;branch=z9hG4bKLUg-Wh1k431dG
To: sip:sp@realm
From: sip:kenny@realm;tag=Q44-Wqgypj
Supported: sec-agree
CSeq: 150378 MESSAGE
Call-ID: 0aU-WjQ5mPzdwMmjHOpsA198dzURHZ
Max-Forwards: 70
Content-Type: text/xml
Event: mp2p_contentupdate
Content-Length: 219

<?xml version="1.0" standalone = "yes" ?>
<contentupdate>
  <clearall/>
  <add>
    <type>File</type>
    <hash>d21db042fdb9312c</hash>
    <cluster></cluster>
    <name>Kirja.jpg</name>
    <extended type="size">25410</extended>
  </add>
</contentupdate>


SIP/2.0 200 OK
Via: SIP/2.0/UDP 62.71.212.186:5060;branch=z9hG4bKLUg-Wh1k431dG
From: <sip:kenny@realm>;tag=Q44-Wqgypj
To: <sip:sp@realm>
```

```
Call-ID: 0aU-WjQ5mPzdwMmjHOpsA198dzURHZ
CSeq: 150378 MESSAGE
Max-forwards: 70
Content-Length: 0
```

A.3 Searching for a File

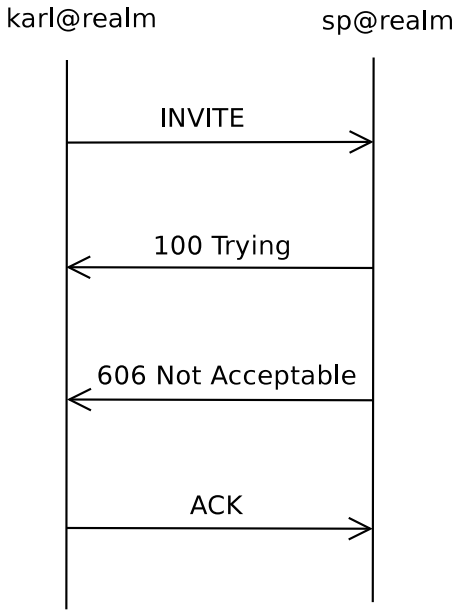

Figure A.3: Commiting search

```
INVITE sip:sp@realm SIP/2.0
Route: <sip:192.168.0.2;lr>
Via: SIP/2.0/UDP 62.71.212.186:5060;branch=z9hG4bKAEg-WsPfzPkUr
From: sip:kenny@realm;tag=XmE-WsITD7
To: sip:sp@realm
Contact: sip:kenny@62.71.212.186
Supported: sec-agree
CSeq: 150379 INVITE
Call-ID: 2rE-WnyV-zpuGcjTxXHPeoiLhz_MTR
Max-Forwards: 70
Content-Type: text/xml
Content-Length: 74

<?xml version="1.0" standalone = "yes" ?>
<request>
  <name>J</name>
</request>


SIP/2.0 100 Trying
Via: SIP/2.0/UDP 62.71.212.186:5060;branch=z9hG4bKAEg-WsPfzPkUr
From: <sip:kenny@realm>;tag=XmE-WsITD7
```

```
To: <sip:sp@realm>
Call-ID: 2rE-WnyV-zpuGcjTxXHPeoiLhz_MTR
CSeq: 150379 INVITE
Content-Length: 0

SIP/2.0 606 Not Acceptable
Via: SIP/2.0/UDP 62.71.212.186:5060;branch=z9hG4bKAEg-WsPfzPkUr
From: <sip:kenny@realm>;tag=XmE-WsITD7
To: <sip:sp@realm>
Call-ID: 2rE-WnyV-zpuGcjTxXHPeoiLhz_MTR
CSeq: 150379 INVITE
Max-forwards: 70
Content-Type: text/xml
Content-Length:   652

<?xml version="1.0" encoding="ISO-8859-1"?>
<reply status="OK">
  <content type="File">
    <hash>69c8e045cc452233</hash>
    <name>Mugi.jpg</name>
    <extended type="size">22751</extended>
    <cluster></cluster>
    <location>
      <username>sip:karl@realm</username>
      <ipaddress>127.0.1.1</ipaddress>
    </location>
  </content>
  <content type="File">
    <hash>8f42eece4b9ffb74</hash>
    <name>16112005.jpg</name>
    <extended type="size">29875</extended>
    <cluster></cluster>
    <location>
      <username>sip:karl@realm</username>
      <ipaddress>127.0.1.1</ipaddress>
    </location>
  </content>
</reply>

ACK sip:sp@realm SIP/2.0
Via: SIP/2.0/UDP 62.71.212.186:5060;branch=z9hG4bKAEg-WsPfzPkUr
Route: <sip:192.168.0.2;lr>
From: sip:kenny@realm;tag=XmE-WsITD7
To: sip:sp@realm
Supported: sec-agree
Call-ID: 2rE-WnyV-zpuGcjTxXHPeoiLhz_MTR
CSeq: 150379 ACK
Max-Forwards: 70
Content-Length: 0
```

A.4 Starting a Download

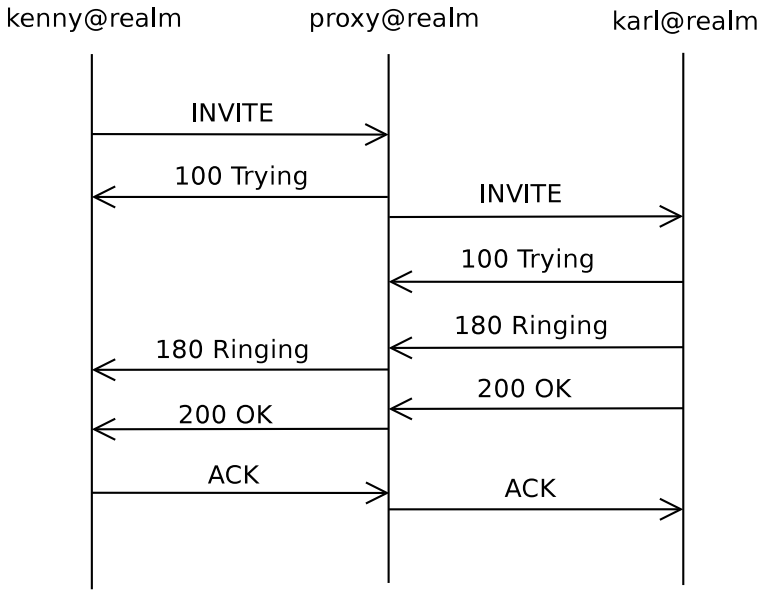

Figure A.4: Starting file download

```
INVITE sip:karl@mp2p.selfip.com SIP/2.0
Route: <sip:mp2p.selfip.com;lr>
Via: SIP/2.0/UDP 62.71.214.87:5060;branch=z9hG4bKssYHAIIIxViYM
From: sip:kenny@mp2p.selfip.com;tag=SJIHAKpZP9
To: sip:karl@mp2p.selfip.com
Contact: sip:kenny@62.71.214.87
Supported: sec-agree
CSeq: 3777 INVITE
Call-ID: 1W4HADZqYFsswpghHSyB6CsaXBBP53
Max-Forwards: 70
Content-Type: application/sdp
Content-Length: 151

v=0
o=kenny 7426438806558691742 7426438806558691743 IN IP4 10.10.10.10
s=69c8e045cc452233
c=IN IP4 10.10.10.10
t=0 0
m=application 1234 TCP MP2P


SIP/2.0 100 Trying
Via: SIP/2.0/UDP 62.71.214.87:5060;branch=z9hG4bKssYHAIIIxViYM
```

```
From: <sip:kenny@mp2p.selfip.com>;tag=SJIHAKpZP9
To: <sip:karl@mp2p.selfip.com>
Call-ID: 1W4HADZqYFsswpghHSyB6CsaXBBP53
CSeq: 3777 INVITE
Content-Length: 0


INVITE sip:karl@62.71.208.198 SIP/2.0
Via: SIP/2.0/UDP 84.249.8.247:5060;branch=z9hG4bKfbed2cf2bef8ed3570354ae6e93c964f8.0
Via: SIP/2.0/UDP 62.71.214.87:5060;branch=z9hG4bKssYHAIIIxViYM
From: <sip:kenny@mp2p.selfip.com>;tag=SJIHAKpZP9
To: <sip:karl@mp2p.selfip.com>
Call-ID: 1W4HADZqYFsswpghHSyB6CsaXBBP53
CSeq: 3777 INVITE
Contact: <sip:kenny@62.71.214.87>
Supported: sec-agree
Max-forwards: 69
Content-Type: application/sdp
Content-Length:   151

v=0
o=kenny 7426438806558691742 7426438806558691743 IN IP4 10.10.10.10
s=69c8e045cc452233
c=IN IP4 10.10.10.10
t=0 0
m=application 1234 TCP MP2P


SIP/2.0 100 Trying
Via: SIP/2.0/UDP 84.249.8.247:5060;branch=z9hG4bKfbed2cf2bef8ed3570354ae6e93c964f8.0,
SIP/2.0/UDP 62.71.214.87:5060;branch=z9hG4bKssYHAIIIxViYM
To: sip:karl@mp2p.selfip.com
From: sip:kenny@mp2p.selfip.com;tag=SJIHAKpZP9
Call-ID: 1W4HADZqYFsswpghHSyB6CsaXBBP53
CSeq: 3777 INVITE
Content-Length: 0


SIP/2.0 180 Ringing
Via: SIP/2.0/UDP 84.249.8.247:5060;branch=z9hG4bKfbed2cf2bef8ed3570354ae6e93c964f8.0,
SIP/2.0/UDP 62.71.214.87:5060;branch=z9hG4bKssYHAIIIxViYM
From: sip:kenny@mp2p.selfip.com;tag=SJIHAKpZP9
To: sip:karl@mp2p.selfip.com;tag=TExNC681xyj3R964
Contact: sip:karl@62.71.208.198
Call-ID: 1W4HADZqYFsswpghHSyB6CsaXBBP53
CSeq: 3777 INVITE
Content-Length: 0


SIP/2.0 180 Ringing
Via: SIP/2.0/UDP 62.71.214.87:5060;branch=z9hG4bKssYHAIIIxViYM
From: <sip:kenny@mp2p.selfip.com>;tag=SJIHAKpZP9
```

```
To: <sip:karl@mp2p.selfip.com>;tag=TExNC681xyj3R964
Call-ID: 1W4HADZqYFsswpghHSyB6CsaXBBP53
CSeq: 3777 INVITE
Contact: <sip:karl@62.71.208.198>
Content-Length: 0


SIP/2.0 200 OK
Via: SIP/2.0/UDP 84.249.8.247:5060;branch=z9hG4bKfbed2cf2bef8ed3570354a
e6e93c964f8.0,SIP/2.0/UDP 62.71.214.87:5060;branch=z9hG4bKssYHAIIIxViYM
To: sip:karl@mp2p.selfip.com;tag=TExNC681xyj3R964
Contact: sip:karl@62.71.208.198
From: sip:kenny@mp2p.selfip.com;tag=SJIHAKpZP9
Call-ID: 1W4HADZqYFsswpghHSyB6CsaXBBP53
CSeq: 3777 INVITE
Content-Length: 0


SIP/2.0 200 OK
Via: SIP/2.0/UDP 62.71.214.87:5060;branch=z9hG4bKssYHAIIIxViYM
From: <sip:kenny@mp2p.selfip.com>;tag=SJIHAKpZP9
To: <sip:karl@mp2p.selfip.com>;tag=TExNC681xyj3R964
Call-ID: 1W4HADZqYFsswpghHSyB6CsaXBBP53
CSeq: 3777 INVITE
Contact: <sip:karl@62.71.208.198>
Content-Length: 0


ACK sip:karl@62.71.208.198 SIP/2.0
Route: <sip:mp2p.selfip.com;lr>
Via: SIP/2.0/UDP 62.71.214.87:5060;branch=z9hG4bK_HkY0zKV4tWPD
To: sip:karl@mp2p.selfip.com;tag=TExNC681xyj3R964
From: sip:kenny@mp2p.selfip.com;tag=SJIHAKpZP9
Supported: sec-agree
Call-ID: 1W4HADZqYFsswpghHSyB6CsaXBBP53
CSeq: 3777 ACK
Max-Forwards: 70
Content-Length: 0


ACK sip:karl@62.71.208.198 SIP/2.0
Via: SIP/2.0/UDP 84.249.8.247:5060;branch=z9hG4bKle6e73de3d298aac70867de5fde62e8f5
Via: SIP/2.0/UDP 62.71.214.87:5060;branch=z9hG4bK_HkY0zKV4tWPD
From: <sip:kenny@mp2p.selfip.com>;tag=SJIHAKpZP9
To: <sip:karl@mp2p.selfip.com>;tag=TExNC681xyj3R964
Call-ID: 1W4HADZqYFsswpghHSyB6CsaXBBP53
CSeq: 3777 ACK
Supported: sec-agree
Max-forwards: 69
Content-Length: 0
```

A.5 Deregistering from the SIP Registrar

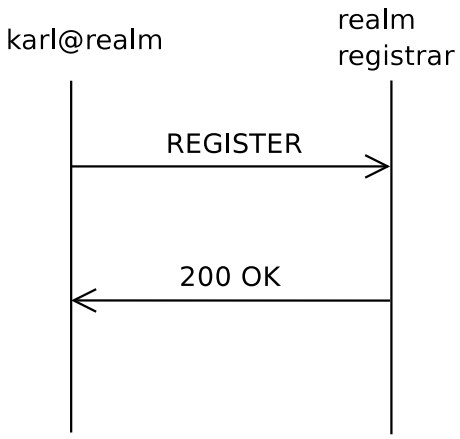

Figure A.5: Deregistering from the SIP registrar

```
REGISTER sip:mp2p.selfip.com SIP/2.0
Route: <sip:mp2p.selfip.com;lr>
Via: SIP/2.0/UDP 62.71.208.198:5060;branch=z9hG4bKkoihx5nEIjVIh
Contact: sip:karl@62.71.208.198;expires=0
To: sip:karl@mp2p.selfip.com;tag=835845152
From: sip:karl@mp2p.selfip.com;tag=SwehxeZsKU
Supported: sec-agree
Call-ID: OAGhx8uNkoQC60njEj8pXgBbpo0EGt
CSeq: 5510 REGISTER
Max-Forwards: 70
Content-Length: 0


SIP/2.0 200 OK
Via: SIP/2.0/UDP 62.71.208.198:5060;branch=z9hG4bKkoihx5nEIjVIh
From: <sip:karl@mp2p.selfip.com>;tag=SwehxeZsKU
To: <sip:karl@mp2p.selfip.com>;tag=835845152
Call-ID: OAGhx8uNkoQC60njEj8pXgBbpo0EGt
CSeq: 5510 REGISTER
Content-Length: 0
```

Appendix B

Document Type Definitions

B.1 File List Update

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!ELEMENT contentupdate (clearall?, (add|remove)*)>
<!ELEMENT clearall EMPTY>
<!ELEMENT add (type, hash, name, cluster?, extended?)>
<!ELEMENT remove (hash)>
<!ELEMENT type (#PCDATA)>
<!ELEMENT hash (#PCDATA)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT cluster (#PCDATA)>
<!ELEMENT extended (#PCDATA)>
<!ATTLIST extended type (size|bitrate) #REQUIRED>
```

B.2 Search Request

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!ELEMENT request (type?, hash?, cluster?, name?, extended?)>
<!ELEMENT type (#PCDATA)>
<!ELEMENT hash (#PCDATA)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT cluster (#PCDATA)>
<!ELEMENT extended (#PCDATA)>
<!ATTLIST extended type (size|bitrate) #REQUIRED>
```

B.3 Search Reply

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!ELEMENT reply (content*)>
<!ELEMENT content (hash, name, cluster?, extended?, location+)>
<!ELEMENT hash (#PCDATA)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT cluster (#PCDATA)>
<!ELEMENT extended (#PCDATA)>
<!ELEMENT location (username, ipaddress?, phonenumber?)>
<!ELEMENT username (#PCDATA)>
<!ELEMENT ipaddress (#PCDATA)>
<!ELEMENT phonenumber (#PCDATA)>
<!ATTLIST reply status (ok|error) #REQUIRED>
<!ATTLIST content type (file|stream|chat) #REQUIRED>
<!ATTLIST extended type (size|bitrate) #REQUIRED>
```