Helsinki University of Technology
Faculty of Electronics, Communications and Automation
Department of Communications and Networking

Juuso Aleksi Lehtinen

Mobile Peer-to-Peer over Session Initiation Protocol

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

August 4, 2008
Juuso Lehtinen

Supervisor: Professor Raimo Kantola

Helsinki University of Technology

Abstract of the Licentiate's Thesis

Author:	Juuso Aleksi Lehtinen		
Name of the Thesis:	Mobile Peer-to-Peer over Session Initiation Protocol		
Date:	August 4, 2008	**Number of pages:**	ix + 75
Faculty:	Faculty of Electronics, Communications and Automation	**Professorship:**	S-38
Department:	Department of Communications and Networking		
Supervisor:	Professor Raimo Kantola		

This work continues on my Master's Thesis work done between July 2005 and January 2006. In my Master's Thesis, we presented how a mobile peer-to-peer file-sharing application can be implemented using the Session Initiation Protocol (SIP) as the underlying signaling protocol.

The main objective of this thesis is to evaluate what kind of special requirements mobile environment poses for peer-to-peer application design, and present how peer-to-peer based services can be efficiently realized in next-generation mobile networks by using SIP with some enhancements as the peer-to-peer signaling protocol.

This thesis is divided into two parts. In the first part, we present different peer-to-peer architectures and search algorithms, and evaluate their suitability for mobile use. We also review some mobile peer-to-peer middleware and file-sharing applications. Then, in the second part, we present our hybrid mobile peer-to-peer architecture consisting of a Symbian based mobile client and a SIP Application Server based super-peer.

Key findings of this thesis are that the mobile peer-to-peer application based on SIP signaling and hybrid peer-to-peer architecture is suitable for mobile use as it minimizes overhead in mobile nodes and allows mobile operator to have control on its users in multi-operator environment. Also, the performance of the application satisfies user requirements.

Keywords: mobile peer-to-peer, session initiation protocol, peer-to-peer over session initiation protocol signaling

Teknillinen Korkeakoulu

Lisensiaattityön Tiivistelmä

Tekijä:	Juuso Aleksi Lehtinen		
Työn nimi:	Istunnon aloitusprotokollaan pohjautuvat mobiilivertaisverkot		
Päivämäärä:	4.8.2008	**Sivuja:**	ix + 75
Tiedekunta:	Elektroniikan, tietoliikenteen ja automaation tiedekunta	**Professuuri:**	S-38
Laitos:	Tietoliikenne- ja tietoverkkotekniikan laitos		
Työn valvoja:	Professori Raimo Kantola		

Tämä työ on jatkoa diplomityölleni, joka tehtiin Heinäkuu 2005 – Tammikuu 2006 välisenä aikana. Diplomityössäni esitimme kuinka mobiilivertaisverkkosovellus voidaan toteuttaa käyttäen Session Initiation Protocol (SIP) protokollaa allaolevana signalointiprotokollana.

Tämän työn päätavoite on selvittää, mitä erikoisvaatimuksia mobiiliympäristö vertaisverkkosovelluksen suunnittelulle asettaa sekä kuinka vertaisverkkopalveluita voidaan tehokkaasti toteuttaa seuraavan sukupolven mobiiliverkoissa käyttämällä laajennettua SIP protokollaa sovelluksen merkinantoprotokollana.

Tämä työ on jaettu kahteen osaan. Ensimmäisessa osassa käsittelemme eri vertaisverkkoarkkitehtuureja ja hakualgoritmeja, sekä arvioimme näiden sopivuutta mobiilikäyttöön. Käymme myös läpi joitain mobiilivertaisverkkotiedostojako-ohjelmia sekä middleware-alustoja. Työn toisessa osassa esittelemme oman mobiilivertaisverkko-arkkitehtuurimme, joka koostuu Symbian mobiilisovelluksesta sekä SIP sovelluspalvelin super-peer solmusta.

Tutkimuksen päälöydökset ovat seuraavat: SIP protokollaa käyttävä hybridi-vertaisverkko-sovellus toimii hyvin matkapuhelinympäristössä, koska se minimoi puhelimeen kohdistuvan rasituksen ja tekee mahdolliseksi matkapuhelinoperaattorin hallita sovelluksen käyttäjiä myöskin monioperaattoriympäristössä. Tämän lisäksi ohjelmiston suorituskyky täyttää käytäjien sille asettamat vaatimukset.

Avainsanat: mobiilivertaisverkot, session initiation protocol, istunnon aloitusprotokollaan pohjautuvat mobiilivertaisverkot

“Every accomplishment starts with the decision to try.”

Unknown source

Acknowledgements

Most of the research described in this thesis has been done as part of a larger mobile peer-to-peer research project in Helsinki University of Technology Networking Laboratory during time between June 2005 and January 2007.

I would like to thank my supervisor and the first examiner for the thesis, Professor Raimo Kantola, for providing invaluable support and guidance during my post-graduate studies and research. I would also like to thank Professor Jukka Manner for being the second examiner for the thesis.

I would also like to thank my past colleagues and co-authors, Nicklas Beijar and Tuomo Hyyryläinen, from Networking Laboratory, as well as research collaborators and co-authors, Marcin Matuszewski and Miguel A. Garcia-Martin, from Nokia Research Center.

Finally, I would like to thank my family and my friends for the support and encouragement they have provided me during the ups and downs of my studies.

August 4, 2008

Juuso Lehtinen

Contents

Abbreviations

3G	Third Generation
3GPP	Third Generation Partnership Project
AOR	Address of Record
AS	Application Server
AuC	Authentication Center
BFS	Breadth-First Search
CPU	Central Processing Unit
CSCF	Call Session Control Function
DC	Direct Connect
DHT	Distributed Hash Table
EVDO	Evolution Data-Only
GUI	Graphical User Interface
HSPA	High-Speed Packet Access
HTTP	Hypertext Transfer Protocol
ID	Identifier
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IM	Instant Messaging
IMS	IP Multimedia Subsystem
IP	Internet Protocol
IRTF	Internet Research Task Force
ITU	International Telecommunication Union

ITU-T	ITU Telecommunication Standardization Sector
JXTA	Juxtapose
MIME	Multipurpose Internet Mail Extensions
MMS	Multimedia Message Service
MP2P	Mobile Peer-to-Peer
MSRP	Message Session Relay Protocol
NAT	Network Address Translation
P2P	Peer-to-Peer
PC	Personal Computer
P-CSCF	Proxy Call Session Control Function
PDP	Packet Data Protocol
PKI	Public Key Infrastructure
RFC	Request for Comments
S/MIME	Secure / Multipurpose Internet Mail Extensions
S-CSCF	Serving Call Session Control Function
SDP	Session Description Protocol
SIMPLE	SIP for Instant Messaging and Presence Leveraging Extensions
SIP	Session Initiation Protocol
SIPPING	Session Initiation Proposal Investigation
SMTP	Simple Mail Transfer Protocol
STP	Spanning Tree Protocol
TLS	Transport Layer Security
TTL	Time-to-live
UA	User Agent
UAC	User Agent Client
UAS	User Agent Server
UE	User Equipment
URI	Universal Resource Identifier
USIM	Universal Subscriber Identity Module
VOD	Video on Demand
VOIP	Voice over Internet Protocol
XML	Extensible Markup Language

Chapter 1 – Introduction

A lot has changed in the world of Internet communications during this decade. A major shift has happened in the traffic patterns of the Internet: Ten years ago, majority of the Internet traffic flows were between personal computers and high-performance web servers, however, today, majority of the Internet traffic is between personal computers in homes, schools, and offices – peer-to-peer networks have established themselves as Internet's major traffic generators [1]. According to recent studies, the share of peer-to-peer traffic is 49% - 83% of all Internet traffic, depending on geographical region [2].

At first, peer-to-peer networks were used only for file-sharing, e.g., Napster [3], Gnutella [3] [4], BitTorrent [5], but as time has passed, many kinds of applications have been built using the peer-to-peer paradigm, e.g., the popular Voice over IP (VOIP) application, Skype [6], or the recently launched Video on Demand (VOD) application, Joost [7]. Growth of peer-to-peer phenomenon has not only created new protocols but also older protocols, traditionally utilized in the client-server paradigm, are now being transformed into peer-to-peer protocols, e.g., Peer-to-Peer Session Initiation Protocol (P2P-SIP) [8].

While peer-to-peer communications has revolutionized the Internet traffic patterns during the past years, there has been a revolution of its own among the mobile phones; mobile phones have gotten close to personal computers in their features and performance. Mobile phones of today have more memory, faster processors, larger screens, and higher network bandwidth than ever before.

Many of the phones sold today are user programmable, meaning that the user can install 3rd party applications of his choice into the phone without consulting with the phone manufacturer.

With the modern mobile phones, people are consuming more media on the go than ever before. The modern mobile phones are as much of personal media players as they are telephones. In addition to being media players, they also function as media recorders, having capabilities similar to standalone audio or video recorders.

1.1 Motivation

The media that people are consuming and producing on their mobile handsets can be currently shared in limited ways; using a cellular service like multimedia message service (MMS), using the Bluetooth connectivity between nearby phones, or using media sharing sites available in the public Internet, or Searching for interesting content in the mobile domain is currently very limited: Popular content can be searched from Internet sites such as *YouTube*, but less popular content that is still valuable for the user, e.g., content created by user's friends or family, is not searchable in any way.

Also, using a mobile web browser for sharing and acquiring media might not be the best tool for the job in the bandwidth limited mobile environment. Sure, if you have one picture to share, it is rather easy to upload it to an image sharing site – but if you have tens or hundreds of pictures that you would like to share with your friends, it is not that convenient to upload all those pictures over a relatively slow radio link – not knowing if any of your friends will ever access any of those pictures. Instead it would be handy if you could share the pictures on demand, as they are requested by your friends.

As we have seen in the fixed Internet, peer-to-peer networks enable easy and efficient way of searching and sharing various types of content. It is a logical step

to bring peer-to-peer communications into the mobile arena, to help people share and search all the mobile content.

When comparing services based on peer-to-peer architecture to those based on the traditional client-server architecture we can observe some of the technical advantages of peer-to-peer architectures. In peer-to-peer architectures, the shared content is available when needed, no uploading to a central server required, and thus only the content that is requested is actually transferred. Users do not have to upload hundreds of pictures to a centralized server, not knowing if anyone will ever access those pictures – thus saving that precious bandwidth. Peer-to-peer architectures also handle the flash crowd phenomenon well when compared to client-server architecture based services, as peer-to-peer architectures naturally distribute the load on several participants of the network [9].

Even though hundreds of different peer-to-peer applications have been deployed in the fixed internet, utilizing tens of peer-to-peer protocols, there have not been many peer-to-peer applications or protocols available in the mobile domain. Some projects have implemented popular peer-to-peer protocols on mobile platforms, e.g., SymTorrent [10], Symella [11], and Mobile eDonkey [12], but none of these projects have really considered the special needs and constraints of mobile domain. Also, most of this work has focused on one service, i.e., on peer-to-peer file-sharing.

Next-generation mobile networks, like Third Generation Partnership Project's (3GPP) IP Multimedia Subsystem (IMS), are largely built onto well known internet protocols, such as Session Initiation Protocol (SIP). These protocols enable users to establish voice and video calls, use presence service, and many other advanced services. These protocols cannot be used for peer-to-peer networking as they are, but with minor modifications, a protocol like SIP, can be engineered to function as the signaling protocol of a peer-to-peer network.

As these protocols are supported by the networks and the terminals natively, it is easier to provide managed peer-to-peer services on IMS protocols than to build the peer-to-peer service framework and the protocols from scratch. IMS protocol suite has protocols ready for the essential peer-to-peer tasks, such as, session initiation, authentication, and accounting.

1.2 Objectives and Scope

This work continues on my Master's Thesis work done between July 2005 and January 2006. In my Master's Thesis [13], we presented how a mobile peer-to-peer file-sharing application can be implemented using the Session Initiation Protocol (SIP) as the underlying signaling protocol. In addition to presenting the implementation, we evaluated the feasibility of the concept by measuring the signaling efficiency and transmission bandwidth available in 3G networks.

The main objective of this thesis is to present how peer-to-peer based services can be efficiently realized in next-generation SIP/IMS networks by reusing their existing protocols as much as possible, and to present some enhancements to these protocols. We also evaluate what kind of special requirements mobile environment poses for peer-to-peer applications and consider those requirements in our application design.

Longer term objective for the research is to develop a peer-to-peer framework over which different kinds of mobile services can be deployed without providing a centralized service architecture in the network. This framework should provide service discovery and service connection services for various overlying applications.

Compared to other mobile peer-to-peer research, we present a unique way of integrating a peer-to-peer network model on top of IMS networks, where SIP is reused as the peer-to-peer signaling protocol, i.e., for uploading resource info from a mobile client to a super-peer, for searching resources in peer-to-peer network, and for initiating resource connection, e.g., file transfer between

mobile peers. We also consider special requirements of mobile environment in our application design.

1.3 Own Contribution

During the research, we authored a journal article, four conference articles, and two post-graduate seminar articles. This thesis presents a coherent picture of the research presented in these articles. This thesis presents our work as it has evolved over time and compares this work to other similar solutions. In addition, this thesis presents a freshened literature review that includes material published after the writing of our articles.

Here we present the contribution of the present author in these publications:

Publication [14]: This paper is independent work of the present author.

Publication [15]: This paper is independent work of the present author.

Publication [16]: This paper is joint work of the authors; the performance measurements and their analysis is independent work of the present author.

Publication [17]: The mobile peer-to-peer client architecture and the implementation part of the paper is joint work of the present author and Tuomo Hyyryläinen.

Publication [18]: The mobile peer-to-peer client architecture and the implementation part of the paper is joint work of the present author and Tuomo Hyyryläinen.

Publication [19]: Ideas behind the signaling schemes presented in this paper are joint work of the authors; the performance measurements and their analysis is independent work of the present author.

Publication [20]: The related work section is independent work of the present author.

1.4 Structure

The structure of this thesis is the following. In Chapter 2, we review common peer-to-peer architectures and different search algorithms used in peer-to-peer networks. In Chapter 3, we discuss special requirements of mobile environment for peer-to-peer applications, and discuss how these requirements affect the choice of peer-to-peer architecture and search algorithm. We also present current state-of-the-art in mobile peer-to-peer application and middleware research. In Chapter 4, we present our mobile peer-to-peer architecture and the key findings of our research on the subject. Finally, in Chapter 5, we provide conclusions and discuss future research possibilities.

Chapter 2 – Peer-to-Peer Architectures and Algorithms

The peer-to-peer paradigm became popular with the file-sharing application Napster. Napster was released in 1999, and it was mainly targeted for sharing music files, even though it allowed sharing of other kinds of files too. Napster was built on a centralized peer-to-peer architecture where a cluster of centralized servers hosted information about the shared files. Allegedly, largely due to the centralized architecture, Napster was shut down due to legal troubles in 2002. Later Napster was reopened as a music store – however, the new Napster was not anymore based on the peer-to-peer paradigm but on the traditional client-server architecture. [21]

Closing of Napster was not the death of peer-to-peer file-sharing. Even before Napster was closed, many other peer-to-peer file-sharing networks appeared, such as Gnutella and Kazaa. Today, Gnutella and Kazaa are in lesser use as BitTorrent has taken their place as the most popular peer-to-peer file-sharing protocol.

The peer-to-peer paradigm has not only been used for file-sharing applications, but also for other applications, such as instant messengers, internet telephony, and video on demand application. Skype is probably the most popular Voice over IP (VOIP) application deployed at large in the Internet, and it is based on semi-centralized peer-to-peer architecture. Joost, a recent video on demand

application, built by the founders of Skype, is also partly based on peer-to-peer paradigm.

The fundamental difference between peer-to-peer and client-server architectures is that in client-server architecture all clients rely on one, centralized, server. All resources are located on this server and the clients request these resources from the server as they need them. The whole network is dependent on the availability of this server – if the server fails, all resources become inaccessible for the clients.

Another problem of the client-server architecture is that all resources must be uploaded to the centralized server so that they are available for the other clients. For example, in case of an image sharing service, the user has to upload all of his images to the centralized server even when he does not know if anyone is going to access those images later. On the other hand, in peer-to-peer architectures the resources are scattered around the network as the peers are hosting the resources themselves. Thus, resources are not uploaded anywhere until they are requested by another peer.

As peer-to-peer communications has gained popularity among the academics, there have been numerous conferences and workshops organized around peer-to-peer and mobile peer-to-peer technologies. Some of the major forums for presenting peer-to-peer research results are:

- IEEE International Conference on Peer-to-Peer Computing [22],
- International workshop on Peer-to-Peer Systems [23],
- IEEE International Workshop on Mobile Peer-to-Peer Computing, organized annually in conjunction with the IEEE International Conference on Pervasive Computing and Communications [24],
- International Workshop on Hot Topics in Peer-to-Peer Systems, organized annually in conjunction with the IEEE International Parallel & Distributed Processing Symposium [25],

- Peer-to-peer oriented workshops, organized in conjunction with the IEEE Consumer Communications and Networking Conference [26], and
- Internet Research Task Force's (IRTF) Peer-to-Peer Research Group (p2prg) [27].

2.1 Architectures

Before we start the review of different peer-to-peer architectures we need a definition for peer-to-peer networking. A good definition is given by Schollmeier [28]:

> *"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 main takeaway from this quote is that the participants of a peer-to-peer network can exchange information directly with each other without passing the information via some centralized entity, and that all the resources in the network are provided by the peers themselves. The peers work simultaneously as Servers and Clients, thus the name, *Servent*.

Depending how the peer-to-peer network topology is organized, the peer-to-peer architectures can be divided into structured and unstructured architectures. Unstructured peer-to-peer architectures can be further divided into centralized, decentralized, and semi-centralized architectures. The main difference between unstructured and structured architectures is that in structured architectures

peers form a defined structure, or a topology, that has to be kept up as nodes join and leave the network. In unstructured peer-to-peer networks, the network is constructed more freely. Milojicic et al. give an extensive introduction to peer-to-peer architectures in [29].

Functionality of all peer-to-peer architectures can be divided into two parts, into the resource search part, and into the part of connecting to the resource. The peer-to-peer architecture defines the logical links between the network peers. These links are used by the peer-to-peer search algorithm for resource location. The second part of peer-to-peer, connecting to resource, is not dependent on the peer-to-peer architecture but happens directly between the peers using direct network layer connectivity without facilitating intermediate nodes.

In this section, we discuss how search works in different peer-to-peer architectures, as the actual connecting to the resource is trivial and nothing special to peer-to-peer. We discuss the traditional client-server architecture and the major peer-to-peer architectures; centralized, decentralized, semi-centralized, and structured peer-to-peer architectures.

2.1.1 Client-Server Architecture

Client-Server architecture is not a peer-to-peer network architecture but it is presented here for reference.

Client-server architecture is the most dominant architecture in the traditional Internet. It is the architecture used between web-servers and browsers, email servers and email clients, etc. In the client-server architecture, a powerful server or a cluster of servers provides a service to many dumb clients. This service can be anything from storing files or databases to remote procedure calls for off-loading complex calculations from clients to the server. In the client-server architecture, each client communicates only with the server, being totally unaware of the other clients served by the same server. Figure 2.1 shows the basic client-server architecture.

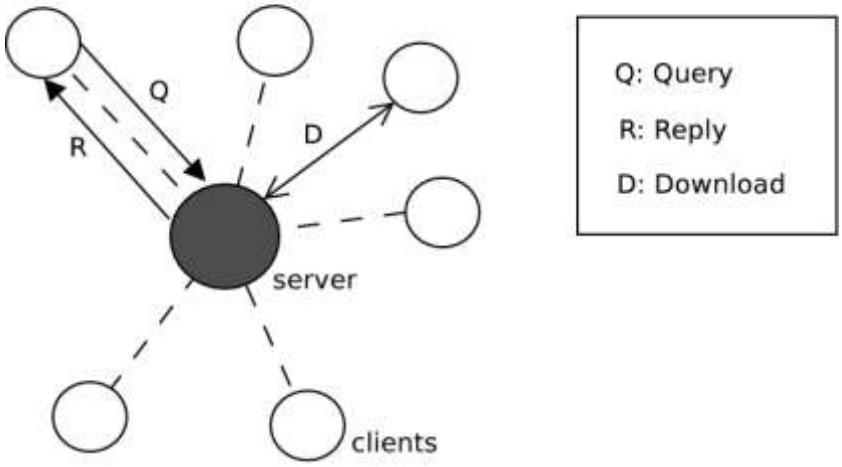


FIGURE 2.1: CLIENT-SERVER ARCHITECTURE

In the client-server architecture, clients send queries to the server, the server processes these queries, and generates appropriate answers – being it an answer to a complex mathematical query, or simply sending a static web-page back to the querying client.

2.1.2 Centralized Architecture

A centralized peer-to-peer architecture resembles the client-server architecture. However, in this architecture clients communicate directly with each other in addition to communicating with a centralized server or a cluster of servers.

In the centralized peer-to-peer architecture, the server works as a centralized index, holding information about the resources or services the clients are hosting. In this architecture, the clients provide information about the resources they are sharing to the server as they join the network. The server will then reply to queries coming from the other clients asking for some resource they need. These replies contain addressing information of the client that has the requested resource. This part of centralized peer-to-peer can be seen being identical to the client-server architecture. Here, the information about resources, i.e., the meta-information, is transferred between peers via the centralized index.

However, after a peer receives reply to its query from the server, it contacts directly the peer providing the needed resource. In this part, the server is no longer part of the communications as the nodes communicate in a peer-to-peer manner. Figure 2.2 shows the basic centralized peer-to-peer architecture.

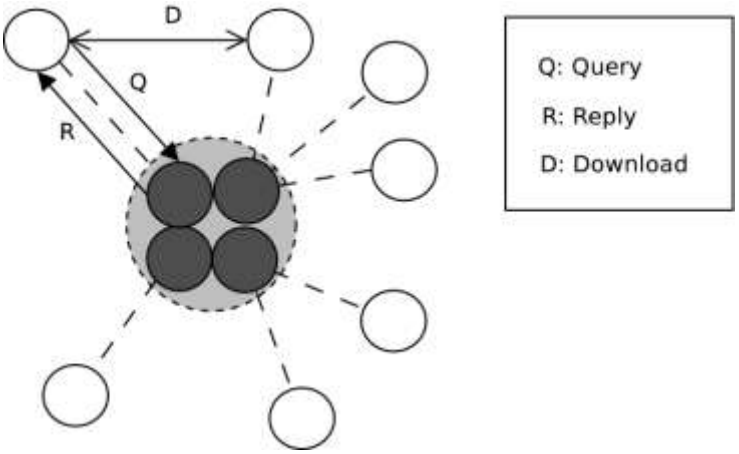


FIGURE 2.2: CENTRALIZED PEER-TO-PEER ARCHITECTURE

There is nothing special concerning the search in the centralized peer-to-peer architecture when comparing it to the client-server architecture. Search is a simple one-hop-query from a client to the server. Only difference lies in the location of the content being searched; whereas in client-server architecture the server hosts both the meta-information about the content and the actual content, in the centralized peer-to-peer architecture the server hosts only the meta-information while the actual content is hosted in the peer nodes.

2.1.3 Decentralized Architecture

In the decentralized peer-to-peer architecture, there is no centralized index. In the decentralized architecture, all peers are equal – peers are connected to each other in an arbitrary fashion, resembling a mesh. Each peer has an index of the resources it is hosting.

In the decentralized architecture, search is routed from peer to peer through multiple hops. Search can be done in numerous ways, the simplest algorithm being flooding search which sends the query message all over the network.

The resource connection, e.g., the file download, is established directly between the endpoint peers so that there are no other nodes in the download path. The basic architecture for decentralized peer-to-peer is presented in Figure 2.3.

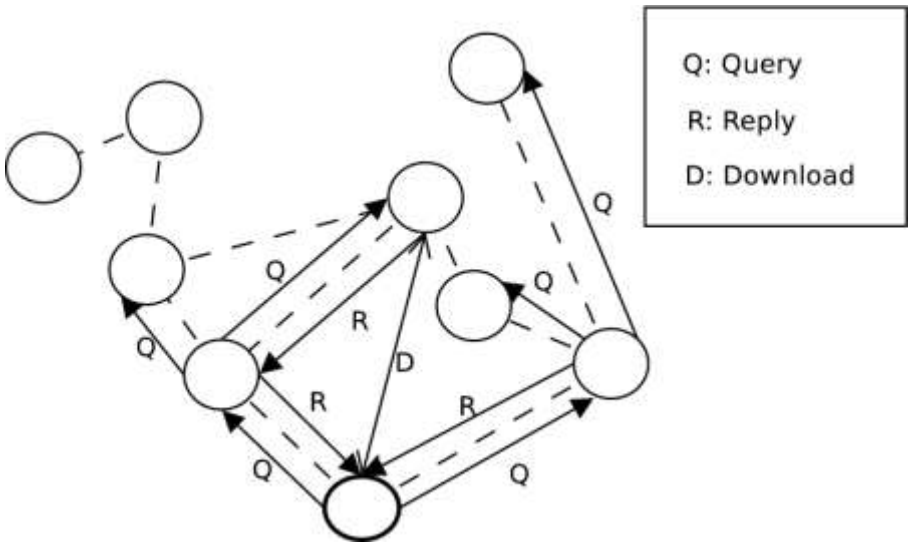


FIGURE 2.3: DECENTRALIZED PEER-TO-PEER ARCHITECTURE

In decentralized peer-to-peer networks, it may be difficult to find the first node to connect to as peers dynamically join and leave the network. In fact, some decentralized peer-to-peer networks might have a bit of centralization, a bootstrapping server, which hosts a list of potential peers in the network to help joiners to find the first peer.

2.1.4 Semi-Centralized Architecture

Semi centralized architecture is a combination of the centralized and decentralized peer-to-peer architectures, thus it is often called a hybrid architecture. In the semi-centralized architecture, there are two kinds of nodes: edge-nodes and super-nodes. The super-nodes are connected to each other in a similar fashion to nodes in the decentralized peer-to-peer architecture. The edge-nodes are connected to the super-nodes in the centralized peer-to-peer fashion. Figure 2.4 presents the semi-centralized peer-to-peer architecture.

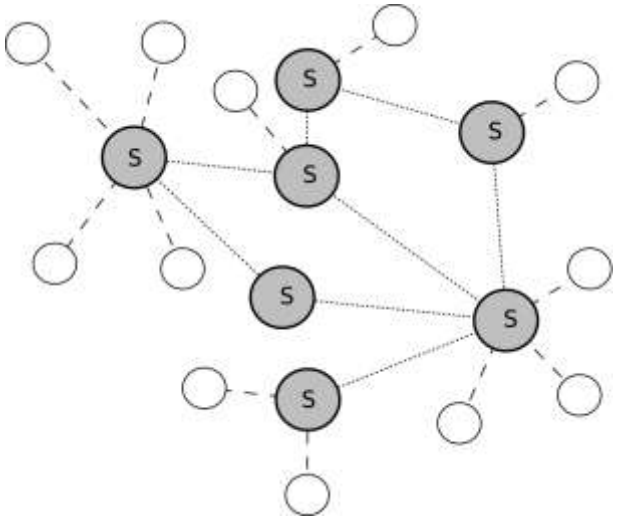


FIGURE 2.4: SEMI-CENTRALIZED PEER-TO-PEER ARCHITECTURE

In the semi-centralized architecture, the super-nodes function as index servers for the edge-nodes. When an edge-node joins the network, it connects to a super-node and uploads the list of its resources to the super-node. When an edge node searches for a resource, it first sends a query to its super-node as a one-hop query. The super-node will then transmit the query further to other super-nodes using a similar algorithm that is used in decentralized peer-to-peer networks. The query does not have to be flooded to edge-nodes because the super-nodes, as the index servers, have total knowledge of the resources available in their edge-nodes. After the reply comes back to the querying edge node, it connects directly to the other edge node hosting the queried resource.

It is important to notice that search is partitioned into two separate layers in the semi-centralized architecture; to the search between the edge-node and the local super-node and to the search between the super-nodes. Whereas, the search between the edge-node and the local super-node is performed in a similar manner to the centralized peer-to-peer architecture, the search between super-nodes is similar to search in the decentralized peer-to-peer architecture.

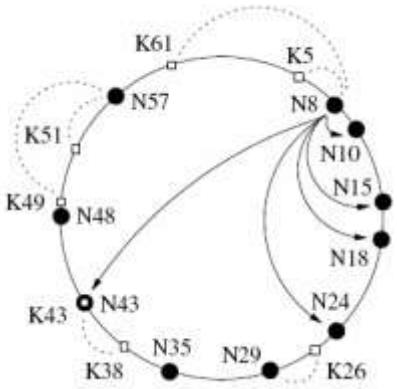
2.1.5 Structured Architectures

Structured peer-to-peer architectures are also known as Distributed Hash Table (DHT) architectures. These architectures have strict structures, e.g., a ring or a hyper-cube. These structures are constantly being updated, so they stay intact as nodes join and leave the network.

In the DHT architecture, every node is given a unique ID or a hash value based on its attributes, e.g., its IP-address. The node ID dictates which part of the hash space the node is responsible for.

When a new resource is added to the network, a hash is calculated for this resource. Then, a link to the resource is stored into the node responsible for the respective part of the hash space. The node stores a link to the resource location, not the resource itself. This way the resource index of the network is distributed deterministically around the network in the Distributed Hash Table.

Figure 2.5 presents the architecture of a popular DHT, Chord. In the figure, the dotted lines indicate which nodes host which keys. Black lines represent the fingers of node N8, i.e., the logical connections to other nodes in the network



Finger Table of N8		
Idx	Target ID	Successor
0	N8 + 1	N10
1	N8 + 2	N10
2	N8 + 4	N15
3	N8 + 8	N18
4	N8 + 16	N24
5	N8 + 32	N43

FIGURE 2.5: CHORD ARCHITECTURE [30]

2.2 Search Algorithms

A common theme among different peer-to-peer architectures is that the actual resource connection (e.g., file download, streaming video, or telephone call)

happens directly between the network peers. What makes peer-to-peer architectures different from each other is the type of search, i.e., how query messages are passed in the network during a search.

Risson and Moors [31] provide an excellent survey of different peer-to-peer architectures and search methods in them. Vanthournout, Deconinck, and Belmans [32] compare resource discovery algorithms in 25 popular protocols.

Next, we review some search algorithms used in peer-to-peer networks. The search algorithms covered are: centralized search, flooding search, modified BFS, iterative deepening, random walks, and structured search. Last, we present short comparison of centralized, flooding, and structured search algorithms.

2.2.1 Centralized Search

The most trivial search method is centralized search. Centralized search is used in the centralized peer-to-peer architecture where a central server holds an index of all resources available in the network. Network peers publish information about the resources they are offering to the central index by directly uploading their resource lists to the server. Other peers can connect to the server and request resources from it. The centralized server then performs search against its index and provides the requesting peer addressing information about the peers having the requested resource.

Because the index is in one place, search in the centralized architecture is fast and comprehensive. All files available in the peer network, i.e., all files published to the centralized index, are checked during the search.

2.2.2 Flooding Search

In a decentralized peer-to-peer network, there is no centralized index of network resources. Instead, every node holds an index of its own resources. To search the network, you have to search the nodes. Depending on how many nodes you

search dictates the actual coverage of your search and the probability of finding the resource you are looking for.

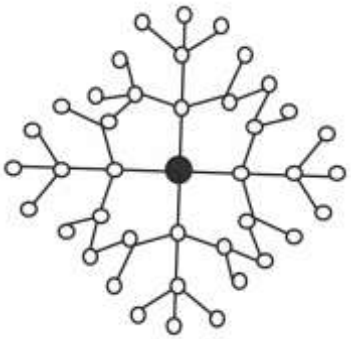


FIGURE 2.6: FLOODING SEARCH [33]

In flooding search – also known as Breadth-First Search (BFS) – a peer requesting a resource sends the request to all its directly connected peers. These peers check their local indexes for matching resources and further send the request towards all the peers they are connected to, except the peer where they got the request in the first hand.

The way the BFS queries propagate in the network is similar to Ethernet broadcast frames. However, where Ethernet networks should not have any loops – thanks to Spanning Tree Protocol (STP) – decentralized peer-to-peer networks often have them. Because of this, every node forwarding the search query has soft state information about the recent searches. If a search query that has been recently forwarded is received again, it is not forwarded again.

The search queries are also given a time-to-live (TTL) or a hop-count which determines how many times an individual search query can be forwarded. By manipulating the TTL we can affect how widely the search is propagated in the network. By using a large TTL, the search coverage is very good but similarly every search is seen by a large number of peers and thus processing load of all these nodes is increased. It should be noted that the search does not terminate when the search target is found but only when the search TTL reaches zero. Figure 2.6 presents flooding search with TTL value of three.

Scalability issues of BFS search are discussed in the measurement study made by Backx et al. [34], where a Gnutella file-sharing network's peer's background bandwidth consumption was measured to be more than 600 kilobytes per minute compared to less than a few kilobytes per minute in architectures where different kind of search algorithms were used (centralized and semi-centralized architectures). However, as Kalogeraki, Gunopulos, and Zeinalipour-Yazti [29] note, flooding search can be quite efficient in limited communities such as in company networks, where the maximum search load is limited by the limited size of the network.

Modified BFS

Kalogeraki, Gunopulos, and Zeinalipour-Yazti [35] present modified BFS which behaves similarly to regular BFS but instead of flooding the search query to every neighbor, each node forwarding the search message floods the query to its neighbors with a certain probability *p*, thus limiting the amount of messages in the network. This algorithm reduces the number of messages compared to the original flooding search but does it on the cost of search coverage. Depending on the selection of *p* it may also cause a large strain on the network when large values of search TTL are used.

Iterative Deepening

Yang and Garcia-Molina [36] present an improvement to regular flooding search called Iterative deepening search. In iterative deepening search, the search TTL is progressively increased so that flooding covers a larger radius on every step. The search can be controlled so that the TTL is increased until a preset number of hits are found, or when a specified TTL-limit is encountered. Lv et al. [37] present a similar search method called Expanding Ring Search. Chang and Liu [38] present a Controlled Flooding Search algorithm for wireless ad-hoc networks where sequences of TTL values are intelligently selected to minimize the cost of searches in terms of power consumption.

2.2.3 Random Walks

Random walk is a search algorithm where each node along the search path forwards the query to a single randomly chosen neighbor. The search starts by the originating node sending out *k* query messages to its randomly chosen neighbors. Each of these query messages is called a walker. Each walker follows its own path so that intermediate nodes forward it to a random neighbor at every step. However, the intermediate nodes do not replicate the walker but send it only to one node forwards. Figure 2.7 presents the random walk search algorithm.

FIGURE 2.7: LONG RANDOM WALK [33]

Lv et al. [37] present two methods for terminating the search in random walks. A TTL based method and *checking method.* In the TTL based method, the walker terminates when its TTL reaches zero, just as in the BFS algorithm. In the checking method, the walker periodically checks with the original search source if the search criterion is fulfilled. The checking method also uses TTL as secondary terminating criteria, but usually with much larger values of TTL, mainly for preventing loops.

The advantage that random walks give compared to flooding algorithms is the reduction in messages sent in the network. In the worst case it produces *k x TTL* messages. This is a huge improvement compared to BFS. However, the major problem of random walks is the search success rate in the network.

Modified Random Walk

Gkantsidis, Mihail, and Saberi [33] present modified random walk with local flooding. In this algorithm, the walker is being forwarded as in the regular

random walk. But in addition to forwarding the walker to a random neighbor, each intermediate node also floods the search to all its neighbors with a small TTL. According to authors, the modified random walk with local flooding reduces the search time compared to regular random walks. Random walk with local flooding is presented in Figure 2.8.

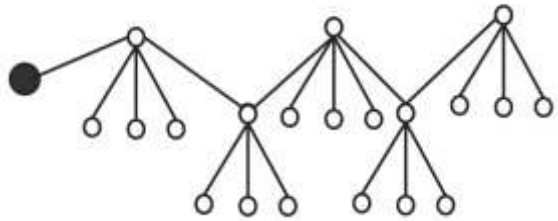


FIGURE 2.8: SHORT RANDOM WALK WITH LOCAL FLOODING [33]

2.2.4 Structured Search

In DHT architectures search is deterministic. A query is routed between nodes so that on every step the query gets nearer to the node responsible for the particular part of the hash space where the pointer to the requested resource lies.

The DHT search is based on the idea that every resource has a unique hash or resource identifier (ID) calculated from its properties, and that every node in the network is responsible for a certain part of the hash space. This way, information about resources is distributed among the network nodes.

When searching for a resource, the requestor calculates the hash for the resource, e.g., from the resource name. Then, it sends the query towards a neighbor node whose ID number is closest to the resource ID. This way, the request is forwarded hop by hop nearer the node whose ID is nearest to the resource ID in the whole network. The final node then replies the requestor if it has the pointer to the resource. It is completely possible that the final node does not have information about the resource, but in this case the asked resource is not available anywhere in the network.

A major benefit of DHT search is that the request is not sent to unneeded nodes along the search process but it is forwarded a finite number of hops, always nearer to its final destination. Thus, the search load for intermediate nodes is also smaller than in decentralized peer-to-peer architectures where some variant of flooding search is used.

On the other hand, a major drawback of DHT search is that the exact resource ID has to be known for the search. Because DHTs are based on calculating a hash of some resource property and deriving the resource ID for routing from that, it is not possible to perform wildcard searches or other searches with partial information. Also if multi-criteria search is to be supported, there must be an own ID space for each possible search criteria, e.g., own resource IDs derived from the resource name, resource creator, resource size, etc.

There has been some work trying to tackle the problem of the exact match nature of the DHT search. For example, Joung, Yang, and Fang [39] present a DHT search architecture where each resource is associated with variable number of keywords or tags. The search query can then contain variable number of these keywords but does not have to contain all the keywords of a resource. This way, resources can be searched with partial information – by knowing only subset of resource keywords – in DHT too. However, the exact match nature of the DHT search stays in the keyword based search architectures too, as the individual search keywords have to be complete and have to match to the keywords associated with the searched resource.

2.2.5 Comparison of Search Algorithms

Only three of the presented search algorithms can be considered comprehensive in their search coverage, i.e., they find the searched resource if it exists in the network. These are the centralized, flooding, and DHT search algorithms. Comparison of the key properties of these algorithms is presented in Table 2.1.

TABLE 2.1: COMPARISON OF SEARCH ALGORITHMS (ADAPTED FROM [30])

System	Per node state	Communication overhead	Wildcard search
Centralized	$O(N)$	$O(1)$	Yes
Flooding Search	$O(1)$	$\geq O(N^2)$	Yes
DHT Search	$O(\log N)$	$O(\log N)$	No

In the centralized search, all state information is held in a single server. The disadvantage of this is that the single server thus becomes a potential bottleneck and is a potential single point of failure. The advantage of the centralized search is that the comprehensive search in the centralized architecture takes only one message exchange – the message from the client to the server and back.

In the flooding search, state information in scattered around the network – every node knows only about its own resources. Thus, if a node fails, only information about that node's resources is lost. To find a resource with flooding search potentially all the nodes in the network have to be searched; thus, the communication overhead of the flooding search is proportional to the square of the number of the nodes in the network.

Finally, in the DHT search, state information is distributed among all the nodes in the network so that every node is responsible for a part of the resource space. Thus, if a node fails, other nodes have to take shared responsibility of the failed node's resource space. On the other hand, because of the DHT structure, the search is routed towards the target on every hop; leading the communication overhead of the DHT search being proportional to the logarithm of the number of the nodes in the network.

Because of the DHT query routing, the DHT search is much more efficient than the flooding search in terms of communication overhead. And its upside compared to the centralized search is that it still allows the resource index to be distributed around the network, avoiding a single bottleneck or a potential point

of failure. However, it must be noted that DHT can be compared with centralized and flooding search in terms of full network coverage only when exact match searches are considered. The DHT search does not support wildcards searches, and thus cannot be used with partial search information.

2.3 Conclusions

A major difference between all peer-to-peer architectures and the client-server architecture is that in peer-to-peer architectures resources are hosted in edge-nodes and not in a centralized server. Availability of popular resources in peer-to-peer networks is very good when compared to a single centralized server, as peers that have downloaded a popular resource from a peer-to-peer network will also be sharing that resource to other peers. In contrast, if the resource is hosted by a centralized server, a failure of the server brings resource availability to zero.

Compared to the client-server architecture, peer-to-peer architectures are scalable; one does not need to expand server capacity as the user count increases – each user brings capacity with it to the network (bandwidth, processing power, and other resources). Thus, peer-to-peer architectures also handle flash crowds very well; i.e., situations where a certain resource becomes suddenly hugely popular and is requested by many network participants. It is also cheap to provide peer-to-peer because there is no need for centralized resources.

On the other hand, in peer-to-peer architectures, a resource is available as long as a node hosting this resource is online. When all the peers that have a particular resource are offline, the resource is not available for other peers. This is a very relevant risk in networks where nodes join and part the network often, and for resources that are not hugely popular, and thus not distributed widely in the network.

Chapter 3 – Mobile Peer-to-Peer

As mobile handsets have gotten more processing power, as faster wireless communication technologies have evolved, and as mobile data plans have gotten cheaper, have many traditional networked applications, such as, email clients and web browsers found their way into the mobile world. In addition to these traditional applications, a small set of peer-to-peer applications have been developed for mobile devices.

These peer-to-peer applications are often ports of peer-to-peer applications used in fixed networks. However, mobile devices have some inherent differences from traditional PCs, and their limitations should be taken into account in peer-to-peer application design.

In this chapter, we discuss the requirements and constraints of mobile environment that have to be taken into account when developing a peer-to-peer system for a mobile platform, and consider suitable architectures for mobile peer-to-peer use. Last, we present some mobile peer-to-peer middleware and application programs.

3.1 Requirements of Mobile Environment

Mobile environment has some technical constraints and special requirements that do not exist in the fixed networks. In this section, we discuss those technical constraints and requirements along with mobile user requirements.

3.1.1 Technical Constraints

The first constraint of mobile environment is the limited network bandwidth available for the mobile device. Although advances in wireless technologies have enabled broadband mobile connections, we still have to take the limited bandwidth into account when designing networked mobile applications. In terms of peer-to-peer this means minimizing non-relevant traffic, e.g., the traffic needed for keeping up the peer-to-peer topology and forwarding of unneeded search traffic.

Also, the modern wireless technologies often provide asymmetric bandwidth, meaning there is more capacity available from the mobile network to the handset than the other way around. This is especially a bad design choice when considering peer-to-peer applications where the content is being distributed from the mobile nodes.

The second constraint is the limited computational power in mobile devices. This includes both, the CPU power and the available program memory. Thus the peer-to-peer application should not have computationally intensive algorithms or use large data structures.

The third constraint is the limited battery capacity of the mobile device. The power consumption is directly tied to the use of the radio resources, CPU cycles, and memory access. By limiting bandwidth use and computationally intensive algorithms in the application the battery can be conserved. Also, selecting the most power efficient radio technology – when there is more than one available – is directly linked to the power consumption.

As we consider these technical constraints in the design of mobile peer-to-peer application, we should select a mobile peer-to-peer architecture that creates a minimal signaling load on the mobile peer, uses no complex algorithms or data structures, and has efficient protocol coding. According to these constraints, we can disregard decentralized peer-to-peer network architectures as they have a high signaling load on all nodes due to the decentralized nature of the search. Structured architectures have fewer problems with search traffic, but they require extensive signaling for topology management to keep the network structure intact as peers join and leave the network.

Centralized and hybrid peer-to-peer architectures, however, meet the constraints. They place most of the overhead to super-nodes which can be located in the fixed network. Mobile nodes only need to worry about keeping up a connection to a single super-node.

3.1.2 Special Issues

In addition to the technical constraints, there are some special needs in the mobile environment that do not manifest themselves in fixed networks.

The first issue is high churn. This means that mobile devices often stay disconnected for a long time, i.e., their connectivity is intermittent. The frequent joins and leaves should affect other nodes minimally – there should not be signaling overhead for other nodes as one node joins or leaves the peer-to-peer network. The effects of high churn should be constrained to a few nodes, preferably to those with good connectivity in the fixed network, e.g., the super-nodes.

The second issue is radio selection in a multi-radio device. Mobile devices are nowadays equipped with several radios that support packet data communications. In addition to their long-range cellular radios (e.g., GSM and 3G WCDMA), they often have short-range radios (e.g., Bluetooth), and medium-range radios (e.g., IEEE 802.11). The mobile device should be able to select the

best radio according to the situation, i.e., use a long-range radio when the device is on the move, and use a short or medium-range, high-bandwidth radio when it is in stationary.

The third issue is operator control. The mobile operators have traditionally provided services in their walled-gardens, i.e., the services have been tied to the operator and have been only usable to customers of that specific operator. Thus, operators have had tight control on the service use. Conversely, peer-to-peer applications are based on openness and free communications between multiple peers over operator boundaries. To enable mobile peer-to-peer communications, the mobile nodes have to be able to communicate with each other over operator boundaries but at the same time each mobile operator should be able to exercise some control on its users.

Considering these issues we can disregard decentralized and structured peer-to-peer architectures as they are not suitable for high churn environments. In both of these architectures, effects of joins and leaves are propagated to several neighboring nodes – depending on the degree of connectivity between the nodes. In structured peer-to-peer architectures the resource index also has to be redistributed every time a new node joins or leaves the network. Structured and centralized peer-to-peer architectures also do not enable operator control, as they do not have centralized points of control.

Centralized and hybrid peer-to-peer architectures are suitable for high churn environments. The effects of mobile devices joining and leaving the peer-to-peer network are constrained to the super-node. They also allow the operator to control the network by hosting the super-node service. Whereas the centralized architecture is suitable for a single operator environment, the hybrid architecture is better for multi-operator environment where each operator needs to have some control on its users by running its own super-node. Operators can use a special architecture, such as that presented by Xie et al. [40]

to enable cooperative traffic control between peer-to-peer applications and the network provider.

The multi-radio connectivity is not directly related to peer-to-peer application design, as it is a feature that is needed by all networked applications in the mobile device. In [14], we present a model for autonomous radio interface selection on mobile handsets.

3.1.3 User Requirements

What users want from the mobile peer-to-peer applications is pretty much the same what they want from the peer-to-peer applications in fixed networks. From application perspective, users want to find content efficiently; they do not always know the exact name of the piece of content they are looking for, so ability to search content with partial information, e.g., with wildcards, is essential.

On the other hand, users want to have the service at affordable price. Compared to fixed broadband connections that are virtually always based on flat-rate charging, the mobile Internet connections have been mostly based on usage and data-transfer charges. The usage based charging model is not suitable for peer-to-peer use, as it is difficult for an ordinary user to estimate the amount of data he has transferred, and thus the price of the application use. However, many cellular operators are starting to provide affordable flat-rate pricing also for mobile users.

Considering the user requirements, the only issue affecting the architecture choice is the ability to perform wildcard searches. All non-structured peer-to-peer architectures support wildcard searches. Thus, user requirements only rule out structured peer-to-peer architectures as they do not support wildcard searches.

3.2 Building a Mobile Peer-to-Peer Architecture

To summarize the most important points in mobile peer-to-peer application design, the architecture should:

1. Minimize the traffic in mobile nodes to conserve bandwidth and processing load, and thus also the battery on the mobile device,
2. minimize adverse effects of high churn on mobile nodes,
3. make wildcard search available for users, and
4. allow the operator to have control on the service.

When selecting the mobile peer-to-peer architecture we have to rule out the structured architecture due to its inability to provide wildcard search and its bad performance in high churn environments. We also cannot consider the decentralized peer-to-peer architecture as high churn constantly breaks the topology, and as the forwarding of search messages is performed by the mobile peers.

As we saw in the previous chapter, the search methods of decentralized peer-to-peer networks place strain on every node. We want to conserve the limited resources of the mobile device so that the device does not have to process searches sent by other peers in the network.

This leaves us with two good choices for the mobile peer-to-peer architecture; the hybrid and the centralized peer-to-peer architecture. The centralized peer-to-peer architecture is suitable for a small, single operator environment, whereas the hybrid architecture provides more scalability, allows different operators to have control on their own users in multi-operator environments, and allows peer-to-peer communication between customers of different operators. The hybrid architecture also allows using lightweight search algorithm between the mobile node and the super-node, whereas a more complex search, e.g., flooding search, can be used between the super-nodes.

Bakos et al. [41] present similar results on mobile peer-to-peer network architecture selection. They present a simulation study of different mobile peer-to-peer topologies, where they conclude that semi-random mesh (i.e., decentralized peer-to-peer) is the best topology for a network of similar devices, e.g., when all the nodes are mobile phones. Whereas, connected stars (i.e., centralized peer-to-peer) topology is good for a network which consists of devices with different capabilities, e.g., mobile phones and fixed network nodes.

3.2.1 Proprietary vs. Standard Protocol

Traditionally, the peer-to-peer file-sharing protocols have not been standardized by any standardization body. The creation of peer-to-peer protocols has been tied to the creation of different peer-to-peer applications. Once these applications have become popular, other developers have developed applications supporting the same peer-to-peer protocol. These protocols have thus become de-facto standards, e.g., BitTorrent and FastTrack protocols.

The advantage of using these de-facto protocols is that they already have a large user base and they are tested by time. The disadvantage of using these protocols is that when there is no official standardization body overseeing the protocol development, the protocol may have multiple non-interoperable versions in development. Different clients may use different protocol versions, and nothing guarantees that the different versions interoperate with each other.

There are not many standardized peer-to-peer protocols. One that we can consider standardized is JXTA [42] which we discuss later in this chapter. Even though there are not many standardized peer-to-peer protocols, nothing prevents extending existing standard protocols for peer-to-peer use. Take, for example, the Session Initiation Protocol (SIP). SIP is not designed to be a peer-to-peer resource sharing protocol but a signaling protocol for internet multimedia communications. However, with minor extensions, SIP can be used for peer-to-

peer resource advertisement, resource location, and connecting to these resources. We discuss these SIP extensions in the next chapter.

There are some advantages in using standardized protocols for peer-to-peer communications. 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 peer-to-peer application on top of SIP allows the application to be integrated closely to the network and enables the mobile operator to implement supporting functionality more easily, e.g., charging and accounting of peer-to-peer application usage.

3.3 Mobile Peer-to-Peer Applications and Middleware

In this section, we review the recent academic research on mobile peer-to-peer applications and middleware. As our mobile peer-to-peer application is currently providing only peer-to-peer file-sharing support, we limit our focus on mobile peer-to-peer file-sharing applications and generic mobile peer-to-peer middleware platforms.

3.3.1 File-sharing Applications

In this section, we discuss some mobile peer-to-peer file-sharing applications. Some of these applications are based on well known peer-to-peer protocols, whereas others have totally original architectures.

Network Memory among Mobile Devices

Sambasivan and Ozturk [43] present a mobile peer-to-peer application that allows mobile peers to share contents of their memory, e.g., pictures, between each other. They present an application that is based on Symbian platform. The

application uses short-range Bluetooth communications, and does not rely on any centralization, thus being based on decentralized peer-to-peer architecture.

In this application, the mobile nodes discover each other dynamically based on proximity and form an ad-hoc network. As Bluetooth is a short-range radio, the authors note that one of their assumptions is that the devices have to be accessible in the Bluetooth range until the resource transfer between the devices is complete. This prevents the use of the application in dynamic scenarios where people come and go frequently. On the other hand, in static scenarios, such as in a class-room environment, the application works just fine.

Mobile Proxy

Raivio [44] presents a hierarchical mobile peer-to-peer architecture based on a concept of mobile proxy. In this architecture, each mobile device connects to a predefined mobile proxy. The mobile proxy is part of the fixed peer-to-peer network and it acts as cache for mobile clients, caching the data mobile devices are uploading so that subsequent uploads can be done straight from the cache, and not over the limited air interface of the mobile device.

In this architecture, the mobile proxy functions as a kind of super-node; the mobile node sends its query first to the mobile proxy which then checks its own index of local mobiles for the queried resource, and after that, floods the query further to the peer-to-peer network. The architecture also allows nearby mobile nodes to communicate directly with each other without the help of the proxy.

SymTorrent

Kelényi, Ekler, and Pszota [10] have developed a full-featured BitTorrent client for the Symbian platform, called SymTorrent. SymTorrent enables mobile users to connect to BitTorrent trackers and transfer files with other nodes connected to the same tracker.

SymTorrent is based on the popular peer-to-peer file-sharing protocol called BitTorrent. BitTorrent differs from most of the traditional peer-to-peer protocols as it does not provide integrated search but depends on people finding the torrent files by other means, e.g., from web sites. Torrent files have a pointer to a tracker server – which is a kind of super-node – that knows which nodes are sharing the piece of content associated with the torrent file. The BitTorrent protocol itself handles only the distributed file-transfer. Thus, there is no single global BitTorrent network, but several mini-networks around each tracker. Architecturally these mini-networks can be seen as centralized peer-to-peer networks where the trackers mediate file-transfers between the edge-nodes.

Mobile eDonkey

Oberender et al. [12] describe a mobile peer-to-peer file-sharing architecture based on the eDonkey protocol. The original eDonkey protocol has been modified to make it more suitable for mobile use. In this architecture, there is an index server that keeps track of the popularity of the shared files in the network and exports this popularity data to the cache peer. The cache peer then stores these popular files in its cache, and the crawling peer supports the index server by linking it to other index servers in the Internet. The resulting architecture is something between centralized and hybrid peer-to-peer.

The benefit of this architecture is that the cache peer allows storing of popular files, residing initially in mobile devices, in the fixed network. Thanks to the cache peer, popular files do not have to be transferred multiple times over the air interface when they are requested by another mobile node.

Symella

Kelényi, Forstner, and Forstner [11] have developed a Gnutella file-sharing client for the Symbian platform. Symella is a Gnutella 0.6 client that works on Symbian smart phones. Symella was released in late 2005, and to our knowledge, it was

the first publicly available mobile peer-to-peer application. The software enables a mobile user to search and download content in a Gnutella network, but it does not enable users to share anything.

As Symella is based on the Gnutella 0.6 protocol, its architecture is hybrid. The hybrid architecture is good for mobile environment, because as leaf nodes the communication overhead in mobile nodes is small. Most of the search traffic is handled by super-nodes located in the fixed network.

Mobile Gnutella

Hu, Thai, and Seneviratne [45] argue that the usual peer-to-peer file-sharing networks, such as Gnutella[1], 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.

In this architecture, the mobile agent handles most of the signaling traffic, such as searches, and directs only download requests to the mobile device. The mobile device can then perform the actual file transfer directly with the other end node, or alternatively the mobile agent can perform the file-transfer on behalf of the mobile device.

Network-Aware MP2P File-sharing

Huang, Hsu, and Hsu [46] discuss how wired peer-to-peer file-sharing applications rely on stable connections and how these assumptions are not suitable for mobile peer-to-peer networks where mobiles roam between

[1] Gnutella versions before 0.6 were based on decentralized peer-to-peer architecture, and thus the search messages were flooded among all network participants.

networks, where network paths between two peers may change rapidly, and where churn is a major concern.

Authors present a system architecture where a mobile peer-to-peer file-sharing network is divided into multiple network-aware clusters. These clusters are formed of nodes located near each other in IP-address topology (e.g., in the same subnet) to improve the performance of the peer-to-peer network. In each cluster, there is a super-node which handles queries from other peers inside the cluster. The super-node holds an index of files available in the cluster, and in case it does not find the queried file in its index, it forwards the query to nearby clusters. The inter-cluster queries are flooded between clusters and they are equipped with a TTL field, so that the query expires after a certain number of forwards.

The authors also describe a mobility aware file discovery control (MADFC) scheme which uses the publish-subscribe method for placing long-lived file queries into the super-node. The super-node will then continuously look up for the queried resource as new nodes are joining and parting the mobile peer-to-peer network.

When a mobile peer later joins the network and publishes its list of files to the super-node, the super-node first checks the file list against any registered queries that some other peer has active. If there is a match, the super-node informs the peer who placed the query. According to authors, this query mode reduces the amount of query messages sent in the network.

The authors also describe a resource provider selection algorithm where a mobile node can select its file-transfer peers according to network performance metrics, such as bandwidth and round-trip time information. The super-node helps the mobile node in the peer selection by providing a list of candidate resource providers with their mobility information.

3.3.2 Middleware

Mobile peer-to-peer middleware provides peer-to-peer communication services for overlying applications. By facilitating these middleware services, the application programmers do not have to bother with implementation details of peer-to-peer protocols.

JXTA/JXME

Juxtapose (JXTA) [47] is an open source peer-to-peer platform originally developed by Sun Microsystems in 2001. A good introduction of JXTA and evaluation of its suitability for mobile use is given by Maibaum and Mundt [42]. In [48], Blundo and Cristofaro describe a Bluetooth based JXME (JXTA for Java Micro Edition) infrastructure.

JXTA creates a virtual network over IP or non-IP network, hiding the underlying protocols from the applications sitting on top of JXTA. JXTA provides several peer-to-peer communication services and protocols for its users [49]:

- Peer Resolver Protocol (PRP) allows a peer to send a search query to another peer.
- Peer Discovery Protocol (PDP) allows a peer to discover other advertisements (peer, group, service, and pipe).
- Endpoint Router Protocol (ERP) allows a peer to query for routing information to route messages through the network.
- Pipe Binding Protocol (PBP) allows a peer to bind a pipe endpoint to a physical peer.
- Rendezvous Protocol (RVP) is the mechanism by which services are bootstrapped within the network.
- Peer Information Protocol (PIP) allows a peer to query for current status of another peer.

JXTA has two categories of peers; super-nodes and edge nodes. Super-nodes are either rendezvous or relay nodes. Rendezvous nodes are used for enabling

communication between edge-nodes in different networks. They setup a DHT network with other rendezvous nodes for inter-network query routing. Relay nodes allow edge-nodes to communicate through firewalls or NATs, and thus be part of the JXTA network. Ordinary edge nodes, or JXTA peers, are organized in peer groups around the super-nodes.

Light weight version of JXTA, called JXME (JXTA for J2ME) has been ported for Java enabled mobile devices. There are two version of JXME available; proxied version for slower J2ME devices, and proxyless version for more powerful mobile devices. The proxied version needs a JXTA Relay to communicate with other JXTA nodes, whereas the proxyless version is similar to a regular JXTA edge node as it does not need the relay.

P2P Services

Keller et al. [50] present a two-layer mobile peer-to-peer service platform that consist of a Core peer-to-peer services layer and an application specific services layer. Universal core peer-to-peer services layer provides basic peer-to-peer services that can be then utilized by application specific application layer components.

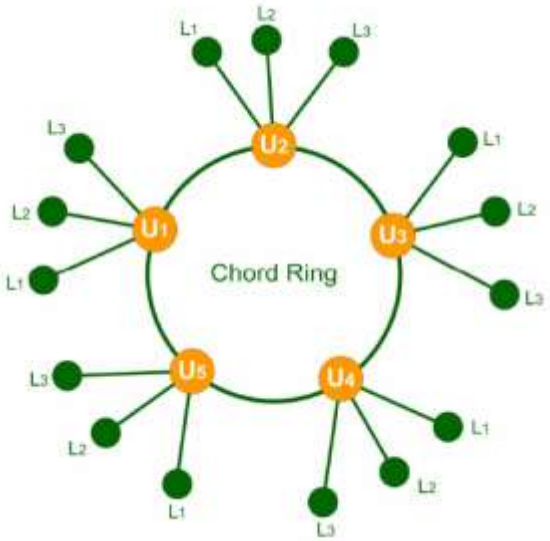


FIGURE 3.1: HIERARCHIAL DHT

The architecture is based on a hybrid peer-to-peer architecture where more capable nodes function as super-nodes. The super-nodes are connected to other super-nodes in a DHT (see Figure 3.1), whereas the leaf nodes are connected to super-nodes but not to the DHT itself. The authors note that the problem of their architecture is that DHT natively supports only exact match queries, and that this is inappropriate for many applications.

PnPAP

Harjula et al. [51] propose an application framework called the Plug-and-Play Application Platform (PnPAP). PnPAP allows mobile applications dynamically selecting among many underlying peer-to-peer and session management protocols. The PnPAP sits between application layer and P2P protocol layer (see Figure 3.2). In addition to conventional peer-to-peer protocols, such as, Direct Connect (DC) and JXTA, the PnPAP application framework also allows using Session Initiation Protocol (SIP) as the underlying communication protocol.

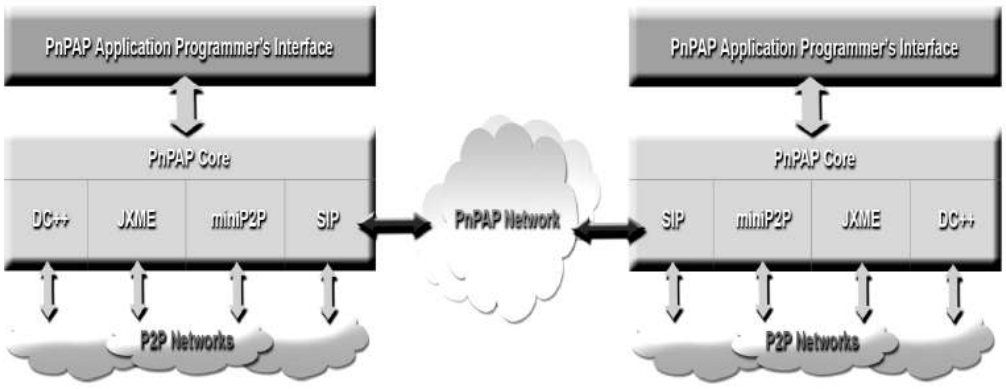


FIGURE 3.2: PLUG-AND-PLAY APPLICATION FRAMEWORK [52]

In another paper, Howie et al. [52] describe how SIP is used by the PnPAP. Authors describe how SIP is used to find resources, initiate downloads, and find new protocol images on other PnPAP nodes. The SIP communication architecture uses SIP REGISTER message to query for available resources from PnPAP SIP registrar. The authors suggest using instant messaging capabilities of SIP, along with MIME encoding, to convey binary images over SIP between PnPAP peers

(see Figure 3.3). However, this is rather inefficient and a hostile way of conveying large binary files, as the SIP message overhead is quite large, and as SIP messages are often software forwarded in the SIP network, thus placing considerable load to SIP proxies.

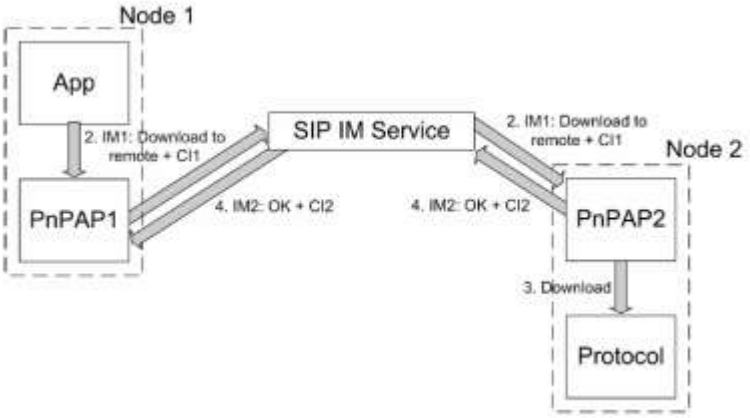


FIGURE 3.3: PNPAP DOWNLOAD USING INSTANT MESSAGING

Mobile Chedar

Kotilainen et al. [53] present Mobile Chedar peer-to-peer middleware for mobile devices based on the Chedar peer-to-peer middleware [54]. Mobile Chedar uses Bluetooth to connect to other peers. Mobile peers in the Mobile Chedar network can also communicate with nodes in a fixed Chedar network by using Chedar gateway peers. The Mobile Chedar can be used to locate unused resources, such as CPU time and storage space in the network.

3.4 Conclusions

In the first part of this chapter, we discussed the special requirements that mobile environment and its users pose on peer-to-peer architecture selection. We concluded that hybrid or centralized architectures are good choices for mobile peer-to-peer applications as they minimize the communication and processing overhead in the mobile peer – whereas peer-to-peer solutions based on decentralized and structured architectures are less suitable for mobile use as

they require mobile devices to be full members of the peer-to-peer network thus placing considerable load on them.

In the second part, we reviewed different mobile peer-to-peer file-sharing applications and mobile peer-to-peer middleware. Most of the applications presented here are based on a hybrid peer-to-peer architecture or some kind of application gateway architecture which abstracts away the complex peer-to-peer network.

Chapter 4 – Mobile Peer-to-Peer over SIP

Next-generation mobile networks, like the Third Generation Partnership Project's (3GPP) IP Multimedia Subsystem (IMS), are largely built onto well known internet protocols, such as the Session Initiation Protocol (SIP). These protocols are well understood and documented, and they are tested in large scale deployments.

To enable mobile peer-to-peer communications in next-generation mobile networks, we have designed a mobile peer-to-peer platform that works on top of SIP. Using SIP instead of a proprietary peer-to-peer protocol ensures that the peer-to-peer platform can be easily adapted to tomorrow's SIP-aware mobile networks.

We begin this chapter with a short overview of IMS and SIP. Then, we present our mobile peer-to-peer application architecture, and discuss its design choices and suitability for mobile use.

4.1 Mobile Peer-to-Peer in IMS

The IP Multimedia Subsystem (IMS) was designed to 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 enables rich communications between mobile terminals over various access network technologies, be it 3G, 4G, or 802.11.

As IMS is built on well defined standard protocols like SIP, it provides a good basis for building a mobile peer-to-peer platform. SIP has properties that are useful for peer-to-peer applications, and with minor modifications it is suitable for peer-to-peer application signaling.

4.2 Session Initiation Protocol

Session Initiation Protocol [55] is a protocol for creating, modifying, and terminating multimedia sessions between two or more participants.

SIP was drafted by the 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. [56]

SIP is an end-to-end signaling 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.

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, and same text encoding rules and header styles as SMTP.

SIP is specified in RFC 3261 [55]. 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 the session participants; thus other protocols have to be used in addition to SIP to create meaningful services.

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?
4. Session setup - establishment of the session parameters.
5. Session management - transferring, modifying, terminating the session and invoking services.

4.2.1 Resource Location

When a SIP enabled User Agent (UA) starts up, it has to register to the *Registrar* of its home domain. The registration is performed by sending a REGISTER request to the registrar. This request includes user's current address and the user's Address of Record (AOR).

The registrar will update UA's current location to the *Location service*, which creates a mapping between the AOR and the terminal address, and sends a 200 OK reply back to the UA to inform that the registration succeeded.

When the user wants to contact another user, all he needs to know is the AOR of the other party. The SIP infrastructure provides message routing that enables the originating user to send the SIP message to the known AOR of the other user. First, the originating user sends an INVITE request to a preconfigured outbound SIP proxy in his home domain, or alternatively to an inbound SIP proxy in foreign user's domain – the UA sending the request functions as the User Agent Client (UAC) for this particular signaling exchange.

If the message was forwarded to the outbound proxy in the home domain, the proxy will resolve the address of 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 that domain to get information about the current location of the session target.

The proxy then uses this location information to route the message to the UA who is the final recipient. The final recipient will send the reply via the same proxies as the request came from. In addition to message routing, these proxies may also be used to implement charging and application layer firewalling functions. The UA which receives the request and sends the response to the requestor functions as the User Agent Server (UAS) for this particular signaling exchange.

After the communicating partners have located each other via proxies, they may start sending SIP messages directly between each other if the intermediate proxies have not requested to stay on the signaling path.

4.2.2 SIP Requests and Responses

SIP messages are sent in a request-response style. There is one final reply per one request. However, there can 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-confirm (INVITE – 200 OK – ACK) pattern. Different SIP requests are presented in Table 4.1.

TABLE 4.1: SIP REQUESTS

Request	Action
REGISTER	Pushes user's current Contact URI to the location service in his home domain.
INVITE	Establishes session between UAs. Is different from other requests because it is a three-way request.
BYE	Terminates the session established with an INVITE request.
CANCEL	Cancels pending requests. Request can be only cancelled if it has not been fully processed yet.
OPTIONS	Queries UA or proxy about the SIP capabilities it supports.
INFO	Conveys call control information during an existing session.
MESSAGE	Transfers user readable messages between terminals.
SUBSCRIBE and NOTIFY	Used for subscribing to and notifying of events related to the SIP system.
PUBLISH	Publishes event state information from UAC's Event Publication Agent (EPA) to Event State Compositor (ESC).
UPDATE	Modifies the state of a pending session.
PRACK	Provides reliable provisional responses.
REFER	Refers another UA to access a resource specified by Uniform Resource Identifier (URI) in the REFER request.

The SIP replies are identified by the reply codes. The reply codes are divided into six response classes, shown in Table 4.2.

TABLE 4.2: SIP RESPONSE CODE CLASSES

Class	Description	Action
1xx	Provisional	Indicate the 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.

4.2.3 SIP over P2P (P2P-SIP)

Regular SIP relies on Proxy, Registrar, and Location services that are located in fixed addresses. However, sometimes it is useful if communication can be initiated without first setting up the complex SIP server infrastructure. Peer-to-Peer Session Initiation Protocol (P2P-SIP) [8] will work in settings where there is no fixed SIP infrastructure available. In P2P-SIP, session establishment and management is collectively handled by the members of the P2P-SIP network, and thus there is no need for dedicated Proxy, Registrar, and Location services.

P2P-SIP is being developed in IETF's P2PSIP working group. Between July 2007 and December 2007, eighteen different P2P-SIP related internet-drafts have been published. As peer-to-peer SIP is very much under development, the final architecture is still unknown. However, good high-level introduction to P2P-SIP is given by Singh and Schulzrinne [57].

P2P-SIP overlay consists of P2P-SIP peers and P2P-SIP clients. The peers run collectively a distributed database algorithm which is used to store information about the mappings between AORs and Contact URIs to provide the location function. It is likely that a DHT will be used to implement this distributed database. This overlay provides the same functionality that SIP Proxies, Registrars, and Location services offer in regular SIP.

In addition to P2P-SIP peers, there may be less capable nodes, called P2P-SIP clients, connected to P2P-SIP peers, and not to the DHT itself. However, the role of the P2P-SIP client is still under debate and it is not clear if it will be included in the final architecture [58]. Matuszewski and Kokkonen [59] present a mobile P2P-SIP architecture where mobile devices function as P2P-SIP clients; thus, limiting the mobile device's communication overhead to that between the mobile client and a P2P-SIP peer – this way avoiding the DHT communication overhead in the mobile device.

P2P-SIP vs. P2P over SIP

In the next section, we present our peer-to-peer application that uses SIP as the signaling protocol. Compared to P2P-SIP, we are enabling peer-to-peer networking over SIP – not SIP over peer-to-peer networks as in P2P-SIP.

In our architecture, we still rely on SIP Proxies and Registrars in the fixed network providing the SIP message routing. However, our architecture does not care how the underlying message routing is implemented – if it is relying on fixed SIP infrastructure or peer-to-peer SIP. Our application merely uses SIP for message routing.

It may not make much sense to use P2P-SIP with our application as our hybrid architecture already relies on some fixed nodes, i.e., super-nodes, in the network anyway. However, P2P-SIP would be useful when used with a decentralized peer-to-peer application which could then work without any fixed infrastructure.

In [60], Harjula et al. present how their Plug-and-Play Application Platform (PnPAP) peer-to-peer middleware can be used over the top of P2P-SIP so that SIP messages are routed between PnPAP participants over P2P-SIP DHT. This version of PnPAP uses the resource sharing framework utilizing the SUBSCRIBE and NOTIFY scheme that we have specified in [61].

In their paper, the authors concluded that if PnPAP mobile nodes functioned as P2P-SIP clients and not as full members of the P2P-SIP DHT, the performance penalty of using P2P-SIP was minimal and the fault tolerance was improved compared to the traditional client-server SIP. On the other hand, if PnPAP nodes functioned as P2P-SIP peers, and thus as members of P2P-SIP DHT, the performance was found to be poor.

4.3 Mobile Peer-to-Peer using SIP

We have created a hybrid mobile peer-to-peer file-sharing platform which consists of a mobile client for Symbian based Nokia Series 60 smart phones and a

SIP Application Server (SIP AS) functioning as a super-peer. The software uses SIP as its underlying signaling protocol to allow its use in any SIP aware network.

Usage of SIP enables peer-to-peer signaling to be routed using SIP URIs as node identifiers. Use of SIP identifiers enables seamless mobility as the changes in node IP-address, and thus in access net connectivity, are abstracted away using the underlying SIP infrastructure.

Hybrid peer-to-peer was chosen as the underlying paradigm to minimize overhead in the mobile device and to allow operators to have control on the peer-to-peer service users by controlling the super-nodes.

The basic architecture of our Mobile Peer-to-Peer application is presented in [18], [17], whereas more detailed architecture and performance analysis is presented in [16]. Detailed mobile client software architecture is presented by Hyyryläinen in [62].

4.3.1 Client Architecture

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, or hash of the file he wants to find. Searches using multiple parameters are also possible.

The basic client functionality is divided into four modules. These are the Registrar, Finder, Transfer, and Graphical User Interface (GUI) modules. The GUI module interfaces with the user, the finder module takes care of the query processing, and the transfer module handles peer-to-peer file transfers and updates on the client's file list to the super-node. The registrar module communicates with the super-peer which peer-to-peer services are running in the mobile device. Client's high-level software architecture and communication relationships are illustrated in Figure 4.1.

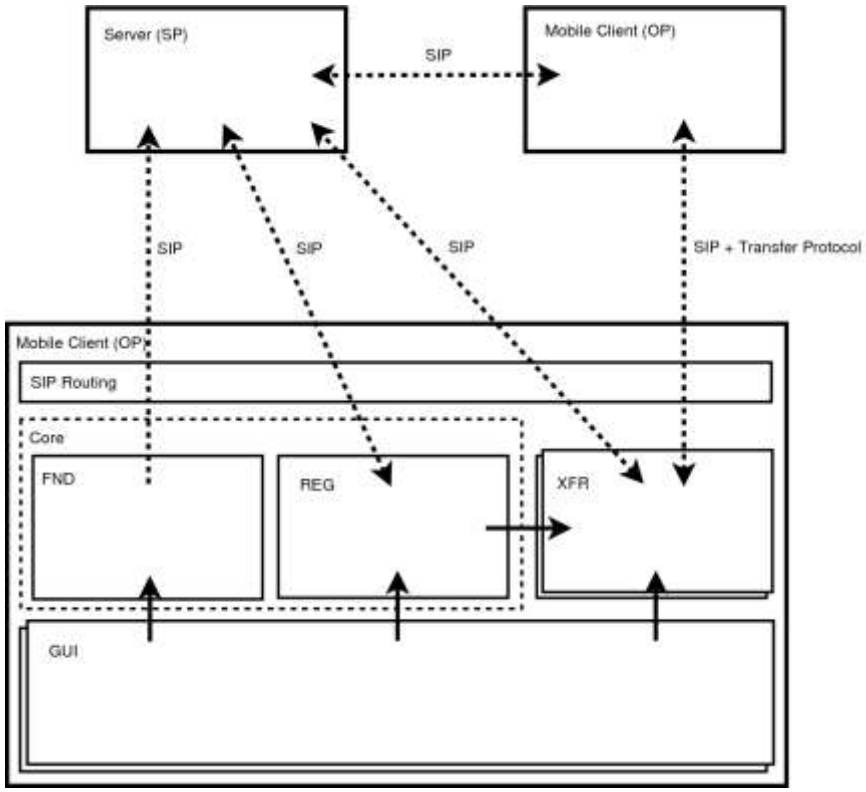


FIGURE 4.1: MP2P APPLICATION ARCHITECTURE

4.3.2 Super-Node Architecture

The super-node maintains information about the shared files on mobile clients. This information includes file names, file hashes, file sizes, and other meta-information.

The super-node interacts with the clients during searches and file list updates. File searches are initiated by a client sending a search request to a super-node. The super-node answers with a reply that contains information about the matching files and the peers having those files.

We initially proposed super-node implementation as a Jiplet [63] attached to a SIP server. However, Jiplet based super-node was never implemented. The first super-node was implemented concurrently with the mobile peer-to-peer client as a standalone Python script.

Morales Reyes [64] describes the second super-node implementation. This super-node was implemented in C++ and it uses a MySQL database for storing the peer-to-peer index.

4.3.3 SIP P2P Application Server Architecture

In [17], we describe an architecture where the super-node is implemented as a SIP application server (SIP AS) to provide full integration with IMS networks. This application server, called the Peer-to-Peer Application Server (P2P AS), interfaces with the Serving Call Session Control Function (S-CSCF) and other P2P ASs using SIP. Mobile devices connect to P2P AS via Proxy-CSCF (P-CSCF) and Serving-CSCF (S-CSCF), as shown in Figure 4.2.

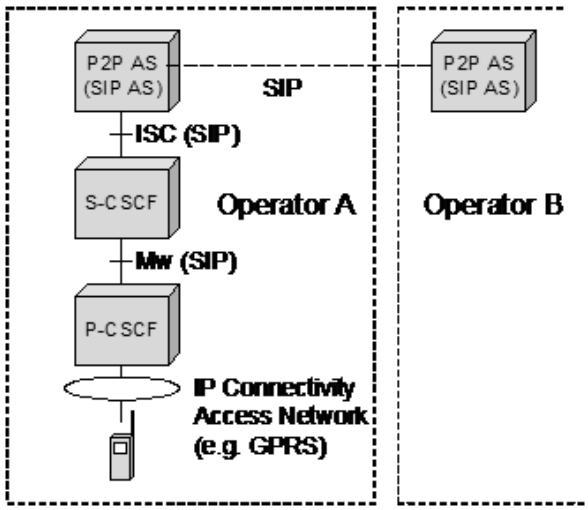


FIGURE 4.2: ELEMENTS IN P2P SIP OVER IMS

Each operator can have one or more P2P ASs which are connected to each other in the decentralized peer-to-peer fashion. This architecture is not limited to IMS networks, but allows connections to other peer-to-peer networks using SIP or some other peer-to-peer protocol.

In [19] we present a refined architecture of a multi-service overlay network that we call SIP P2P over IMS. In this architecture, there is one or more SIP P2P ASs per operator. ASs create an overlay of their own with each other. They behave as front end towards the mobile devices, make resources of the network available

to other super-nodes, help user equipment to get search results, and maintain a peer-to-peer overlay network for resource sharing. Different overlay network algorithms can be used depending what the provided service will be, e.g., unstructured flooding-algorithm for file-sharing service, or structured DHT for locating chat contacts with full email addresses. General architecture is presented in Figure 4.3.

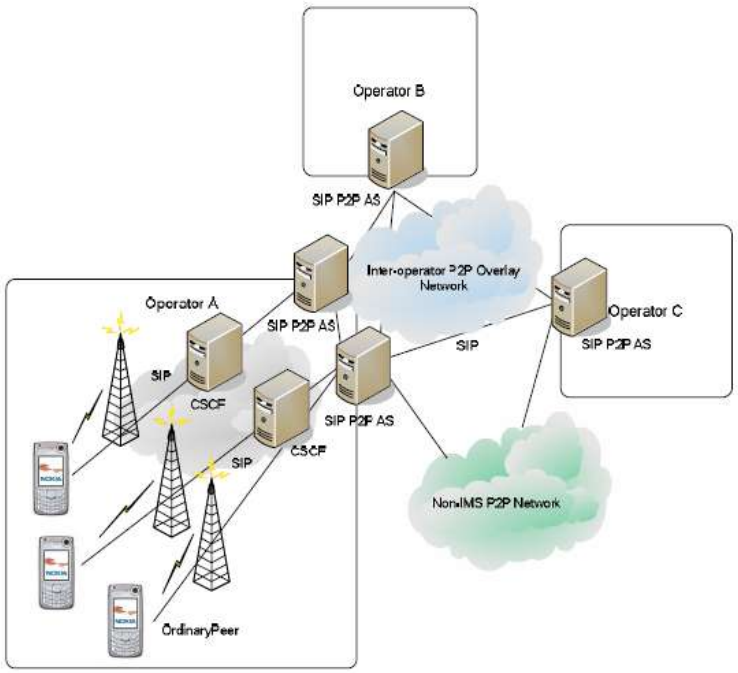


FIGURE 4.3: SIP P2P OVER IMS ARCHITECTURE

Users can publish the availability of one or more resources in their devices, perform searches, initiate file downloads, join audio or video streaming sources, conferences, or chat rooms.

File transfer is done using the Message Session Relay Protocol (MSRP), and Network Address Translation (NAT) and firewall traversal is accomplished by help of MSRP relays.

In [17], we discuss how charging functions can be implemented with the help of AS. Charging can be implemented with normal IMS charging mechanisms where S-CSCF and P2P AS analyze the peer-to-peer SIP signaling and take part in the

charging process. Operators can implement any charging scheme for the usage of SIP P2P services, including subscription based charging. P2P AS collects application-level usage records which can be used as basis for charging as well. One model would allow free searches and charge only for actual media consumption, like file downloads or video streaming.

In [16], we discuss some further enhancements to the application server architecture; such as P2P AS functioning as a cache for popular content, as a gateway to an external peer-to-peer network, or as a peer-to-peer manager which optimizes the use of network resources.

4.3.4 SIP Signaling

When a User Equipment (UE) joins a network, it publishes information about its shared resources to a P2P AS. When the UE performs search it sends a search request to the P2P AS. The P2P AS provides search results that contain a list of available resources and SIP URIs where these resources are available, e.g., content cache, streaming server, chat group manager, or other UE where the resource is stored. To fetch a resource, the UE initiates appropriate SIP session towards such endpoint to fetch the resource. This final SIP session is between the resource holder and the requester and it does not involve the P2P AS.

Initial Signaling Scheme

Our initial architecture used two standard SIP methods to implement all of its functionality: INVITE and MESSAGE. The use of these message types was largely dictated by the limitations of the SIP stack we used for the initial client implementation. Signaling flow utilized in the initial architecture is presented in Figure 4.4.

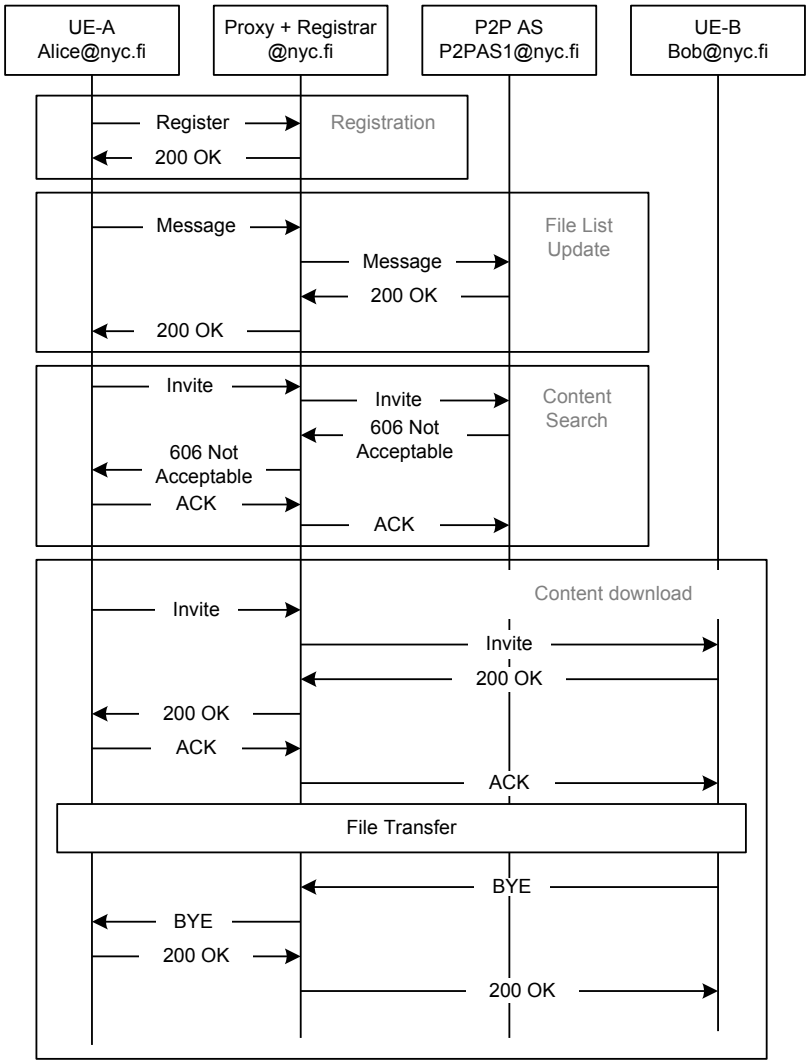


FIGURE 4.4: INITIAL SIGNALING

In our initial signaling scheme, nodes upload information about the files they are sharing to a P2P AS in MESSAGE requests, whereas search requests are conveyed in bodies of INVITE messages. The P2P AS sends search replies in the following *606 Not Acceptable* replies. An error message was selected for the search reply as it terminates the establishment of an unwanted session. However, a generic *request-reply* type of signaling message, such as HTTP GET - 200 OK, would have better suited for the situation; however, SIP is missing such a generic *request-reply* type of a signaling message.

The actual content download is initiated by sending an INVITE request to the peer that has the file of interest. The Session Description Protocol (SDP) in the message body specifies the hash of the file to be downloaded.

The file list updates and search requests encode their bodies in Extensible Markup Language (XML); thus, they can be extended easily in the future. In [16] we present a messaging flow for registering to service, updating file list, performing content search, and performing a file transfer.

Refined Signaling Scheme

We do not discuss the initial signaling scheme further but present the refined signaling scheme introduced in [19]. Further information, such as detailed message formats are given in [61].

Resource Publication

The first thing the UE does when it starts up the peer-to-peer application is the resource publication to the P2P AS. The resource publication signaling flow is presented in Figure 4.5.

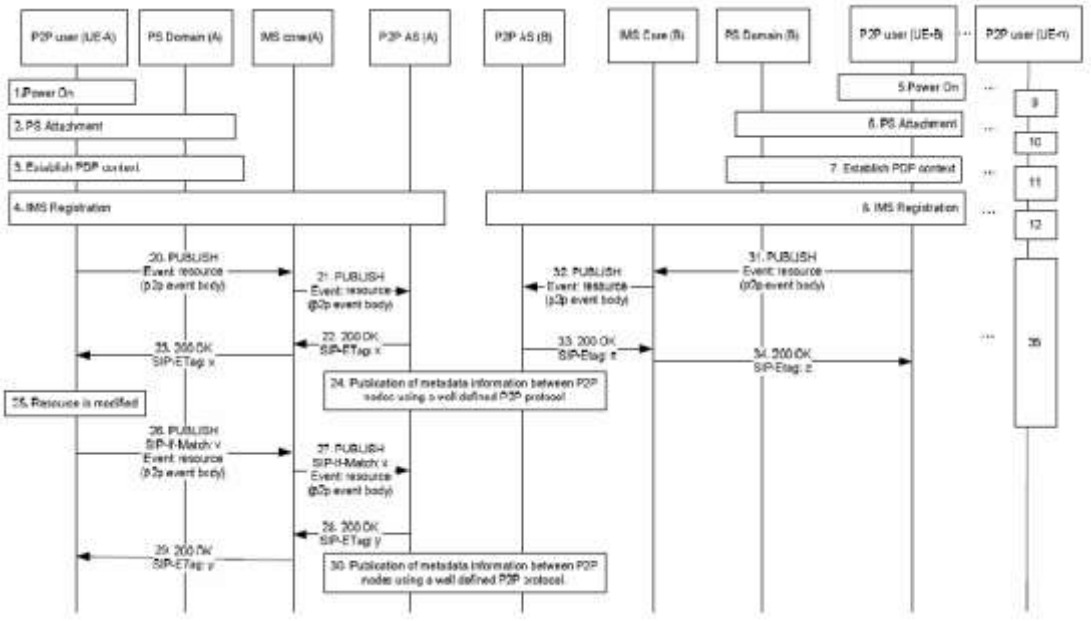


FIGURE 4.5: RESOURCE PUBLICATION

At first, the UE does a regular IMS registration: After the UE is powered on (1) and attached to a packet-switched network (2), it establishes a Packet Data Protocol (PDP) context (3), and registers to IMS (4).

When the peer-to-peer application is launched, the UE publishes availability of its shared resources to the P2P AS. SIP PUBLISH method is used with resource publication event package [65] to publish resource information to the P2P AS (20-21). The P2P AS replies with 200 OK including a SIP-ETag header that contains the entity-tag allocated to the published resource (22-23).

Next, the P2P AS may need to publish or update resource information in the P2P AS overlay network depending on the actual overlay architecture (24). In case a flooding algorithm is used for inter-P2P AS searches, nothing is done in this phase. In case of DHT algorithm, like Chord, is used, the resource metadata has to be stored into the appropriate node in the overlay.

If the shared resource is later modified in the UE (25), the UE refreshes previous publication by sending a new PUBLISH request where SIP-If-Match header is used to match entity-tag of the resource (26-27). The P2P AS replies with 200 OK that contains a new entity-tag related to the modified resource (28-29) and publishes modified information forwards in the P2P AS overlay (30).

Search

To find resources in the peer-to-peer network, the UE performs search query to the P2P AS. The search signaling flow is presented in Figure 4.6.

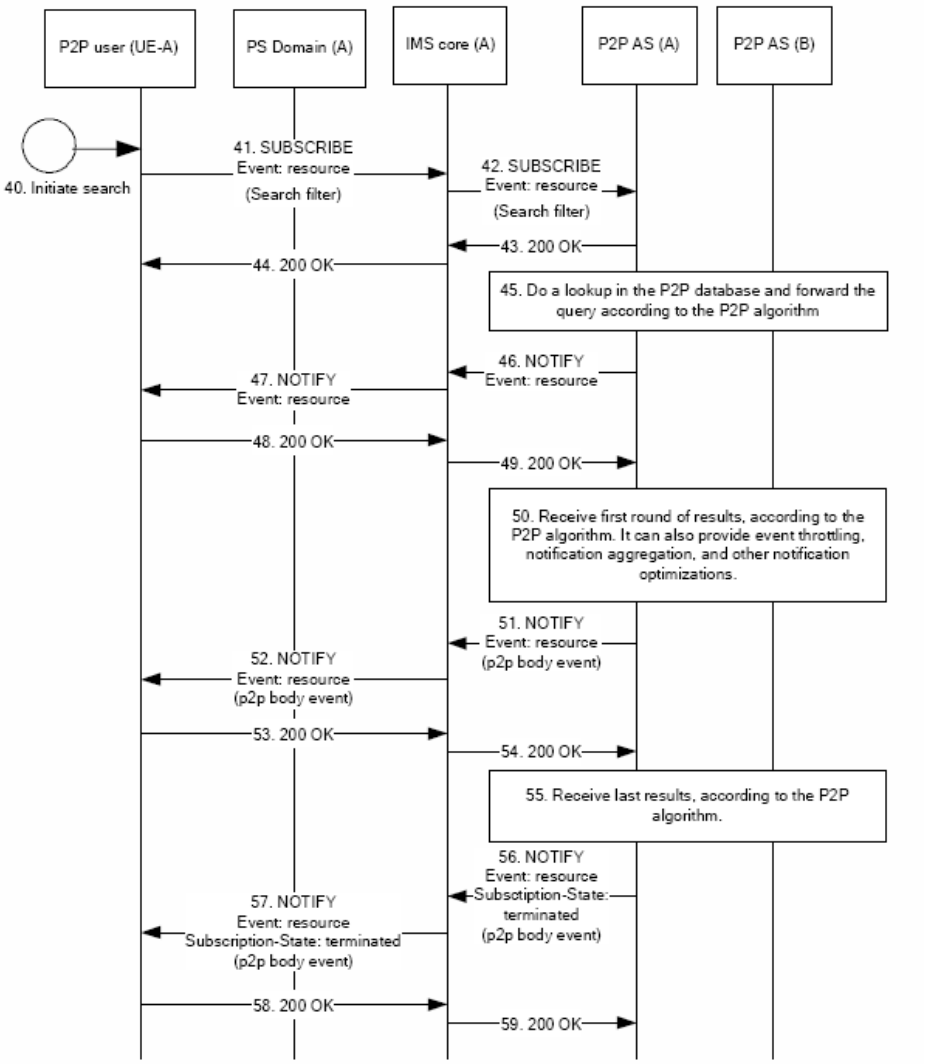


FIGURE 4.6: RESOURCE SEARCH

When a user initiates search in the peer-to-peer application (40), the UE sends a SUBSCRIBE request for the resource event package. This request is sent to the P2P AS (41-42) where it creates a soft-state subscription – meaning that the subscription will last for some time (determined by the Expires header field). The SUBSCRIBE request contains a Search filter [66] that specifies the search parameters.

The P2P AS answers with a 200 OK message, this message does not have any search results but it signals to the client that the search query has been successfully received at the P2P AS (43-44).

Next, the P2P AS looks up its own database and forwards the query further to other P2P ASs in the overlay (45).

Then P2P AS sends a NOTIFY request to the UE. This request usually contains a first collection of search results. In case the P2P AS did not have matching results in its local database and its waiting for search results from the overlay, it will send an empty NOTIFY request. The empty NOTIFY request is sent due to protocol reasons, as an immediate NOTIFY is required after a successful SUBSCRIBE as mandated by RFC3265 [67] (46-47). The UE answers with 200 OK (48-49).

After the P2P AS receives the first set of results from the overlay (50) it will send results to the UE in a NOTIFY request (51-52), and the UE will acknowledge this with 200 OK (53-54).

Later, when the P2P AS receives more results for the search, it sends further NOTIFY requests to the UE. When the last batch of results arrives from the overlay or when the search state expires in the P2P AS (55), the P2P AS sends the last NOTIFY request to the UE and sets the Subscription-State header value to *terminated* to indicate that the search state exists no more in the P2P AS.

The search method used in our architecture is good for mobile use as it is incremental. The user does not have to wait for initial search results even if the search in overlay is taking some time. However, the use of incremental search creates some communication overhead as more SIP messages are passed between the P2P AS and the UE.

Resource Connection

Finally, to acquire the interesting resource or connect to the resource, the UE has to establish a session to the resource holder. The signaling flow for connecting to the resource in another UE is shown in Figure 4.7

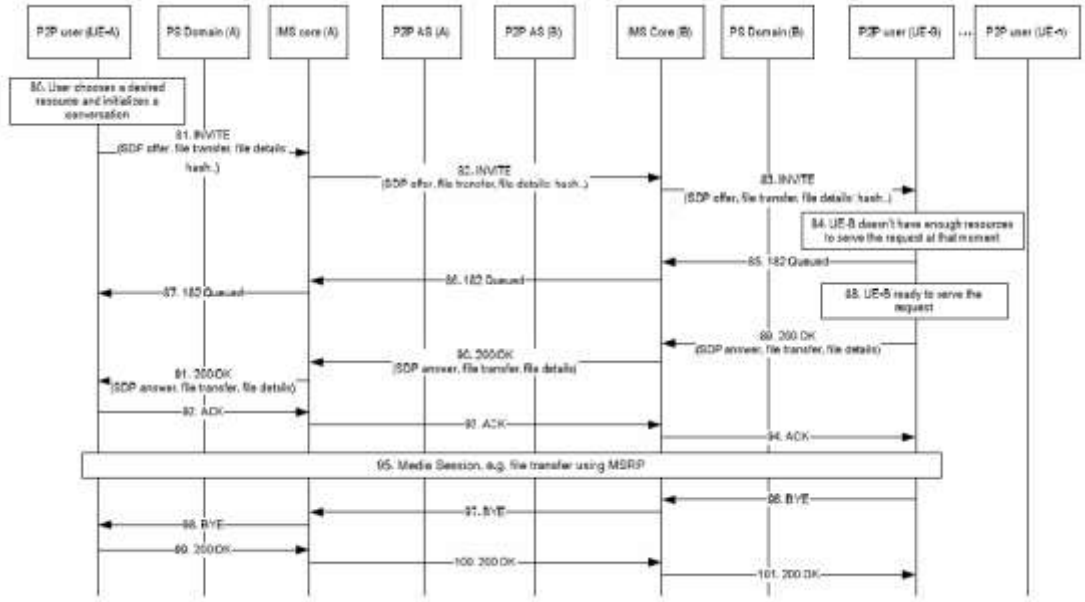


FIGURE 4.7: RESOURCE CONNECTION

When the user has found the resource he wants to connect to, he selects the resource from the search results to initiate the resource connection (80). Then, the UE-A sends an INVITE request to the SIP URI specified in the search results (81-83). This INVITE contains an SDP offer indicating a file transfer operation and some metadata indicating the resource to be retrieved.

If the UE-B does not, for some reason, have resources to fulfill the request (84) it will send 182 Queued message to the UE-A to inform that the request is queued and will be answered later (85-87). Later, when the UE-B is ready to process the request, it answers with 200 OK containing the SDP answer (89-91), which the UE-A replies with ACK (92-94).

The actual connection to the resource happens using some other protocol, such as MSRP for file transfer (95).

When the media session ends, either of the UEs sends a BYE request (96-98) that is answered by a 200 OK reply by the other UE (99-101).

4.3.5 System Performance

In [14] and [16], we present measurements results for our initial signaling architecture. Message sizes were measured and they are presented in Table 4.3.

TABLE 4.3: SIP MESSAGE SIZES

Action	Request	Size (bytes)	Reply	Size (bytes)
Register	REGISTER	370	200 OK	300
Search	INVITE ACK	430–480 320	606 Not Acceptable	370–1380
File list update	MESSAGE	450–1380	200 OK	250
Download	INVITE ACK	540 390	200 OK	290
De-register	REGISTER	380	200 OK	250

In [19], we present publication and search delay calculations for our enhanced signaling architecture. These are presented in Table 4.4.

TABLE 4.4: PUBLICATION AND SEARCH DELAY FOR ENHANCED ARCHITECTURE

Action	*Request / Reply*	*Delay*
Publication	PUBLISH	240 ms
	200 OK	150 ms
	Sum	**390 ms**
Search	SUBSCRIBE	260 ms
	Search in P2P AS	10 - 2000 ms[2]
	NOTIFY[3]	340 ms
	Sum	**610-2600 ms**

As we can see, messages are quite large in size as SIP is a text-based protocol and as it has many mandatory protocol fields. However, as the measurement results

[2] Search delay in P2P AS varies largely depending on the size of the P2P network. In case some matches are found in the serving P2P AS, the first NOTIFY can be returned almost instantly; whereas, if the search has to propagate to a distant P2P AS, the delay can be much longer.

[3] NOTIFY size varies greatly depending how many results are found. E.g., the NOTIFY used in these calculations contains information about one file found from two different peers.

show, the actual delay of transmitting any of these messages is less than one second.

In [16] we present measurements on MP2P application memory use. The memory use of the application varied between 200 and 350 kilobytes, whereas the SIP stack and SIP profile manager consumed additional 170 kilobytes of memory.

In [13] we present measurements on transfer speeds between mobile devices in a 3G network. We achieved speed of 100kbit/s with the limiting factor being the bandwidth from the mobile device towards the network. Compared to peer-to-peer transfer speeds over Bluetooth in [43], where the best measured performance for transmitting a 10 kilobyte image was 3.5 seconds, resulting in the average transfer speed just below 23kbit/s, we can note that the 3G transfer speeds were over four times faster than those of Bluetooth. In addition, we are not limited to the proximity of Bluetooth connectivity with 3G. With more advanced radio technologies, such as High-Speed Packet Access (HSPA) and Evolution Data-Only (EVDO), the transfer speeds should be an order of magnitude higher.

Acceptable Performance

In [20] we present an analysis of a questionnaire survey, where 98 potential peer-to-peer application users were asked about their peer-to-peer usage habits. Among other questions, the potential users were asked about acceptable search delays and acceptable image download delays when using a mobile peer-to-peer application. These results are shown in Table 4.5 and Table 4.6 respectively.

TABLE 4.5: ACCEPTABLE SEARCH DELAYS

Search delay (sec)	***Percentage of respondents accepting***
2	100
30	74
60	45
>60	24

TABLE 4.6: ACCEPTABLE IMAGE FILE DOWNLOAD DELAYS

Image file download delay (sec)	***Percentage of respondents accepting***
2	100
30	86
60	56
120	28
>120	8

From the results we can see that 100% of the potential users are satisfied with search delays less than two seconds and 74% with delays less than thirty seconds. When comparing these results to the measured signaling delays, we can see that the signaling delays in our application are no problem as they are always below the two second threshold.

In cases where search takes more than two seconds to complete, due to delays in the P2P AS overlay, the incremental search functionality provides results as they become available, thus enhancing the user experience.

When questioned about acceptable image file download delays, all users were happy with a sub two-second delay, whereas 86% were happy with a sub thirty-second delay, and 56% were happy with sub one-minute delay. If available bandwidth is roughly 100kbit/s, which is a typical 3G upstream bandwidth, we cannot presume that an image file can be downloaded in a sub two-second

timescale as only about 25kB can be transmitted in that time, and that does not even include signaling delays. However, in 30 seconds we are able to download 750kB of data, which should be enough for image transfer in mobile context.

As a concluding note, the signaling delays are not an issue at all. When downloading larger than 100kB objects, the file transfer delay starts to dominate over the signaling delay – thus being the limiting capability for all networked mobile applications, not just for peer-to-peer applications.

4.3.6 Securing Mobile Peer-to-Peer

According to [68], the security problems of P2P systems include: authentication, encryption, privacy and confidentiality, and ability to deal with malicious nodes. A bit different categorization is used by Daswani et al. [69], who organize the security issues of P2P data-sharing into four areas: availability, file authenticity, anonymity, and access control.

In a P2P file-sharing environment we can divide security issues by functionality into two main categories – into security issues of search and into security issues of content transfer. Whereas content searches are done using a fairly static super-peer, the actual content is transferred from peer to peer, where the other peer can be any random, never-seen-before node.

In [15], we discuss search and download security issues, as well as availability and anonymity issues of mobile peer-to-peer networks. In this paper, we discuss how securing both peer-to-peer signaling and content downloads is important to prevent unwanted entities from gaining private information from the peer-to-peer traffic.

We found out that if searches are done in centralized or hybrid peer-to-peer architectures where a node sends queries to a single super-peer, securing this connection is rather easy as the mobile node can have a shared secret with the super-peer that is used for authenticating and signaling encryption between the

mobile node and the super-node. One candidate for such shared secret is the secret key *K*, which is shared between the mobile node Universal Subscriber Identity Module (USIM) and the network Authentication Center (AuC).

However, because downloads are done between random peers, it is much more difficult to secure this inter-peer connection as it is not feasible to have shared secrets between all possible peers in the network. Thus, authentication between the peers has to be based on Public Key Infrastructure (PKI) or on a centralized authentication server.

In the paper, we also propose using Secure / Multipurpose Internet Mail Extensions (S/MIME) and Transport Layer Security (TLS) in our SIP based peer-to-peer architecture to secure the signaling and download connections.

4.4 Conclusions

In this chapter, we presented our SIP based mobile peer-to-peer architecture. Our application architecture is based on the hybrid peer-to-peer architecture where network operators are running super-nodes in their networks as SIP P2P Application Servers. Our application architecture satisfies all requirements for a mobile peer-to-peer application stated in Section 3.2:

1. It minimizes the traffic in mobile nodes to conserve bandwidth and processing load, and thus also the battery on mobile device,
2. it minimizes adverse effects of high churn on mobile device,
3. it enables users to perform wildcard searches, and
4. it lets mobile operator to have control on the service.

As our architecture uses text based SIP for signaling, the signaling is not the most efficient. However, as the measurement results show, the delays of transmitting the signaling messages were less than one second in all cases, which, according to our user study, satisfies 100% of users. Only the initial joining to the network and publishing the list of resources to the P2P AS potentially takes more time.

Chapter 5 – Conclusions

Finally, in this chapter, we provide conclusions on our mobile peer-to-peer research. We begin this chapter by reviewing the research objectives. Then, we present the key findings of the thesis. Last, we give ideas for some future research topics on the subject.

5.1 Objectives Revisited

The main objective of this thesis is to present how peer-to-peer based services can be efficiently realized in next-generation SIP/IMS networks by reusing their existing protocols as much as possible, and to present some enhancements to these protocols. We also evaluate what kind of special requirements the mobile environment poses for peer-to-peer applications and consider those requirements in our application design.

Longer term objective for the research is to develop a peer-to-peer framework over which different kinds of mobile services can be deployed without providing centralized service architecture in the network. This framework should provide service discovery and service connection services for various overlying applications.

5.2 Results

Compared to other mobile peer-to-peer research, where the research has often focused on the peer-to-peer protocols used in the fixed Internet, and on modification of these protocols to be used in the mobile environment, we present a unique way of integrating a peer-to-peer network model on top of IMS networks. We have considered the special requirements of the mobile environment in our research, and built the application architecture considering how to meet those requirements best.

In our architecture, SIP is reused as the peer-to-peer signaling protocol, i.e., for uploading resource info from a mobile client to a super-peer, for searching resources in the peer-to-peer network, and for initiating resource connections, e.g., file transfers between mobile peers.

We present a SIP based hybrid peer-to-peer application architecture where the SIP Application Server (SIP AS) functions as a peer-to-peer super-peer. We show that the hybrid peer-to-peer architecture is the best fit for mobile peer-to-peer network as it minimizes overhead in mobile nodes and as it allows the mobile operator to have control on its users – even in multi-operator environment. Indeed, other mobile peer-to-peer research projects presented in this paper are also mostly based on hybrid or centralized peer-to-peer architectures, due to these architectures' low overhead in the mobile nodes.

Finally, we present measurement results on the application performance and compare these results to user requirements acquired from the user survey. Comparing these two, we see that our application satisfies the user requirements for the application performance.

5.3 Further Discussion

We presented a SIP enabled peer-to-peer service framework that can be used as the basis for a multitude of peer-to-peer services. Our framework is not limited

to one type of peer-to-peer service but it provides generic tools for resource advertisement, resource discovery, and resource connection.

Use of mobile peer-to-peer services can provide mobile operators cost savings as the service infrastructure is distributed among the end-nodes. The operator might have to run a super-node, but the infrastructure costs of running a simple super-node, compared to running the whole service based on the traditional client-server model, are minimal.

One potential issue in low-bandwidth environments is that the SIP is not the most efficient protocol as it is text based and as it has many mandatory protocol fields. If messaging overhead of SIP is considered as too large, use of binary SIP or SigComp should be considered.

5.4 Future Research Possibilities

As people are sharing personal and private information in peer-to-peer systems they might want to control who has access to this information; this is also shown in the results of our mobile peer-to-peer survey [20]. Traditional centralized access control and group management techniques cannot be directly applied to inherently distributed peer-to-peer networks. Some interesting questions regarding group management are: who is controlling the group, is there one controller or is control distributed among many peers, who is authenticating peers that want to join the group, etc.

An interesting aspect, especially from mobile point-of-view, is caching in peer-to-peer networks. If a mobile device is hosting a resource that becomes immensely popular it might have problems serving it to large crowds. Usually peer-to-peer networks tackle this problem so that all peers who get the resource are also sharing it. However, due to intermittent nature of mobile connections, it might be useful to cache this popular material in nodes that are located in the fixed network and that have fast connections. Cache servers might also improve general scalability of mobile peer-to-peer networks.

Last, continuing on the research we did in [20], it would be interesting to perform user tests with real mobile peer-to-peer applications to see how users use peer-to-peer applications in the mobile environment, and how this differs from the usage of peer-to-peer applications in the fixed network.

References

1. **Cho, Kenjiro, Fukuda, Kentsuke, Esaki, Hiroshi and Kato, Akira.** The Impact and Implications of the Growth in Residential User-to-User Traffic. *Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications.* 2006, pp. 207 - 218.

2. **ipoque.** Internet Study 2007. [Online] [Cited: February 11, 2008.] http://www.ipoque.com/media/internet_studies/internet_study_2007.

3. **Howe, Anthony J.** Napster and Gnutella: a Comparison of two Popular Peer-to-Peer Protocols. s.l. : University of Victoria, February 28, 2002.

4. **Ripeanu, Matei.** Peer-to-Peer Architecture Case Study: Gnutella Network. *First International Conference on Peer-to-Peer Computing (P2P'01).* 2001.

5. **Cohen, Bram.** Incentives Build Robustness in BitTorrent. *Workshop on Economics of Peer-to-Peer Systems (P2PEcon'03).* 2003.

6. **Baset, Salman A. and Schulzrinne, Henning.** An Analysis of the Skype Peer-to-Peer Internel Telephony Protocol. *IEEE Infocom 2006.* 2006.

7. **Joost.** Joost - The new way of watching free, full-screen, high-quality TV on the internet. [Online] [Cited: August 29, 2007.] http://joost.com/.

8. **P2PSIP.** *P2PSIP.* [Online] [Cited: August 29, 2007.] http://www.p2psip.org/.

9. **Rubenstein, Dan and Sahu, Sambit.** Can Unstructured P2P Protocols Survive Flash Crowds. *IEEE/ACM Transactions on Networking.* June 2005, Vol. 13, 3, pp. 501-512.

10. SymTorrent. *Budapest University of Technology and Economics, Department of Automation and Applied Informatics.* [Online] [Cited: November 7, 2007.] http://symtorrent.aut.bme.hu/.

11. Symella. *Budapest University of Technology and Economics, Department of Automation and Applied Informatics.* [Online] [Cited: November 10, 2007.] http://symella.aut.bme.hu/.

12. **Oberender, Jens O., Andersen, Frank-Uwe, de Meer, Hermann, Dedinski, Ivan, Hoßfeld, Tobias, Kappler, Cornelia, Mäder, Andreas and Tutschku, Kurt.** Enabling Mobile Peer-to-Peer Networking. *Wireless Systems and Mobility in Next Generation Internet, Lecture Notes in Computer Science.* 2005, Vol. 3427, pp. 219 - 234.

13. **Lehtinen, Juuso.** *Design and Implementation of Mobile Peer-to-Peer Application.* Department of Electrical and Communications Engineering, Helsinki University of Technology. 2006. MSc Thesis.

14. **Lehtinen, Juuso.** Towards Intelligent Multi-Access: Autonomic Network Interface Selection in Mobile Environment. *S-38.4030 Postgraduate Course on Networking Technology.* 2006.

15. **Lehtinen, Juuso.** Secure and Mobile Peer-to-Peer File Sharing. *Publications in Telecommunications Software and Multimedia.* TML-C21, 2006, Mobile Communities - Seminar on Internetworking.

16. **Matuszewski, Marcin, Beijar, Nicklas, Lehtinen, Juuso and Hyyryläinen, Tuomo.** Content Sharing in Mobile P2P Networks: Myth or Reality? *International Journal of Mobile Network Design and Innovation.* 2006, Vol. 1, 3-4, pp. 197-207.

17. **Beijar, Nicklas, Matuszewski, Marcin, Lehtinen, Juuso and Hyyryläinen, Tuomo.** Mobile Peer-to-Peer Content Sharing Services in IMS. *The International Conference on Telecommunication Systems, Modeling and Analysis 2005, ICTSM 2005.* 2005.

18. **Matuszewski, Marcin, Beijar, Nicklas, Lehtinen, Juuso and Hyyryläinen, Tuomo.** Mobile Peer-to-Peer Content Sharing Application. *3rd IEEE Consumer Communications and Networking Conference (CCNC 2006).* 2006, Vol. 2, pp. 1324-1325.

19. **Matuszewski, Marcin, Garcia-Martin, Miguel A., Beijar, Nicklas and Lehtinen, Juuso.** Resource Sharing and Discovery on Top of IMS. *4th IEEE Consumer Communications and Networking Conference (CCNC 2007).* January 2007, pp. 484-490.

20. **Matuszewski, Marcin, Beijar, Nicklas, Lehtinen, Juuso and Hyyryläinen, Tuomo.** Understanding Attitudes Towards Mobile Peer-to-Peer Content Sharing Services. *IEEE International Conference on Portable Information Devices.* 2007, pp. 1-5.

21. **Giesler, Markus.** Consumer Gift Systems. *Journal of Consumer Research.* 2006, Vol. 33, pp. 283-290.

22. IEEE International Conference on Peer-to-Peer Computing. *P2P.* [Online] [Cited: October 30, 2007.] http://www.ida.liu.se/conferences/p2p/portal/.

23. International workshop on Peer-To-Peer Systems. *IPTPS.* [Online] [Cited: October 30, 2007.] http://www.iptps.org/.

24. IEEE International Conference on Pervasive Computing and Communications. *PerCom.* [Online] [Cited: October 30, 2007.] http://www.percom.org/.

25. IEEE International Parallel and Distributed Processing Symposium. *IPDPS.* [Online] [Cited: October 30, 2007.] http://www.ipdps.org/.

26. IEEE Consumer Communications & Networking Conference. *CCNC.* [Online] [Cited: October 30, 2007.] http://www.ieee-ccnc.org/.

27. Peer-to-Peer Research Group. *Internet Research Task Force.* [Online] [Cited: November 1, 2007.] http://www.irtf.org/charter?gtype=rg&group=p2prg.

28. **Schollmeier, Rüdiger.** A Definition of Peer-to-Peer Networking for the Classification of Peer-to-Peer Architectures and Applications. *First International Conference on Peer-to-Peer Computing (P2P'01).* 2002.

29. **Milojicic, Dejan S., Kalogeraki, Vana, Lukose, Rajan, Nagaraja, Kiran, Pruyne, Jim, Richard, Bruno, Rollins, Sami and Xu, Zhichen.** Peer-to-Peer Computing. s.l. : HP Laboratories, Palo Alto, 2002.

30. **Steinmetz, Ralf and Wehrle, Klaus, [ed.].** *Lecture Notes in Computer Science: Peer-to-Peer Systems and Applications.* s.l. : Springer Berlin / Heidelberg, 2005. Vol. 3485/2005. ISBN: 978-3-540-29192-3.

31. **Risson, John and Moors, Tim.** Survey of research towards robust peer-to-peer networks: search methods. *Computer Networks: The International Journal of Computer and Telecommunications Networking.* 2006, Vol. 50, 17, pp. 3485 - 3521.

32. **Vanthournout, Koen, Deconinck, Geert and Belmans, Ronnie.** A taxonomy for resource discovery. *Personal and Ubiquitous Computing.* 2005, Vol. 9, 2, pp. 81 - 89.

33. **Gkantsidis, Christos, Mihail, Milena and Saberi, Amin.** Hybrid search schemes for unstructured peer-to-peer networks. *Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies.* 2005, Vol. 3, pp. 1526 - 1537.

34. **Backx, Peter, Wauters, Tim, Dhoedt, Bart and Demeester, Piet.** A comparison of peer-to-peer architectures. *Eurescom Summit 2002.* 2002.

35. **Kalogeraki, Vana, Gunopulos, Dimitrios and Zeinalipour-Yazti, D.** A local search mechanism for peer-to-peer networks. *Proceedings of the eleventh international conference on Information and knowledge management.* 2002, pp. 300 - 307.

36. **Yang, Beverly and Garcia-Molina, Hector.** Efficient Search in Peer-to-Peer Networks. *International Conference on Distributed Computing Systems.* 2002.

37. **Lv, Qin, Cao, Pei, Cohen, Edith, Li, Kai and Shenker, Scott.** Search and Replication in Unstructured Peer-to-Peer Networks. *Proceedings of the 16th international conference on Supercomputing.* 2002, pp. 84 - 95.

38. **Chang, Nicholas B. and Liu, Mingyan.** Controlled Flooding Search in a Large Network. *IEEE/ACM Transactions on Networking.* April 2007, Vol. 15, 2, pp. 436-449.

39. **Joung, Yuh-Jzer, Fang, Chien-Tse, and Yang, Li-Wei.** Keyword Search in DHT-Based Peer-to-Peer Networks. *IEEE Journal on Selected Areas in Communications.* January 2007, Vol. 25, 1, pp. 46-61.

40. **Xie, Haiyong, Krishnamurthy, Arvind, Silberschatz, Avi and Yang, Richard Yang.** P4P: Explicit Communications for Cooperative Control Between P2P and Network Providers. 2007.

41. **Bakos, Balázs, Csúcs, Gergely, Farkas, Lóránt and Nurminen, Jukka K.** Peer-to-peer protocol evaluation in topologies resembling wireless networks. An experiment with Gnutella query engine. *The 11th IEEE International Conference on Networks (ICON2003).* 2003, pp. 673-680.

42. **Maibaum, Nico and Mundt, Thomas.** JXTA: A Technology Facilitating Mobile Peer-To-Peer Networks. *Proceedings of the International Workshop on Mobility and Wireless Acces (MOBIWAC).* 2002.

43. **Sambasivan, Veena and Ozturk, Yusuf.** Network Memory Among Mobile Devices. *IEEE International Conference on Pervasive Services.* 2007, pp. 153-156.

44. **Raivio, Yrjö.** A Peer-to-Peer Overlay Architecture for Mobile Networks. *T110.7190 Research Seminar on Telecom Software.* 2005.

45. **Hu, Tim Hsin-ting, Thai, Binh and A., Seneviratne.** Supporting mobile devices in Gnutella file sharing network with mobile agents. *Eighth IEEE International Symposium on Computers and Communication (ISCC'03).* 2003, pp. 1035-1040.

46. **Huang, Chung-Ming, Hsu, Tz-Heng and Hsu, Ming-Fa.** Network-aware P2P file sharing over the wireless mobile networks. *IEEE Journal on Selected Areas in Communications.* January 2007, Vol. 25, 1, pp. 204-210.

47. **Sun microsystems.** JXTA Technology - Overview. [Online] [Cited: February 16, 2008.] http://www.sun.com/software/jxta/.

48. **Blundo, Carlo and De Cristofaro, Emiliano.** A Bluetooth-Based JXME Infrastructure. *9th International Symposium on Distributed Objects, Middleware, and Applications.* 2007.

49. WhatIsJxta. *JXTA Wiki.* [Online] [Cited: February 17, 2008.] http://wiki.java.net/bin/view/Jxta/WhatIsJxta.

50. **Kellerer, Wolfgang, Despotovic, Zoran, Michel, Maximilian, Hofstatter, Quirin and Zols, Stefan.** Towards a Mobile Peer-to-Peer Service Platform. *Proceedings of the 2007 International Symposium on Applications and the Internet Workshops.* 2007.

51. **Harjula, Erkki, Ylianttila, Mika, Ala-Kurikka, Jussi, Riekki, Jukka and Sauvola, Jaakko.** Plug-and-play application platform: towards mobile peer-to-peer. *Proceedings of the 3rd international conference on Mobile and ubiquitous multimedia.* 2004, pp. 63 - 69.

52. **Howie, Douglas, Harjula, Erkki, Ala-Kurikka, Jussi and Ylianttila, Mika.** Harnessing SIP for autonomous mobile peer-to-peer networking. *IEEE Global Telecommunications Conference (GLOBECOM '05).* 2005.

53. **Kotilainen, N., Weber, M., Vapa, M. and Vuori, J.** Mobile Chedar - a peer-to-peer middleware for mobile devices. *Third IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom 2005).* 2005, pp. 86-90.

54. **Auvinen, A., Vapa, M., Weber, M., Kotilainen, N. and Vuori, J.** Chedar: peer-to-peer middleware. *20th International Parallel and Distributed Processing Symposium (IPDPS 2006).* 2006.

55. **Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M. and Schooler, E.** RFC 3261 - SIP: Session Initiation Protocol. s.l. : IETF, June 2002. RFC3261.

56. **Johnston, Alan B.** *SIP: Understanding the Session Initiation Protocol.* 2nd Edition. s.l. : Artech House Publishers, 2003.

57. **Singh, Kundan and Schulzrinne, Henning.** Peer-to-peer internet telephony using SIP. *International Workshop on Network and Operating System Support for Digital Audio and Video.* 2005, pp. 63 - 68.

58. **Bryan, D., Matthews, P., Shim, E., Willis, D.** Concepts and Terminology for Peer to Peer SIP. *draft-ietf-p2psip-concepts-01.* November 15, 2007.

59. **Matuszewski, Marcin and Kokkonen, Esko.** Mobile P2PSIP - Peer-to-Peer SIP Communication in Mobile Communities. *5th IEEE Consumer Communications and Networking Conference, 2008. CCNC 2008.* January 2008, pp. 1159-1165.

60. **Harjula, Erkki, Ala-Kurikka, Jussi, Howie, Douglas and Ylianttila, Mika.** Analysis of Peer-to-Peer SIP in a Distributed Mobile Middleware System. *Global Telecommunications Conference (GLOBECOM 2006).* 2006.

61. **Garcia-Martin, Miguel A., Matuszewski, Marcin, Beijar, Nicklas and Lehtinen, Juuso.** Sharing Files with the Session Initiation Protocol (SIP). *draft-garcia-sipping-file-sharing-framework-01.* [Internet-Draft]. November 16, 2007.

62. **Hyyryläinen, Tuomo.** *Mobile P2P Client Implementation on Symbian.* Department of Electrical and Communications Engineering - Networking Laboratory, Helsinki University of Technology. 2006. Special Assignment.

63. **CafeSip.org.** Jiplet Container - a container for SIP applications. [Online] [Cited: February 16, 2008.] http://www.cafesip.org/projects/jiplet/index.html.

64. **Morales Reyes, Victor Hugo.** *Design and implementation of a distributed file directory for mobile peer-to-peer.* Department of Electrical and Communications Engineering, Helsinki University of Technology. 2007.

65. **Garcia-Martin, Miguel A. and Matuszewski, Marcin.** A Session Initiation Protocol (SIP) Event Package and Data Format for Describing Files. *draft-garcia-sipping-file-event-package-00.* [Internet-Draft]. June 8, 2007.

66. **Khartabil, H., Leppanen, E., Lonnfors, M., Costa-Requena, J.** An Extensible Markup Language (XML)-Based Format for Event Notification Filtering. *RFC 4661.* September 2006.

67. **Roach, A. B.** Session Initiation Protocol (SIP)-Specific Event Notification. *RFC 3265.* June 2002.

68. **Singh, Kundan and Schulzrinne, Henning.** Peer-to-Peer Internet Telephony using SIP. *Proceedings of the international workshop on Network and operating systems support for digital audio and video.* 2005, pp. 63-68.

69. **Daswani, Neil, Garcia-Molina, Hector and Yang, Beverly.** Open Problems in Data-sharing Peer-to-peer Systems. *ICDT 2003.* 2003.