UNIVERSITY OF TARTU
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
Institute of Computer Science
Computer Science

Pätris Halapuu

Establishing Peer-to-Peer Distributed File Sharing System With Mobile Host

Bachelors' thesis (6EAP)

Supervisors: Satish Narayana Srirama, PhD

Carlos Paniagua, MSc

Author: …….”….” May 2013

Supervisor: …….”….” May 2013

Approved for defence

Professor: …….”….” May 2013

TARTU, 2013

Contents

Glossary

PDA – Personal Digital Assistant

REST – Representational State Transfer

OSGi – Open Services Gateway initiative

J2ME – Java 2 Platform, Micro Edition

ZeroConf – Zero Configuration Networks [1]

GPS – Global Positioning System

3G, 4G – Third and Fourth Generation of mobile phone communication technology standards

SOAP – Single Object Access Protocol

JXTA – Juxtapose

URI – Uniform Resource Identifier

XML – Extensible Markup Language

JSON – JavaScript Object Notation

JAR- Java Archive file

FTP – File Transfer Protocol

DHT – Distributed Hast Table

SHA-1 – Secure Hash Algorithm

WSDL – Web Services Description Language

UDDI – Universal Description, Discovery and Integration

DNS – Domain Name System

JmDNS – Java multicast Domain Name System

CPU – Central Processing Unit

RAM – Random Access Memory

1. Introduction

Mobile devices such as tablets, PDAs, mobile phones etc. have developed rapidly during last decade and become inseparable part of people's everyday life. These devices are equipped with embedded sensors, camera, touchscreen, more memory, powerful processor, mobile 3G and 4G networks and Wi-Fi capability as well as efficient power consumption mechanisms. These improvements have led to mobile devices being able to perform tasks that usually personal computers are capable of. What is more due to release of Android OS and iOS applications for such mobile devices have increased as well as their complexity. Being online has become ubiquitous as Wi-Fi and mobile data networks are available in most of the places. For example over 45% of world's population is covered by 3G mobile network [2]. Needlessly to say that makes using web services form mobile devices a common thing. Such improvements lead to next generation of services which can be provided not only from dedicated servers but also from mobile phones.

The concept of mobile web services provisioning is not new and has been in the ground for some time. Srirama et al. proposed the concept of Mobile Host [3] in 2006 where the mobile device acts as service provider. Mobile Host enables seamless integration of user specific services to the enterprise by following web service standards, also on the radio link and via resources constrained smart phones. Moreover Mobile Host fosters the new generation of ubiquitous and context-aware applications enabling the consumption of web services anywhere at any time from the handset.

Mobile Host has been updated to latest technologies like for example REST architecture which replaced SOAP so web services would be focused on systems resources. Early versions of Mobile Host were developed in PersonalJava and J2ME and meant for Symbian devices but as now the biggest share in smart phones market is held by Android then Mobile Host was upgraded for that platform by Paniagua [4] in 2012.

Contribution of this thesis was to develop Peer-to-Peer (P2P) distributed file sharing system to Mobile Host for Android. This feature comes in handy when we talk about services that enables file sharing. As Mobile Host by its nature can join or leave network at any moment then accessing files that client is interested in becomes critical as file to be downloaded would be offered only by one provider as in regular client-server architecture. P2P distributed file sharing capability for Mobile Host provides users more reliable file sharing environment in distributed manner as files are downloaded as pieces from all the

online peers who have pieces of desired file. What is more, small metadata torrent files are hosted by Mobile Hosts and published as a service. This setup assures the independency from other platforms and hosts.

1.1 Outline

Chapter 2: discusses the state of the art addressed by this thesis. Chapter first introduces concept and architecture of Mobile Host proposed by Srirama et al. and developed by Paniagua. It describes how to write new services to Mobile Host and then moves on to explaining Peer-to-Peer domain and introduces BitTorrent protocol. As last things, Libtorrent library, implementation on BitTorrent protocol is discussed and some of the related works are pointed out.

Chapter 3: describes how Mobile Host publishes services, connects and works using two different approaches – ZeroConf network and P2P based network. This chapter also describes how Mobile Host can publish and discover advertisements in Wide Area Networks.

Chapter 4: discusses different P2P protocols and their implementation. Also suitability of those protocols and implementations are being explained and finally pointed out why Libtorrent was chosen for this thesis.

Chapter 5: describes the contribution of the thesis by firstly going over the aim of this thesis then giving sample scenario where this program could come in handy and then moving to describing some of the key features of work that has been done with some snapshots of application code. Final part of this paragraph is about performance analysis.

Chapter 6: provides the conclusions about the findings of the thesis.

Chapter 7: describes the future work directions such as improving search engine to cloud based platform, adding more features to distributed file sharing client. This chapter also proposes to improve layout design to make it more user-friendly.

2. State of the Art

During last decade extensive research has been done in mobile web services provisioning domain. Concept of Mobile Host has been proposed by Srirama et al. [3] in 2006 for resource constrained smart phones to act as both service providers and clients in mobile networks. Several challenges arise when establishing such Mobile Enterprises – for example discovery aspects, and quality of services.

Mobile Host acts as both service provider and client. Smart phone acting as mobile web service client is common – some of the interesting web service applications like weather forecast, company news, information search etc. are accessed from smart phones daily. Though mobile devices acting as web service client is common [5], the domain of mobile web service provisioning is not and was first studied at RWTH Aachen University since 2003 [19].

Mobile devices are capable of providing different data to user like GPS, network information etc. That data can be used during runtime to provide users most relevant services like map specific context aware services.

Huge number of different web services being provided by Mobile Host in wireless networks is possible and that makes the discovery of services quite complex. The mobile nature of smart phones brings challenges like addressability, reachability and reliability. These issues have been addressed as Mobile Host has been upgraded to technologies such as ZeroConf and OSGi by Paniagua [4] in 2012.

2.1 Mobile Host

Developments in web service specifications protocols standardization and latest developments in mobile devices domain has led to new types of web services provided by mobile devices. Smart phones today have different sensors, camera(s), touch screen, GPS capability, enough computing power etc. to provide different customized and specific applications and services to users. What is more, due to ubiquity of web access as 3G and 4G mobile communications technology has developed and widely spread [2] and Wi-Fi is accessible in most common daily visited places like in restaurants, home, work, conferences etc. comes possibility not only to be client of mobile web services but also to provide them.

The early concept of Mobile Host was proposed by Srirama et al. [3] in 2006 where mobile device acts as web service provider. Its firs implementation was built on top of a

normal Web server working as web service handler being light weight as it was meant for resource constrained smart phones. Mobile Host used SOAP architecture where it listened for incoming GET/POST HTTP requests on a server socket. First implementation was developed in PersonalJava and tested on Sony Ericsson P800 smart phone.

Discovery for Mobile Host was realized using another interesting approach – using JXTA open source P2P protocol specification where virtual P2P network was established. In such network services are published to others as JXTA advertisements so that they can be sensed as JXTA services among other peers. As described, discovery solution was implemented through mobile P2P network. [25]

2.2 Mobile Host in Android

Nowadays smart phones are using wide variety of services like location based services, mobile cloud services, mobile web services etc. from different providers. Considering latest developments, a new type of services provider comes in – mobile phone users themselves with help of Mobile Host. Latest version of Mobile Host was developed for Android OS taking into account that it has the biggest market share in smart phones operating systems. Along with upgrading to Android OS other improvements were made. SOAP architecture was replaced with REST architecture, ZeroConf was used, OSGi framework was used as module system and services platform to handle components life cycle without reboot and of course when talking about Android applications development, Android Software Development Kit was used when establishing Server Sockets communication so HTTP requests could be handled.

Apache Felix, implementation of the OSGi R4 Service Platform was used in developing Mobile Host. OSGi framework is service platform for Java programming language that implements a complete and dynamic component model. Components in the form of bundles can be remotely installed, started, sopped, updated and uninstalled without the necessity of reboot. Services registry lets bundles to do the detection whether there are any changes in services (added new or removed) and adapt accordingly.

There are two different mechanisms for interacting with web services – SOAP and REST. As an improvement, SOAP architecture was replaced with REST architecture in Mobile Host for Android. For providing services from mobile devices, REST is the logical option due to its cacheable and stateless characteristics. The architecture of the Mobile Host for Android is shown in Figure 1.

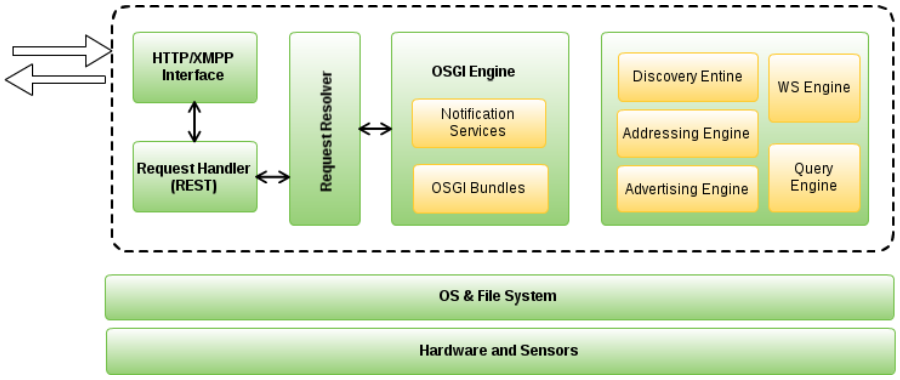

Figure 1: Architecture of Mobile Host for Android [4]

Similarly to first implementation of Mobile Host in 2006, Mobile Host for Android listens for HTTP as well as XMPP connections. When HTTP/XMPP interface receives connection then it is passed to Request Handler which does the parsing of the request and its parameters. According to RESTful philosophy, entities, collections, services and other key recourses are identified by assigned URI. The standard methods are mapped to resource-specific semantics where all the resources implement the same uniform interface. This enables client the following way of requesting through GET HTTP request: document "doc.docx" accessed in URI "documents/doc.docx". Request is passed to the Request Resolver when the REST handler has parsed the request. Request Resolver then communicates with the OSGi engine for resolving the request.

As a next step, OSGi Engine comes in – Request Resolver gets an instance of the service which is running in the OSGi Engine. Once service instance is acquired, it invokes the method corresponding to the HTTP request. For example if the HTTP Request is sent as a GET then the Request Resolver will invoke the *doGet* method of the service and the control gets passed to the OSGi Service. Then the OSGi Service implements the service logic like retrieving documents, pictures, contacts, GPS location etc. Response is also prepared in OSGi service that later gets delivered to client. Response can have any format (XML, JSON, plain text etc.) or mime type according to the logic of the service. As a final step, response is written to the socket according to HTTP or XMPP protocol by OSGi Service.

2.3 Writing a New Service

Mobile Host enables the development of next generation applications for mobile devices which can be context aware, location based etc. For example location (GPS) data provi-

sioning service has been implemented in Mobile Host for Android which passes GPS sensors data when requested.

The architecture of Mobile Host makes it easy to develop new services. All the services in Mobile Host are OSGi Services. All OSGi Services are deployed as bundles that need to be registered in OSGi engine so they could be called out. OSGi Service provides information about service (service name etc.) to OSGi Engine in the registration process. Request Resolver later uses that name for invocation. Two Java interfaces must be implemented by OSGi Service: *BundleActivator* and *SroidService*. The *BundleActivator* interface contains methods required to start, stop and register the service in the OSGi Engine. The *SroidService* interface contains methods for the service provisioning that are done in RESTful fashion guaranteeing that the OSGi Services can handle GET, PUT, DELETE and POST HTTP methods. In *SroidService* interface the method *doCreate* can be used for any variable or process initialization.

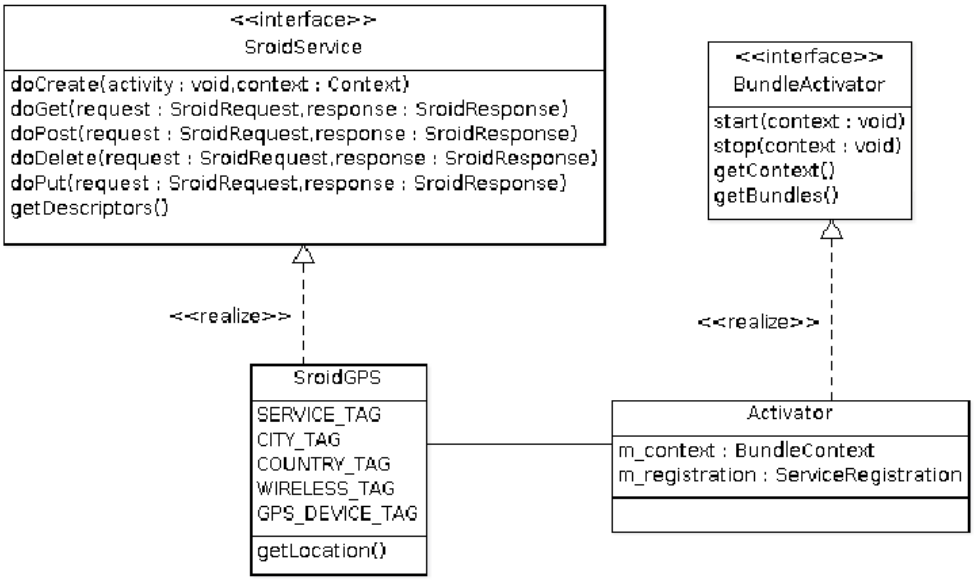

Figure 2: Class Diagram for a GPS Mobile Host Service [4]

For handling requests, *SroidRequest* Java Class is provided by Mobile Host. It encapsulates the logic for parsing the HTTP request and maps the parameters to *HashMap* which is accessible from the OSGi Services. For handling responses *SroidResponse* Java Class is provided by Mobile Host. This class similarly to *SroidRequest* encapsulates the logic for writing information to the socket previously established by the client. Class dia-

gram that describes the classes and relationships necessary for creating sample GPS mobile web service for Mobile Host in Android is shown in Figure 2. Any other mobile web service can be developed for Mobile Host in Android following given structure.

New OSGi services can be deployed as JAR files that must contain the implementation of Activator and *SroidService* interfaces. Mobile Host takes JAR files from the folder *SroidServices* in the Android file system and deploys them in OSGi Engine. After that they are ready to be used.

To broaden features of Mobile Host (main feature is web services provisioning as discussed before), file sharing in distributed fashion capability was implemented. For that P2P architecture was studied and one of its protocols – BitTorrent was used.

2.4 Peer-to-Peer domain

Peer-to-Peer (P2P) network architecture consists of group of nodes (peers) who are able to connect to each other and are able to share resources [6]. In this kind of network every device is in a role of peer and can act as both client and server [7]. As name states, P2P network does not need central server to communicate like traditional client-server architecture like for example traditional FTP service [8]. Internet Protocol (IP) is used in most cases for data exchange. There are direct edges in this kind of network between any two nodes that know each other [9]. The Figure 3 illustrates how P2P network is made up.

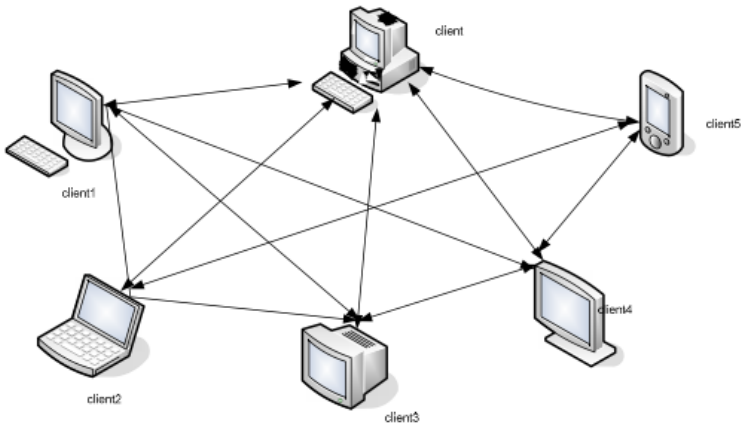

Figure 3: P2P sample network [23]

2.4.1 BitTorrent protocol

BitTorrent protocol is one of the most common P2P file sharing protocols that was designed and first implemented in the spring of 2001 by Bram Cohen. It is used for reducing network and server impact of distributing large files as peers in this network connect to "swarm" of other peers to download and upload from each other at the same time. BitTorrent protocol can also work successfully over low bandwidth networks so many mobile devices like smart phones can efficiently distribute files to others [10]. Usage of that kind of protocol can help replacing large file servers while efficiently distributing files to many recipients and as it has low bandwidth usage, large spikes in internet traffic in certain area can be prevented. [11]

Creating and sharing files is easy – user who wants to share file(s) has to make small descriptor file called torrent and share it to others via web, email etc. As file is made, user has to start BitTorrent client and become BitTorrent node to this file, start acting as seed, seeding the file. When other users have the small torrent file they can become nodes with their BitTorrent client and start acting as peer or leecher downloading it by connecting to others nodes (called seed(s) and peers) [10].

Torrent file divides the file being sent into segments that are called pieces. When user receives a new piece it starts to share that same piece to others at the same time so initial file owner does not have to send all the pieces to all the peers. That helps to lower the bandwidth and increase reliability of getting the file. Number of peers who can be peers is unlimited by BitTorrent protocol.

BitTorrent protocol specification [12] states that data requests are made over different TCP connections to different machines. In case of classic downloading requests are usually made via single TCP connection to only one machine. BitTorrent also downloads in a random or "rarest-first" [13] approach to ensure high availability, contrary classic client-server downloads are sequential. BitTorrent protocol specification also defines that pieces contained in descriptor are cryptographically hashed (SHA-1 is used) to ensure any modification of the pieces could be reliably detected.

The torrent file can hold different metadata. These files have information about file name, announcer/tracker URL section, info section containing file(s) name(s), file(s) size(s), pieces length, SHA-1 hash code for each piece etc. This information is used by BitTorrent client to verify integrity of the data. [14]

BitTorrent protocol uses also a server called tracker. It helps to communicate between peers to initiate downloads and for updating knowledge about new peers being only major critical point. Once the information from tracker is obtained, communication between peers can go on without a tracker [15]. Some of the most popular public tracers are OpenBitTorrent and PublicBitTorrent.

2.4.2 Libtorrent library

Libtorrent is one of the implementations of BitTorrent protocol. It is open source and its first release was in September 2005. Libtorrent is written in C++ programming language. It is kept up to date with most of the BitTorrent extensions. Most notable features of Libtorrent are support for Mainline DHT, IPv6, HTTP seeds and µTorrent's peer exchange. It is being constantly optimized to work in more different environments to be the most suitable BitTorrent protocol implementation for mobile devices as well as desktops and seed-servers [16]. To gain platform independence, Libtorrent uses Boost.Asio – open source cross-platform C++ library for network programming providing a consistent asynchronous I/O model using a modern C++ approach [17].

2.5 Related Works Regarding Mobile Host

There have been several different publications and thesis in the domain on mobile web service provisioning. First paper was published in 2003 by Pratistha et al. [18] where it was proposed an infrastructure that provides the capability of hosting web services on mobile devices.

Srirama [19] developed the first version of Mobile Host using PersonalJava as part of his Master thesis. In 2009 Kim and Lee [20] proposed a lightweight framework that hosts web services on mobile devices and supports service migration. In 2011 Asif and Majumdar [21] discussed partitioning frameworks for mobile web service provisioning. Later in 2012 Paniagua [4] updated Mobile Host to latest technologies like REST, OSGi, ZeroConf and Android OS as part of his Master thesis.

In Paniagua's thesis [4], there is written about P2P protocol JXTA [22] and about identifying peers through JXTA network is described. However, peerdroid – JXTA implementation for Android is not suitable P2P library for establishing distributed file sharing system.

In Filbert's paper [23] a Peer-to-Peer system has been described briefly and as a research work, developed on Android platform using peerdroid [24] and JXTA v2.5. How-

ever Filbert's program was developed as an independent program and file sharing is not distributed. In this paper peer-to-peer system is developed as part of Mobile Host web service provisioning system and file sharing is distributed as BitTorrent protocol declares.

2.6 Summary

The developments in Mobile Host, web services domain, P2P domain, improvements in smart phones and improved transmission capabilities in cellular networks have led to domain of mobile web services with capability of distributed file sharing. This chapter summarized the research associated in this domain. Also related works have been briefly described and pointed out in which way this work is different.

3. Discovery of Services

Discovery of services in Mobile Host plays crucial part of the whole system. Usually web services' publishing is done using WSDL advertisements in UDDI registry. Considering Mobile Host, such centralized solution is not the best idea because number of services might grow big and mobile networks are quite dynamic due to the movement of nodes. What is more, clients are most likely to change their status in network by joining, leaving or switching operator making the binding information in the WSDL documents outdated. This makes it necessary to have service discovery that is dynamic. This challenge has been addressed by Carlos [4] using two different mechanisms for provided services: directory-based with overlay support discovery which is P2P based and directory-less with overlay support using ZeroConf.

3.1 Discovery in P2P Domain

Early implementation of Mobile Host used JXTA P2P networks for indexing, advertising and addressing the services [25]. In this approach when virtual network is established then services that are provided by Mobile host are published as JXTA advertisements. These advertisements contain usual WSDL information about the services and are language-neutral. Advertisements are exchanged when peers discover each other and caching may be done locally. Advertisements in such form have lifetime that specifies the availability of the resources – service and network. For services that are available longer than their life-time advertised before, they are to be republished. This gives the possibility of having only up to date services available.

3.2 Discovery in ZeroConf Network

New approach was implemented in Mobile Host for Android using ZeroConf. It configures hosts automatically assigning dynamically an IP address and domain name. Also service discovery mechanism and domain resolution is provided by ZeroConf. It makes it easier for users as they do not have to assign host names nor IP addresses and can query for available services in the network. JmDNS which gives a local domain name to the device is used in the implementation of Mobile Host – it is also totally compatible with other ZeroConf platforms such as Bonjour for Apple. No DNS server with global knowledge is needed as ZeroConf has name-to-address translation capability relying on Multicast DNS that sends DNS queries over local network using IP multicast. Service discovery in Zero-Conf enables devices to find all available instances of certain type of service and maintain the service directory. Service name is resolved to IP address and port number when found

for service invocation. Layer for indirections between IP address with port and service is provided by service directory. It enables to keep up to date list of available services. To accomplish service discovery, a Multicast DNS query is sent to a given type of domain and reply from all the matching services is received and listed in the local service directory.

3.3 Discovery in Wide Area Network

Wide Area Bonjour based on DNS Service Discovery is used in Mobile Host to enable services to be discovered and self-configured in Wide Area Network. A few DNS records are added to a DNS server in order to advertise a set of services. One can set up small DNS server by himself but there is also possibility to use some provider's services like Dyn DNS [26]. Three types of record sets are necessary: to make the domain browsable, to list the services entities and to describe each named service defined in services entities. The Figure 3 shows the DNS records for advertising a web service in DNS Service Discovery. From the Figure 3, (1) enables the DNS dynamic updates, (2) and (3) enables the domain to be discovered, (4) sets the domain to be chosen as default, (5) enables domain to be show up in list of potential registration domains, (6) to be chosen as default registration domain and (7) to enable clients with empty string requests to brows not only the "local" zone. [4] [26] [27]

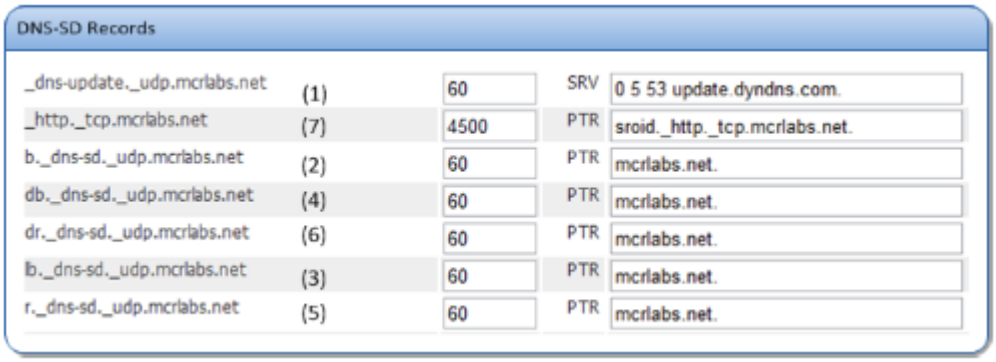

Figure 4: DNS Service Discovery records in dyn.com [4]

3.4 Improved Discovery with Apache SOLR

JXTA provides keyword based search mechanism for discovering services that are advertised in P2P network. This basic search also is able to handle WSDL information returning large number of services. That approach is not the best for smart phones. To make the search more effective, Apache SOLR 3.6.0 was used by Paniagua [4] for indexing data,

doing the full-text search, hit highlighting etc. Apache SOLR is open source search platform based on Apache Lucene Project [28].

3.5 Related Works Regarding Discovery

Discovery plays crucial part in any provisioning domain. There have been many different studies carried out, papers published and thesis written regarding discovery of web services. First publication relevant to this thesis was published in 2003 by Yang et al. [29]. In this paper it was proposed an infrastructure for organizing and efficiently accessing mobile web services in broadcast environments that defines a multi-channel model to carry information about mobile web services [30]. In early implementation of Mobile Host in 2004 Srirama [30] targeted discovery problem with using WSDL and UDDI. In 2006 Srirama [31] published paper where it was proposed mobile web services discovery mechanism with P2P technology using JXTA. In 2009 Steller et al. [32] proposed the mTableaux algorithm for optimizing the reasoning process and facilitate web services discovery [30]. In 2012 Paniagua [4] improved Mobile Host discovery mechanism in his Maser thesis using ZeroConf network and Apache SOLR.

3.6 Summary

This chapter describes the Mobile Host web services discovery mechanisms using JXTA P2P based approach and ZeroConf capabilities. Early implementation of service discovery and advertising in Mobile Host is explained as well as the Mobile Host for Android implementation from the service discovery and publishing aspect. Both local discovery and global addressing have been explained in this chapter. Related works considering discovery has been also pointed out in this chapter.

4. Case Study

Peer-to-peer file sharing is not new and has been in the ground for some time. There are many different P2P protocols for example JXTA, BitTorrent, Gnutella, Freenet etc. Not all implementations of those protocols are well documented, kept up to date, maintained or have open source libraries. Neither not all of them are suitable for making P2P distributed file sharing system.

Several different P2P protocols and their implementations were studied. Peerdroid [24], JXTA P2P protocol implementation for Android was not suitable for the thesis as it only supports direct file sharing between two nodes in P2P network. That means that file sharing cannot be easily (without writing complete middleware that does it) done in distributed fashion. Another reason why JXTA based P2P protocol implementations were not good enough was that it is currently no longer maintained since last version 2.5 and its supportive community is not very active anymore based on forums, emails and StackOverflow.com. Similarly to previous sip2peer [33] was studied. Sip2peer is Session Initiation Protocol based middleware for the implementation of any P2P application. In this case also distributed file sharing was not possible.

Different BitTorrent protocol implementations were studied as well as BitTorrent specification defines the distributed file sharing capabilities. Yaircc [34], small Java BitTorrent library and torrent4j [35], Java BitTorrent library for example were both not that well documented and without proper support due to lack of community behind projects and probably due to small number of users. Same applied for studied Java BitTorrent API bitext [36] which is still in Beta since 2007.

Finally as most suitable BitTorrent library, Libtorrent was chosen. It has good documentation, active community and it is being kept up to date. Although it is written in C++ programming language and Android main programming language is Java, Android has good Java Native Interface to interact with native code written in C or C++.

Distributed P2P file sharing capability gives Mobile Host users the advantage to have file sharing service that is more reliable than usual client-server architecture on which file sharing in Mobile Host was relying before. Distributed P2P file sharing by its nature makes accessing and sharing files more comfortable and reduces the load on network bandwidth as files are divided into segments which are downloaded from all the peers who have those pieces not from only one certain provider. What is more, small metadata torrent

files are stored in Mobile Host and published as a service. That approach enables Mobile Host to stay independent from other platforms and torrent hosting sites.

5. Contribution

The aim of this thesis was to add peer-to-peer file sharing system capability to Mobile Host. As a result BitTorrent client was added to Mobile Host with also capability of making torrent files. Apache SOLR 4.2.0 was used as search engine to provide full text search and data upload.

5.1 Scenario of usage

Sample use case of developed Android application would be for example in conference where visitor makes a video of some performance and others are interested in this recorded piece. Making torrent file of it and publishing it to others is trivial with developed system. Sharing this video would be easy and network load would be balanced as downloading and uploading content would become distributed.

5.2 Making .torrent file

According to BitTorrent protocol (explained and described before) a torrent file making capability was added to Mobile Host. Static file size of 256kB was assigned for each piece. To make .torrent file, user has to give 3 arguments to *createTorrent* method: the name of torrent file; file to make the torrent of; tracker of user's choice. Example of creating torrent file can be seen in Figure 5.

```
Torrent.createTorrent(new File("/sdcard/fileshare/uploads/1.torrent"),
        new File("/sdcard/fileshare/uploads/1.pdf"),
        "http://tracker.openbittorrent.com:80/announce");
```

Figure 5: Creating torrent

Encoding and hashing is being done in Torrent class and hash info added as torrent file metadata along with all the other metadata such as file name, length, pieces length, one static announcer's URL (just in case as user input given tracker URL is invalid). Part of the code can be seen in Figure 6.

```
import java.io.ByteArrayOutputStream;

public class Torrent {
    private static void encodeObject(Object o, OutputStream out) throws IOException {
    private static void encodeLong(long value, OutputStream out) throws IOException {
    private static void encodeBytes(byte[] bytes, OutputStream out) throws IOException {
    private static void encodeString(String str, OutputStream out) throws IOException {
    private static void encodeMap(Map<String,Object> map, OutputStream out) throws IOException{
    private static byte[] hashPieces(File file, int pieceLength) throws IOException {
    public static void createTorrent(File file, File sharedFile, String announceURL) throws IOException {
        final int pieceLength = 256*1024;
        Map<String,Object> info = new HashMap<String,Object>();
        info.put("name", sharedFile.getName());
        info.put("length", sharedFile.length());
        info.put("piece length", pieceLength);
        info.put("pieces", hashPieces(sharedFile, pieceLength));
        Map<String,Object> metainfo = new HashMap<String,Object>();
        metainfo.put("announce", announceURL);
        metainfo.put("announce", "udp://tracker.istole.it:80/announce");
        metainfo.put("info", info);
        metainfo.put("comment", "Torrent made by Mobile Host");
        metainfo.put("created by", "MobileHost");
        metainfo.put("url-list", "http://apps.bittorrent.com/torrents/torrentdata/e");
        OutputStream out = new FileOutputStream(file);
        encodeMap(metainfo, out);
        out.close();
    }
}
```

Figure 6: Torrent class

5.3 Publishing and search with SOLR

Apace SOLR version 4.2.0 was used to do the searching for Mobile Hosts who are online and have P2P file sharing capability. Mobile Host with distributed file sharing capability has one service named as “torrent service”. For searching active Mobile Hosts with torrent service, SOLR client was installed to mcrlabs.net domain and as default, port 8983 was used. Mobile host uploads services data as services.txt file to this address with SOLR API. Thanks to Apache Tika integration in SOLR 4.2.0, a full text search in txt files can be done. Services file is generated right after program is started and services are published. After that *SolrCellRequestDemo.main()* is called out and services data is sent to SOLR. Sending data to SOLR server is made very simple thanks to solr-solrj.4.2.0 java library. All necessary code for uploading data to SOLR server is shown in Figure 7.

```java
package ut.ee.mh;

import java.io.File;
import java.io.IOException;

import org.apache.solr.client.solrj.SolrServer;
import org.apache.solr.client.solrj.SolrServerException;
import org.apache.solr.client.solrj.impl.HttpSolrServer;
import org.apache.solr.client.solrj.request.ContentStreamUpdateRequest;
import org.apache.solr.common.util.NamedList;

public class SolrCellRequestDemo {
      public static void main () throws IOException, SolrServerException {
        SolrServer server = new HttpSolrServer("http://mcrlabs.net:8983/solr");
        ContentStreamUpdateRequest req = new ContentStreamUpdateRequest("/update/extract");
        req.addFile(new File("/sdcard/fileshare/uploads/services.txt"), null);
        NamedList<Object> result = server.request(req);
        System.out.println("Result: " + result);
      }
}
```

Figure 7: SOLR posting data.

Example search with Apache SOLR to get all torrent service providers is following: *http://mcrlabs.net:8983/solr/collection1/select?q=services:torrent.* Collection1 is default SOLR collection (also called core) to where services.txt is added. Query "services" is selected and keyword torrent is looked up.

5.4 BitTorrent client

BitTorrent client was made using Libtorrent library. For that Libtorrent was imported to project and to interact with native code Java Native Interface (JNI) was used. C++ class was added for using Libtorrent methods in Java code. In Figure 8 is a piece of code in C++ for exporting one of the necessary methods - *GetTorrentName* method for Java to use and in Figure 9 this method is used inside Java class. The same method is later used in BitTorrent client downloader service to get torrent file names (for example in Figure 10, *AddTorrent* method uses *GetTorrentName* method). Libtorrent has large variety of different extensions that BitTorrent protocol can provide but not all Libtorrent functions were exported to this project as not all were necessary for most common distributed file sharing client, for example private torrents, SSL torrents, IP filter and so on.

```
JNIEXPORT jstring JNICALL Java_ut_ee_libtorrent_LibTorrent_GetTorrentName
    (JNIEnv *env, jobject obj, jstring TorrentFile)
{
    jstring result = NULL;
    try{
        std::string torrentFile;
        JniToStdString(env, &torrentFile, TorrentFile);

        boost::intrusive_ptr<libtorrent::torrent_info> t;
        libtorrent::error_code ec;
        t = new libtorrent::torrent_info(torrentFile.c_str(), ec);
        if (ec){
            LOG_ERR("%s: %s\n", torrentFile.c_str(), ec.message().c_str());
        }
        else{
            LOG_INFO("%s\n", t->name().c_str());
            result = env->NewStringUTF((t->name()).c_str());
        }
    }catch(...){
        LOG_ERR("Exception: failed to get torrent name");
    }
    return result;
}
```

Figure 8: C++ class for exporting Libtorrent methods

```
public class LibTorrent {
    static {
        System.loadLibrary("torrent");
    }

    public native String GetTorrentName(String TorrentFile);
```

Figure 9: Imporing Libtorrent methods

```
static public void AddTorrent(Context context, String FileName, int progress, long progressSize, int storageMode, String savePath){
    if(FileName.equals("undefined"))
        return;
    for(int i=0;i<Torrents.size();i++){
        TorrentContainer tc = Torrents.get(i);
        if(tc.Name.equals(FileName))
            return;
    }
    try {
        File file = new File(FileName);
        FileInputStream input = new FileInputStream(file);
        input.close();
    } catch (Exception e){
        return;
    }
    String contentName = DownloadService.LibTorrents.GetTorrentName(FileName);
```

Figure 10: Using Libtorrent methods

It is possible to do several different activities in BitTorrent client: change download directory, start/stop/resume/pause download, remove torrent from download list, browse file directory for torrent files, view download status and information, change session options (port, upload limit, download limit). Snapshots of the options can be viewed in Figure 11 and torrent info and operations can be viewed in Figure 13.

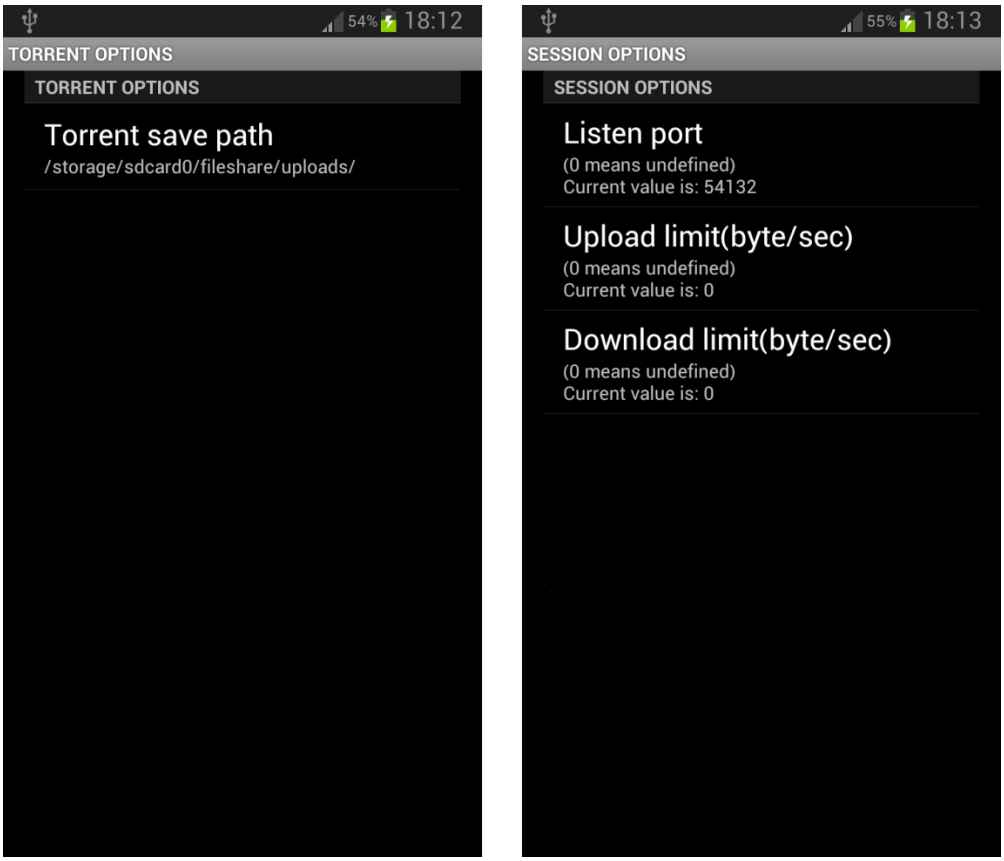

Figure 11: Options snapshots in BitTorrent client

5.5 Workflow

At first, when Mobile Host is started (let's call it Mobile Host 1 in Figure 14), by default all the services that Mobile Host is providing are started and published to Bonjour network (step 1 in Figure 14) with JmDNS - service discovery protocol which is an implementation of ZeroConf. After that, services that already exist in this network are listed in main view (see Figure 12) and services info is sent to SOLR as services.txt file.

Figure 12: Mobile Host main view

As next step, user can create new torrent file or open BitTorrent client to start downloading other torrents. To create torrent, user has to select file to make torrent file of, assign name to new torrent file and add announcer of user's choice (see step 2 in Figure 14). As torrent file is created and BitTorrent client is working, the created torrent's info is announced to tracker and swarm is created (or joined if other seeders of the same torrent file are online).

When new Mobile Host (let's call it Mobile Host 2) joins Bonjour network, same steps are taken as described in first steps (see step 3 and 1 in Figure 14). After that user can either go through main view services info to look for Mobile Hosts who are providing distributed file sharing service or do keyword based search with Apache SOLR (see step 4 in Figure 14) and get list of Mobile Hosts who provide distributed file sharing service.

When Mobile Host 2 finds that Mobile Host 1 is sharing torrents it can browse Mobile Host 1 shared file directory and download torrent file that Mobile Host 2 user would like to have (see step 5 in Figure 14).

As the last step, Mobile Host 2 can go to BitTorrent client and start downloading the file that he has torrent file of (see Figure 13) from all the users that have pieces of that file in BitTorrent swarm (see step 6 in Figure 14).

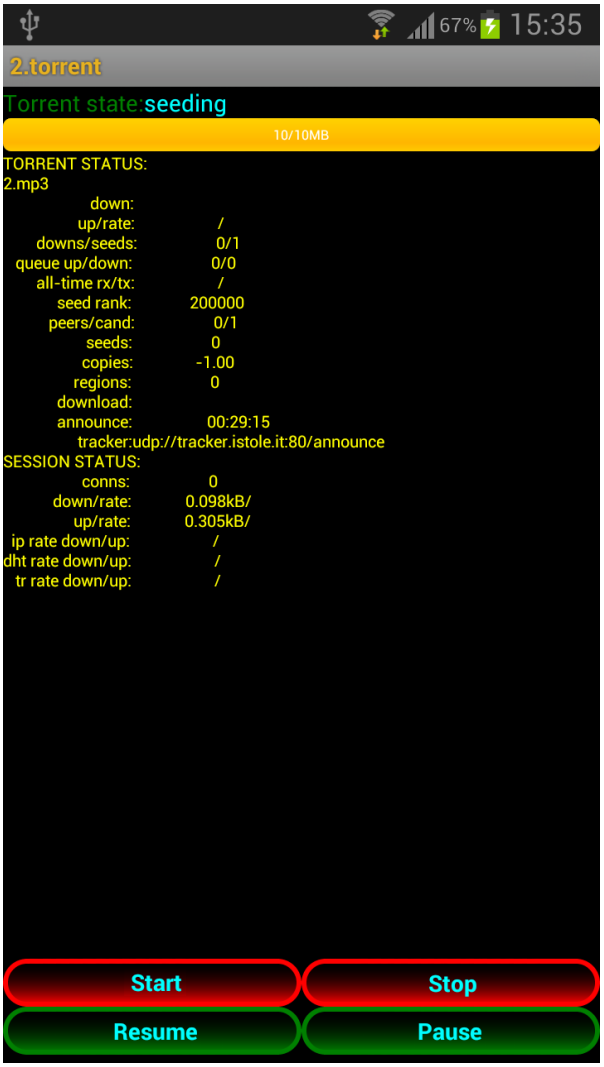

Figure 13: BitTorrent client downloading view

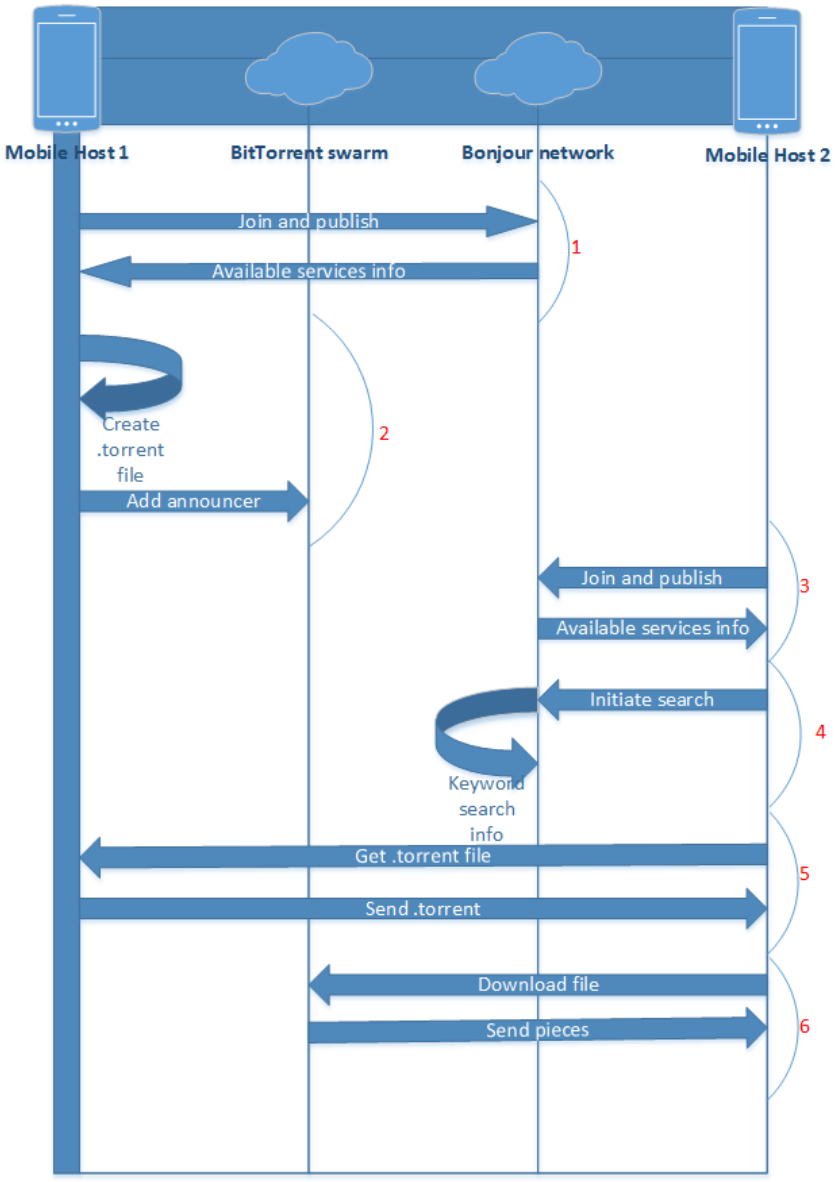

Figure 14: Workflow of discovery and P2P file sharing

5.6 Evaluations

Making torrent metadata file from the file that user wants to share plays important role in this way of sharing files as this torrent file is shared in client-server fashion. It is important that this file is as small as possible. Torrent file sizes according to file size in case of 256KB size pieces and time taking to make that torrent file can be seen Table 1.

File Size in MB	1	10	50	100	300
Torrent File Size in KB	0,3	1	4,2	8,1	23,7
Time Taken in Seconds to Create Torrent File	0,04	0,37	1,55	2,59	7,62

Table 1: Torrent making time according to file size and torrent file size accordingly

As seen from Table 1 torrent file size is almost in linear dependency from actual file size which is logical because amount of hashing done for pieces with static piece size of 256KB depends on the initial file size. Same applies for time taken to make a torrent file.

Performance analysis for SOLR was carried out with 1000 services. Average of 93 milliseconds was time taken for making the list of results in the server side. Measuring was done on the same set of services 50 times with different keywords and result of that can be seen in Figure 15. Server in what SOLR is running has following parameters: AMD Opteron CPU running at 1000.00 MHz with cache size of 1024KB, 3GB RAM, Ubuntu 11.04. Querying services in mobile device in different networks like Wide Area Network, Local Area Network or 3G mobile data network was considered not necessary because the rest of the time loading the list of services in the phone depends completely on Internet speed. The same applies for evaluating torrent sharing performance as no limitations are made by the developed feature for Mobile Host.

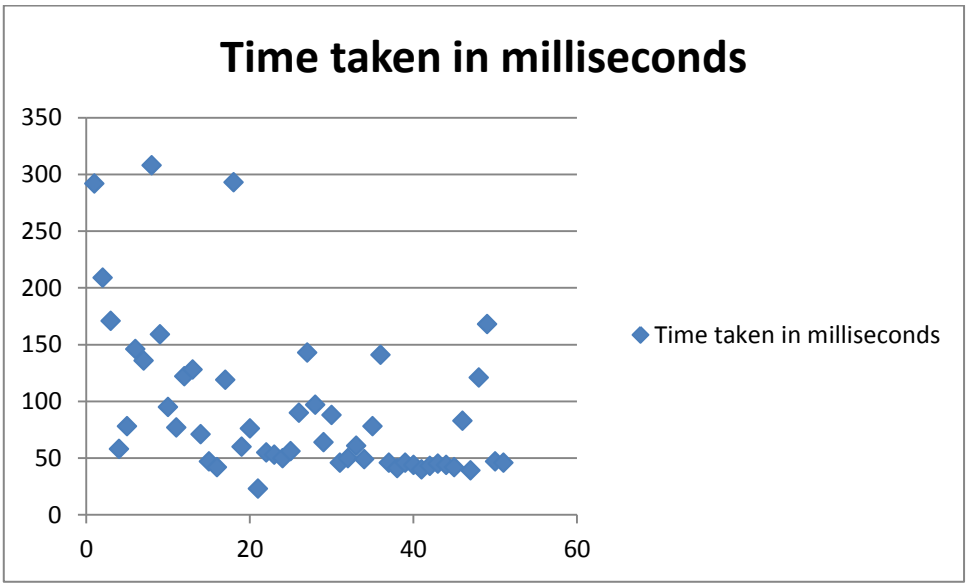

Figure 15: Time taken to Query services in server side with SOLR

5.7 Summary

As contribution of this thesis P2P distributed file sharing system to Mobile Host for Android was developed using Libtorrent library. Developed system has capability of handling torrent downloading, uploading and creating. For making services search more convenient SOLR was set up and used. Performance of torrent creating and SOLR searching was measured and analyzed.

6. Conclusions

The goal of this thesis was to design and develop a Peer-to-Peer distributed file sharing system on top on Mobile Host. Since BitTorrent is one of the most common protocols for transferring files in P2P networks, BitTorrent library was used.

Libtorrent, implementation of BitTorrent protocol was selected for making the client because it is open source and kept up to date with BitTorrent extensions. As Libtorrent is written in C++ and has many unnecessary extensions then not all possible extensions were used in the client as for that one class was necessary so java code could reach C++ library functions.

JXTA – open source P2P protocol was also studied but not used for making distributed file sharing system because it does not support that functionality and is currently not maintained since release 2.5. Contrary to that, BitTorrent protocol C++ implementation - Libtorrent is being kept up to date, is widely used and it's well documented and is easy to use.

Apache SOLR 4.2.0 was used for making system more user friendly providing full text search through available services to filter out distributed file sharing torrent service. Apache SOLR was chosen because it has powerful search engine with convenient API and it is easy to install and use.

7. Future work directions

This work has developed Peer-to-Peer distributed file sharing system on top of Mobile Host and sample search engine was used to make user experience better. As part of the future research directions, we are interested in updating searching to cloud based platform such as Amazon CloudSearch (still in Beta version).

Although developed BitTorrent client has all necessary functions for distributed file sharing, there could be more as Libtorrent has many extensions that were considered as not important for this thesis. Private torrents feature could be added along with SSL torrents and IP filter for users to have more choice of how to share files.

In terms of user experience, layout and design of Mobile Host and BitTorrent client could be improved. Currently everything is manageable, understandable and everything works but from simple users perspective the application is not very appealing as not much time has been invested in design of graphical user interface.

Partnervõrgul baseeruva hajusa failijagamissüsteemi loomine kasutades *Mobile Host'i*

Bakalaureuse töö

Pätris Halapuu

Resümee

Viimase kümne aasta jooksul on mobiilsed seadmed nagu näiteks nutitelefonid, sülearvutid, pihuarvutid jne saanud lahutamatuks osaks igapäeva elust. See aga on tekitanud nõudluse võimsamate, kiiremate ja energiasäästlikumate seadmete järgi. Lisaks on iOSi ja Androidi operatsioonisüsteemide väljalaske tõttu suurenenud nii mobiilirakenduste arv kui ka keerukus.

Sarnane areng on toimunud ka veebiteenuste valdkonnas ja nutitelefonides on ligipääs veebiteenustele muutunud elementaarseks. See aga on viinud järgmise sammuni – veebiteenuste pakkumine otse nutitelefonidest. See kontseptsioon pole uus ja seda on põhjalikult uurinud S. N. Srirama, kes pakkus välja Mobile Host (Mobiilne Veebiteenuse Pakkuja) lahenduse 2006. aastal, ning mida on C. Paniagua uuendanud Android OS'ile aastal 2012 kasutades REST arhitektuuri ja OSGi't.

P2P (Peer-to-Peer ehk partnervõrk) põhinevad programmid nagu näiteks failide jagamine ja sõnumite saatmine on tänapäeval arvutikasutajate seas laialdaselt levinud. Arvutid üle maailma on ühendatud omavahel ja jagavad ressursse selles süsteemis ilma keskse serverita. Iga arvuti selles võrgus on võrdne sel moel, et on võimeline ligi pääsema ja alla laadima ressursse teistest masinatest selles süsteemis.

Töö kirjeldab põgusalt Mobile Hosti, P2P arhitektuuri, valitud P2P BitTorrenti protokolli ja kuidas neid tehnoloogiaid kasutati P2P hajusa failide jagamise süsteemi loomiseks.

Uurimuse käigus arendati välja hajus failide jagamise süsteem Mobile Hosti lisana kasutades BitTorrenti protokollil põhinevat C++ keeles kirjutatud Libtorrenti teeki. Valminud programm on võimeline looma torrent faili, avama ja laadima ning jagama vastavaid faile. Lisaks on võimalik muuta failide hoiustamise asukohta ja määrata mõningaid sessiooni seadeid nagu näiteks alla ja üles laadimise kiirust piirata ning porti määrata. Failide jagamise teenust pakkuvate seadmete otsingu lihtsustamiseks seati üles Apache SOLR 4.2.0 veebiteenus mcrlabs.net serverisse millega suhtlus käib automaatselt.

References

1. Zero Configuration Network project site [online] URL: http://www.zeroconf.org/ Accessed 8 May 2013
2. International telecommunications union report 2011 [online] URL: www.itu.int/ITU-D/ict/facts/2011/material/ICTFactsFigures2011.pdf Accessed 8 May 2013
3. S. N. Srirama, M. Jarke, W. Prinz, Mobile web service provisioning, in: AICT-ICIW '06: Advanced Int. Conf. on Telecommunications and Int. Conf. on Internet and Web Applications and Services, IEEE Computer Society, 2006, p. 120. 1, 2, 10
4. Carlos Paniagua - Discovery and Push Notification Mechanisms for Mobile Cloud Services Master Thesis Tartu 2012
5. Yang, X., Bouguettaya, A., Medjahed, B., Long, H., & He, W. (2003). Organizing and accessing web services on air. Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on, 33(6), 742-757.
6. From P2P and Grids to Services on the Web: Evolving Distributed Communities Taylor, I.J. Harrison, A.B. Computer communications and networks URL: http://books.google.ee/books?id=DVKauOs36qUC 2009 Springer London
7. Rüdiger Schollmeier, A Definition of Peer-to-Peer Networking for the Classification of Peer-to-Peer Architectures and Applications, Proceedings of the First International Conference on Peer-to-Peer Computing, IEEE (2002)
8. Peer-to-peer distributed storage US 8051205 B2 [online] URL: http://www.google.com/patents/US8051205 Accessed 8 May 2013
9. Anitha, A., J. Jaya Kumari, and G. V. Mini. "A survey of P2P overlays in various networks." Signal Processing, Communication, Computing and Networking Technologies (ICSCCN), 2011 International Conference on. IEEE, 2011.
10. Hypersupreme.com article, torrent explained [online] URL: http://hypersupreme.com/torrent.php Accessed 8 May 2013
11. BitTorrentconcept blog about how it works [online] URL: http://bittorrentconcept.blogspot.com/ Accessed 8 May 2013
12. BitTorrent protocol specification [online] URL: http://www.bittorrent.org/beps/bep_0003.html Accessed 5 May 2013 Accessed 8 May 2013
13. Urvoy-Keller (December 2006). "Rarest First and Choke Algorithms Are Enough" SIGCOMM.

14. San Jose State University computer engineering department report on BitTorrent protocol [online] URL: http://www.openloop.com/education/classes/sjsu_engr/engr_ms_network/fall2006/preso/bt/BitTorrent.doc fall 2006 Accessed 8 May 2013
15. Extratorrent.com article about torrents trackers [online] URL: http://extratorrent.com/article/23/torrent+trackers.html Accessed 29 April 2013
16. Libtorrent project [online] URL: http://www.rasterbar.com/products/libtorrent/ Accessed 8 May 2013
17. Boost.Asio home page [online] URL: http://www.boost.org/doc/libs/1_43_0/doc/html/boost_asio.html Accessed 5 May 2013 Accessed 8 May 2013
18. D. Pratistha, N. Nicoloudis, and S. Cuce, "A micro-services framework on mobile devices," in International Conference on Web Services, Nevada, USA, 2003.
19. S. Srirama, "Concept, implementation and performance testing of a mobile web service provider for smart phones," Master's thesis, RWTH Aachen, Germany, 2004.
20. Y. Kim and K. Lee, "A lightweight framework for mobile web services," Computer Science-Research and Development, vol. 24, no. 4, pp. 199–209, 2009.
21. M. Asif and S. Majumdar, "Partitioning frameworks for mobile web services provisioning," International Journal of Parallel, Emergent and Distributed Systems, vol. 26, no. 6, pp. 519–544, 2011.
22. Sun Microsystems Inc., JXTA Java Standard Edition v2.5 Programmers Guide, United States of America: Sun Microsystems; 2007.
23. James Filbert – Developing a Multi-Purpose Chat Application for Mobile Distributed Systems on Android Platform Bachelor's Thesis Helsinki 2010
24. Project PeerDroid [online] 25 October 2009 URL: http://code.google.com/p/peerdroid/ Accessed 8 May 2013.
25. Srirama, S. N., Jarke, M., Zhu, H., & Prinz, W. (2008, June). Scalable mobile web service discovery in peer to peer networks. In Internet and Web Applications and Services, 2008. ICIW'08. Third International Conference on (pp. 668-674). IEEE.
26. Dyn DNS services provider [online] URL: http://dyn.com/support/bonjour-and-dns-discovery/ Accessed 8 May 2013
27. DNS service discovery [online] URL: http://www.dns-sd.org/ Accessed 8 May 2013

28. Apache SOLR home page [online] URL: http://lucene.apache.org/solr/ Accessed 8 May 2013
29. X. Yang, A. Bouguettaya, B. Medjahed, H. Long, and W. He, “Organizing and accessing web services on air,” IEEE transactions on systems, man, and cybernetics - part a: systems and humans, vol. 33, no. 6, pp. 742–757, November 2003.
30. S. N. Srirama, C. Paniagua: Mobile Web Service Provisioning and Discovery in Android Days, The 2013 IEEE International Conference on Mobile Services (MS 2013), June 27 - July 02, 2013. IEEE. (Accepted for publication)
31. S. Srirama, “Publishing and discovery of mobile web services in peer to peer networks,” in Proceedings of First International Workshop on Mobile Services and Personalized Environments (MSPE’06), vol. P-102. Lecture Notes in Informatics, GI, November 2006, pp. 15–28.
32. L. A. Steller, S. Krishnaswamy, and M. M. Gaber, “Cost efficient, adaptive reasoning strategies for pervasive service discovery,” in Int. conf. on Pervasive services (ICPS ’09). ACM, 2009, pp. 11–20.
33. Sip2peer project home page [online] URL: http://code.google.com/p/sip2peer/ Accessed 8 May 2013
34. Yaircc project home page [online] URL: http://yaircc.sourceforge.net/project-info.html Accessed 8 May 2013
35. Torrent4j project home page [online] URL: http://yaircc.sourceforge.net/project-info.html Accessed 8 May 2013
36. Bitext project home page [online] URL: http://bitext.sourceforge.net/ Accessed 8 May 2013

Non-exclusive licence to reproduce thesis and make thesis public

I, Pätris Halapuu

(date of birth: 31. July 1991),

herewith grant the University of Tartu a free permit (non-exclusive licence) to:

1.1. reproduce, for the purpose of preservation and making available to the public, including for addition to the DSpace digital archives until expiry of the term of validity of the copyright, and

1.2. make available to the public via the web environment of the University of Tartu, including via the DSpace digital archives until expiry of the term of validity of the copyright,

Establishing Peer-To-Peer Distributed File Sharing System With Mobile Host,

supervised by Satish Narayana Srirama and Carlos Paniagua ,

2. I am aware of the fact that the author retains these rights.

3. I certify that granting the non-exclusive licence does not infringe the intellectual property rights or rights arising from the Personal Data Protection Act.

Tartu, 10.05.2013