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

Hannes Metssalu

Demonstrating Android P2P capabilities through a prototype application

Bachelor's Thesis (9 ECTS)

Supervisor: Artjom Lind, MSc

Tartu 2015

Demonstrating Android P2P capabilities through a prototype application

Abstract:
Nowadays more and more people are communicating using electronic devices. This means that all kinds of data are transferred between devices. These are often private data but sent in a very public manner. When using traditional client-server approach, data may be seen or even altered by the server which means that the authenticity and privacy of data is always under question when using untrusted servers. Furthermore many people prefer to use mobile devices (tablet or mobile phone) instead of a PC for communicating and changing data yet there still does not exist a simple way to do it with certain privacy.

This thesis analyzes different methods for sending and receiving data between clients in a P2P (peer-to-peer) way instead of using traditional client-server model. Also a proof-of-concept application is written for Android which demonstrates how to easily enable P2P communication between multiple devices. Application will support sending messages between peers and also includes an example Hangman game for demonstrating game programming with P2P communication.

Keywords: P2P, Android, mobile device, game, Sip2Peer

Androidi P2P võimaluste demonstreerimine läbi prototüüprakenduse

Lühikokkuvõte:
Tänapäeval suheldakse aina rohkem elektroonilisi seadmeid kasutades. See tähendab, et seadmed vahetavad palju andmeid. Tihti on need andmed isiklikud, kuid saatmine toimub väga avalikul viisil. Kasutades levinud klient-server lähenemisviisi, võib server andmeid näha või isegi muuta, mis tähendab, et andmete autentsus ja privaatsus on rikutud, juhul kui kasutatakse ebausaldusväärset serverit. Lisaks eelistavad paljud inimesed suhtlemiseks ja andmevahetuseks mobiilseid seadmeid (tahvelarvutit või telefoni) tavalisele arvutile, kuid ikka veel ei eksisteeri lihtsat ning turvalist viisi selle tegemiseks.

See töö analüüsib erinevaid andmevahetusmeetodeid P2P (peer-to-peer) viisil, mis erineb traditsioonilisest klient-server andmevahetusmudelist. Lisaks luuakse näiterakendus Androidile, mis demonstreerib, kuidas lihtsal moel luua P2P ühendus mitmete seadmete vahel. Rakendus toetab sõnumite saatmist klientide vahel ning sisaldab Hangmani mängu, mis demonstreerib mängude programmeerimist P2P suhtluse abil.

Võtmesõnad: P2P, Android, mobiilne seade, mäng, Sip2Peer

Contents

1 Introduction

1.1 Problem overview

Most of the communication between electronic devices involves passing data through servers. This means that we have to rely on central servers sending our data around the globe without us knowing who reads or changes it. P2P communication often has its advantages over client-server model, yet there are no reliable and simple ways to achieve that. This thesis tries to analyze differences between client-server and P2P model and also bring out some of the current P2P applications both for PCs and mobile devices (for mainly Android OS). Creating P2P connection poses many difficulties due to firewalls. These will be looked upon and a solution will be proposed. Also a prototype application will be written, which overcomes these issues and demonstrates different ways to use P2P communication in Android applications.

1.2 Motivation

Currently the use of P2P in mobile devices is very limited. Nowadays P2P is often associated with piracy due to its most extensive use but this should not be the case. P2P communication could be beneficially used in almost every application. Also, creating P2P connection between two peers poses multiple issues due to firewalls and Network Address Translators (NAT), which will be discussed later. For mobile devices even more problems arise. Demonstrating overcoming issues will help to popularize developing P2P applications for mobile devices.

1.3 Goal

The goal of this thesis is as follows:

- Analyze advantages and disadvantages of P2P over client-server communication.
- Analyze the current state of P2P frameworks and applications for mobile devices.
- List out problems of creating P2P connections between mobile devices.
- Offer a solution to creating a successful connection between two mobile devices.
- Create a prototype application which can successfully create P2P between mobile devices.
- Motivate developers to write P2P applications for mobile devices.

1.4 Outline

Chapter 2 Peer-to-peer communication will be compared to traditional client-server model. Current P2P applications and frameworks will be analyzed and summarized.

Chapter 3 Specification for the prototype application will be given. Problems involving P2P connections will be brought out and explained. A mobile framework for solving these problems will be proposed.

Chapter 4 Detailed overview and a manual for developed application will be given.

Chapter 5 Created program will be compared with other similar programs.

Chapter 6 Thesis will be summarized and most important points will be brought out.

2 State of the art

This chapter discusses the main uses on P2P and also gives a detailed overview of what peer-to-peer communication means.

2.1 Peer-to-Peer

Peer-to-peer computing can be defined as the sharing of computer resources and services by direct exchange. P2P network consists of distributed clients or peers. Peers are all equally privileged participants in the network. Each peer in the network is often called a node. In such network peers can communicate directly with each other. In case of more common client-server model, all the data is sent and requested through centralized server.

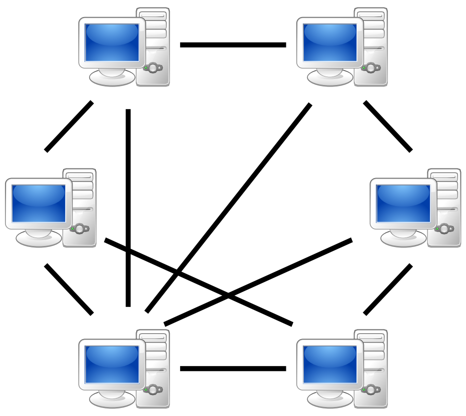


Figure 1: P2P

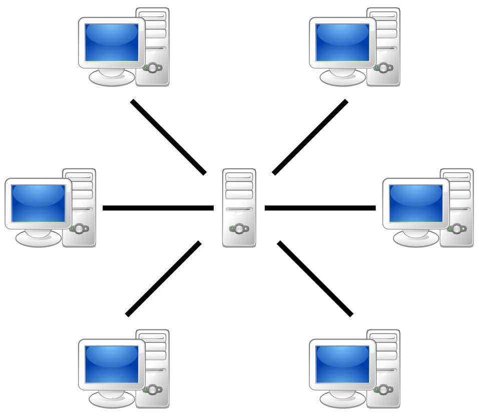


Figure 2: Client-server model

P2P networks have been previously used in many application domains, but the use of it started growing by the creation of different file sharing systems, the most popular being Napster in 1999 [Bar02]. In year 2008, P2P accounted nearly a third of traffic in North America. This was due to the fact that peer-to-peer file sharing programs were used extensively for piracy. By now, this number has declined to about 8% and is still declining [Fie14]. Yet, P2P still has its advantages over client-server model and has many potential use cases.

2.1.1 Advantages

Resilience

Since P2P has no central peer, it is not possible to collapse the network without taking down each individual peer.

Low cost

No special software or hardware is required to set up a P2P network.

Easy setup

In a small network, every major operating system has its own feature to set up data exchange easily.

2.1.2 Disadvantages

Scalability issues

As the number of users grow, the data is very clustered and accessing it by many people simultaneously causes the peers to slow down drastically.

No administrators

Due to nature of P2P networks there is nobody to monitor which data is being sent and for which purposes is it used.

Security problems

There is no way to stop any kind of traffic between nodes which enables viruses to spread easily without fear of detection by anti-virus software or network administrator.

Data loss

Information is distributed over all peers which means that in the case of peer failure, this part of network is also inaccessible.

2.2 Related works

This section contains some popular applications which rely on P2P communication for both PC and mobile devices. Skype which used to have its own closed source P2P protocol, is also worth mentioning, but as of June 20, 2014, Skype announced this protocol to be deprecated and started using Microsoft Notification Protocol 24 which does not involve the use of P2P networking. [Pet14]

2.2.1 Freenet

Freenet was originally created in 2000 by Ian Clarke as a student project at the University of Edinburgh [Mar00]. It is written in Java, works on all major operating systems and is still actively being developed by The Freenet Project Team. Its goal was to create a peer-to-peer based platform for communication without restrictions or censorship. Freenet also claims to be used strongly anonymously. By now it has grown into a huge network of nodes all over the world. Freenet enables peers to access data and webpages which are uploaded into Freenet network. Storage is distributed between peers which means that every user acts as a small storage space for the whole system. Every bit of information is replicated between peers so that no user could not be responsible for any piece of information. More information [1]

2.2.2 LogMeIn Hamachi

Hamachi is an application that is capable of establishing direct connections between computers with whom it would be otherwise impossible due to firewall and router limitations, namely NAT (which will be discussed later). It is capable of emulating local area network

[1] The Freenet Project [confirmed on 14.05.2015] `https://freenetproject.org/index.html`

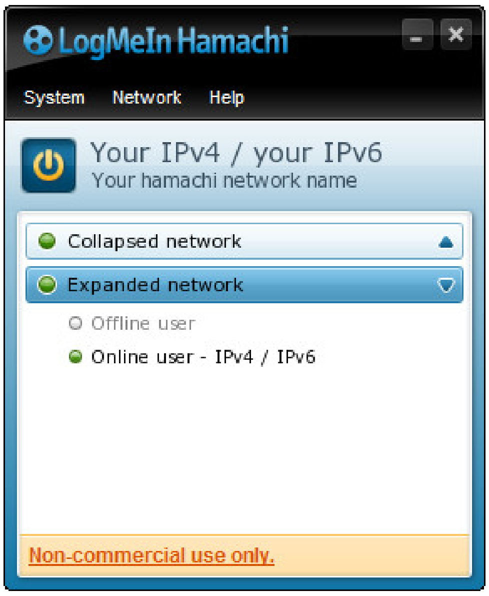


Figure 3: LogMeIn Hamachi

virtually with devices which are not in local network. The exact means of how Hamachi bypasses NAT and firewall for creating direct tunnels are not made public. LAN connections are very useful for remote administration or computer gaming. All major operating systems are supported by LogMeIn Hamachi. More information at Hamachi webpage. [2]

2.2.3 BitTorrent Sync

BitTorrent Sync is a peer-to-peer file synchronization tool for Windows, Mac, Linux, Android, iOS, Windows Phone, Amazon Kindle and BSD, still actively developed by BitTorrent, Inc. It was first released in the beginning of 2013 by Konstantin Lissounov. BT Sync can synchronize files between devices on a local network or between remote devices over the Internet. For creating connections with peers it uses a modified version of BitTorrent protocol [3]. BT Sync has no limitations on speed or size of data synchronized other than the size of peers' storage space and the speed of the connection. There have been claims that BitTorrent Sync is not secure enough for private data but these claims were looked at and given a response quickly by BitTorrent team which showed that there were no major security issues [4].

2.2.4 P2P developement frameworks/libraries

Android Wi-Fi Peer-to-peer (Wi-Fi Direct) Allows Android 4.0 or later devices to connect directly to each other via Wi-Fi without using an intermediate access point. It has very simple methods to create a local network of peers over Wi-Fi yet it lacks the possibility to use mobile data and create connections with remote peers. Wi-Fi

[2]LogMeIn Hamachi [confirmed on 14.05.2015] https://secure.logmein.com/products/hamachi/
[3]BT Sync technology [confirmed on 14.05.2015] http://www.getsync.com/how-it-works
[4]BT reply to Hackito [confirmed on 14.05.2015]
http://www.networkworld.com/article/2849452/microsoft-subnet/bittorrent-reply-to-hackito-report-on-bittorrent-syncs-bad-crypto-no-cause-for-concern.html

P2P Complies with Wi-Fi Direct^{TM} certification program. [5]. The documentation is very broad and the library is suitable for creating applications which require creating a small peer-to-peer network in a local area.

P2P-communication-framework-for-android An easy to implement peer-to-peer framework allowing a nearbly cluster of Android devices to easily communicate with each other. Uses bluetooth for creating a network cluster. Open-source, but not very well documented. Nowadays bluetooth has much alternatives with better range and stability, which makes it outdated for most modern applications but still has its specific use cases. More information [6].

OpenPeer An open P2P signalling protocol with their main objectives being open-source, secure, private and scalable. OpenPeer has beta SDKs for both Android and iOS developers. Its core-library is written in C++. OpenPeer has its own central server for distributing information and creating connections with other peers. OpenPeer is suitable for developing mobile cross-platform applications which require direct peer-to-peer communication over the Internet. More information [7].

Hive2Hive An open-source library, written in Java, for secure, distributed, P2P-based file synchronization and sharing. It has a very detailed documentation and simple API. More information: [8].

Sip2Peer An open-source SIP-based middleware for the implementation of any peer-to-peer application without constraints on peer device and architecture. This library will be used to create a prototype application for Android and will be looked upon in detail later. [9].

[5]Wi-Fi Direct protocol [confirmed on 14.05.2015] `http://blog.broadcom.com/wp-content/uploads/2013/10/Wi-Fi-Direct-White-Paper.pdf`

[6]p2p-communication-framework-for-android [confirmed on 14.05.2015] `https://code.google.com/p/p2p-communication-framework-for-android/`

[7]OpenPeer specification [confirmed on 14.05.2015] `http://docs.openpeer.org/OpenPeerProtocolSpecification/`

[8]Hive2Hive repository [confirmed on 14.05.2015] `https://github.com/Hive2Hive/Hive2Hive`

[9]Sip2Peer [confirmed on 14.05.2015] `https://code.google.com/p/sip2peer/`

3 Problem statement

Creating an application which has to communicate via the Internet with other devices has always been a dubious task. This is even more so when using mobile devices and P2P due to restrictions that establishing direct connections have. Since generally client-server model is more suitable for mobile applications, there is no proper documentation or examples on how to develop a successful mobile application that is capable of both creating direct connections between each other and sending/receiving data without issues. This thesis tries to give out an example application which would solve the main difficulties of P2P and also demonstrate some use cases of a P2P mobile application. The application will be able to send/receive messages and include a proof-of-concept level Hangman game. Sip2Peer Java library will be used which will help in successfully creating P2P connections and managing peers.

3.1 Difficulties of P2P

The biggest problem of creating a successful peer-to-peer connection is NAT and firewalls which both block incoming packets unless configured properly. There are ways to get through NAT such as NAT Traversal. Theoretically NAT is only a problem for IPv4 standard on which the majority of Internet is currently based. There is currently undergoing a switch to much newer standard IPv6 which takes a lot time but will eventually solve the NAT problem for P2P connections [Dee98]. Also for mobile devices a big problem is that mobile connection changes the device's IP address very often which poses an issue when trying to maintain connection between both peers.

3.1.1 Network Address Translator

Network Address Translators (NAT) were first introduced to stop Internet from running out of IPv4 (Internet Protocol version 4) IP addresses. IPv4 uses 32-bit address which limits the space to $2^{32} \approx 4$ billion different addresses (the practical number is really much smaller). NAT is located between the public Internet and the network it serves. Its primary function is to translate public IP addresses and port numbers into specific local IP addresses of local machines which are hidden from the public Internet. After a machine has sent packets to outgoing server, NAT knows that incoming packets from the same server have to be routed back to the same machine. In all cases, we must assume that an application will send and receive packets on the same port. In such case the incoming data gets sent through NAT to correct machine but in the case of P2P and an unconfigured NAT router, NAT does not have any information on where to send packets and they get destroyed [SFK08].

3.1.2 NAT Traversal

NAT Traversal (NAT-T) is a general term for different techniques that establish and maintain Internet connection between local and remote client through NAT. These techniques are typically required in peer-to-peer network applications. The majority of NAT-T techniques do not work for all types of NATs. These methods include:

- Socket Secure (SOCKS)
- Universal Plug'n'play Internet Gateway Device (UPnP IGD)

- Interactive Connectivity Establishment (ICE)
- Application-level gateway (ALG)
- Hole punching
- Session Border Controllers

3.2 S2P library

Sip-2-peer (S2P) is a Java library for implementing peer-to-peer communication in any application without any constraints on peer device nor architecture. It has tutorials for both PC and Android devices. S2P has its own NAT traversal management through SIP protocol and is perfectly suitable for this thesis' prototype application.
Session Initiation Protocol (SIP) is a communications protocol for signaling and controlling multimedia communication sessions (Internet telephony for voice and video calls) but can also be used in assisting NAT Traversal successfully. Session Border Controller (SBC) is a part of SIP network elements, which handles NAT Traversal. [BS] In S2P case, SBC is a node with public IP that allows a peer to check if it is behind a NAT and to request a public IP and port that can be used by the requesting node as a contact address and can be advertised to other peers. Like all the other methods for NAT Traversal, SBC and SIP do not work for every NAT. Most of the mentioned NAT Traversal methods do not work for Carrier Grade NAT's (CGN) [10].

3.3 Application specification

The main purpose of the application is to demonstrate the usage of S2P library for Android for creating a successful peer-to-peer network. It will include a feature to send messages to other peers and a small Hangman game.
Ideally a working network needs:

1. Boostrap peer which saves data for every peer in network and can share this data with other peers.
2. SBC server with public IP address which helps connect peers that are behind NAT by assigning them a public IP address and advertising it to others.
3. Client peers which communicate with each other and can request other peers' data (IP address) from bootstrap peer. Every client peer can also be a bootstrap peer.

The main specifications:

- Uses S2P library
- Written in Java
- Supports Android devices with API (Application Programming Interface) version above 15

[10] Carrier Grade NAT [confirmed on 14.05.2015] `https://www.apnic.net/community/ipv6-program/about-cgn`

Main features:

- Directly connect to other peers using their IP
- Request network members' IP addresses from bootstrap peer
- Ping other peers to acquire their IP address
- Send chat messages to other peers
- Host a Hangman game
- Play Hangman game

One of the ideas which this game demonstrates is that only the host needs the game code in its application. Other players only receive information about the status of the game and send messages back to host without really having any knowledge or code how the game works. This reduces storage space and offers security for game developers and owners.
This Hangman game needs a host and players. Host will pick a word, for example "machine". A message "Lets play, the current status is _ _ _ _ _ _ _" will be send out to every peer in the network 4. After a client has sent for example "a" back to host 5, a new message will be sent out to every peer. This message includes already guessed letters and the current status, for example "Current status: _a_ _ _ _ _ ; Guessed letters: {a,b,c}" 6. After all the letters of the word are guessed the host will send out a message "Game is over, the winner is <winner's name>" 7.

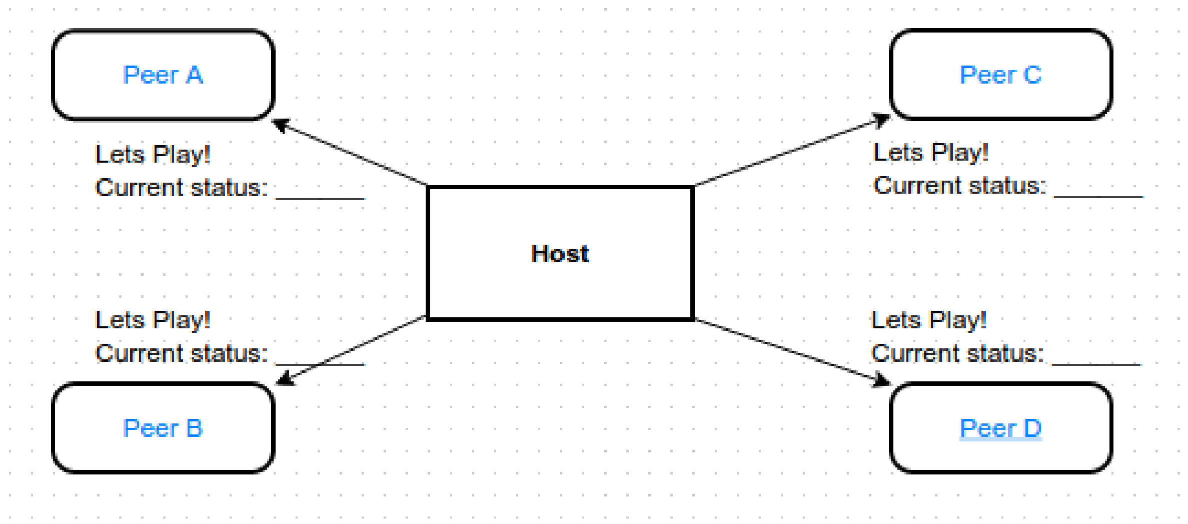


Figure 4: Welcome message by host

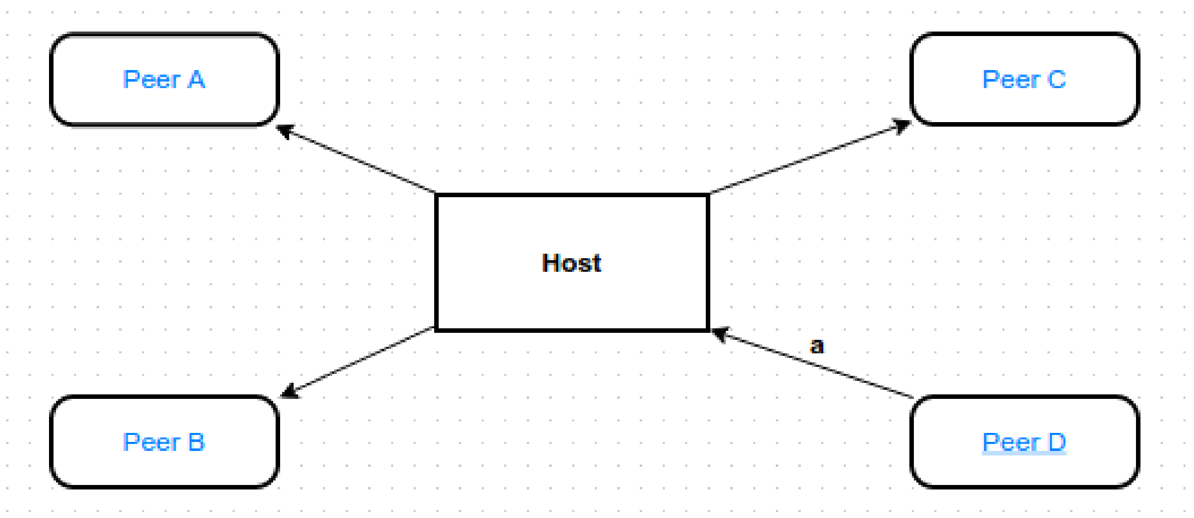


Figure 5: Guess by Peer D

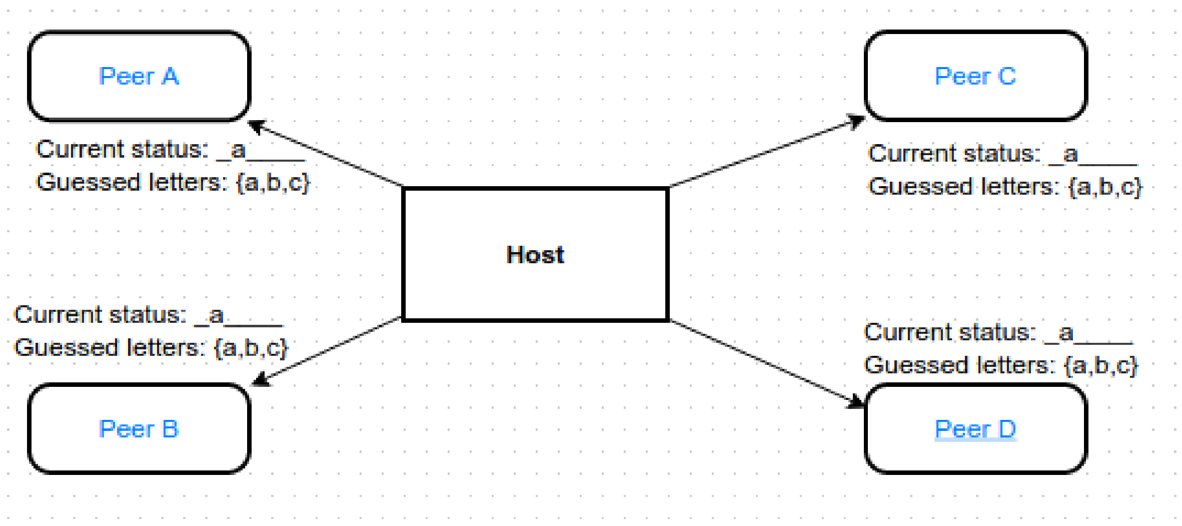


Figure 6: Status message by host

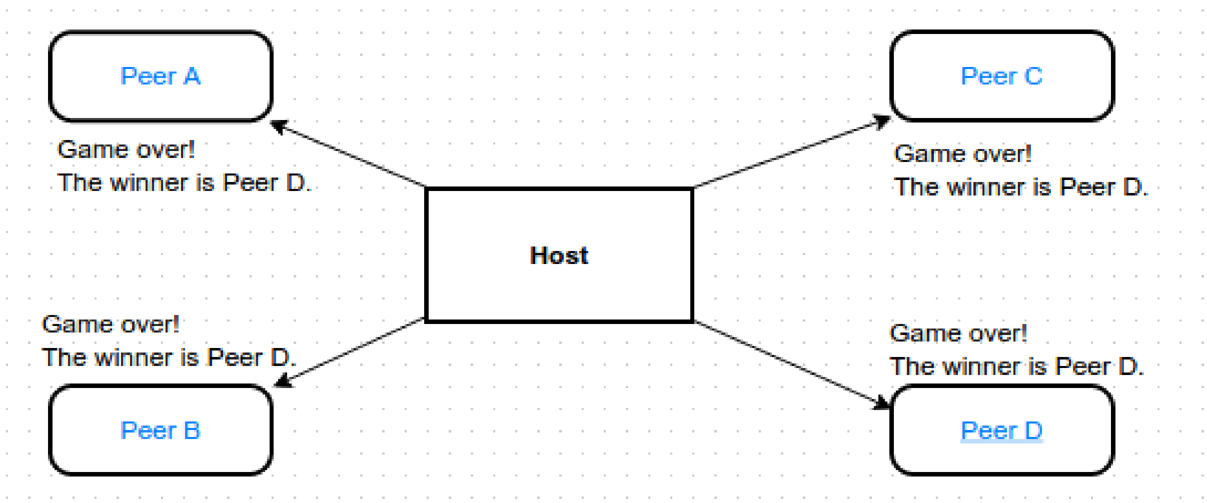


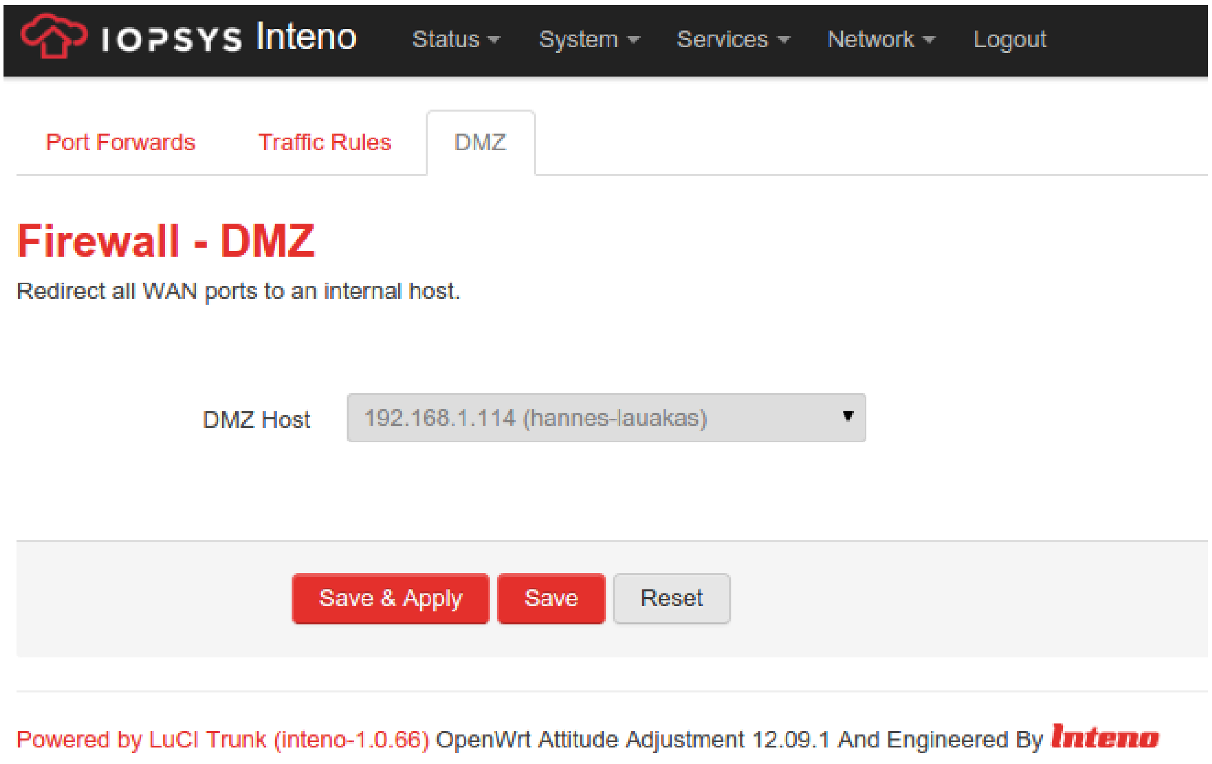
Figure 7: Game over message by host

4 Implementation

The application was written using Android Studio 1.2 under Linux environment. All the testing was done using Genymotion (an emulator for Android devices) and multiple Google Nexus 5 virtual images. This thesis is written in LaTeX using TeXstudio 2.8.4.

4.1 Demo application

S2P library offers multiple template projects both for Android and PC developing. SBC and Bootstrap peer were both set up on PC without changing any major code. Since SBC requires public connectivity with remote network, ports had to be opened and router configured. The first issue was properly configuring the router. The router used was Inteno DG301AL. The solution was to use DMZ (demilitarized zone) which forwards all the incoming data to a certain IP in a local network which in our case is the PC with SBC and bootstrap peer. After configuring the router and setting up the configuration files



Firewall - DMZ

Redirect all WAN ports to an internal host.

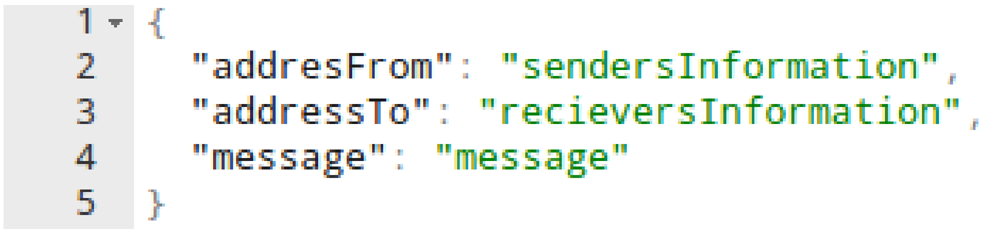
DMZ Host: 192.168.1.114 (hannes-lauakas)

Powered by LuCI Trunk (inteno-1.0.66) OpenWrt Attitude Adjustment 12.09.1 And Engineered By Inteno

Figure 8: Router DMZ settings

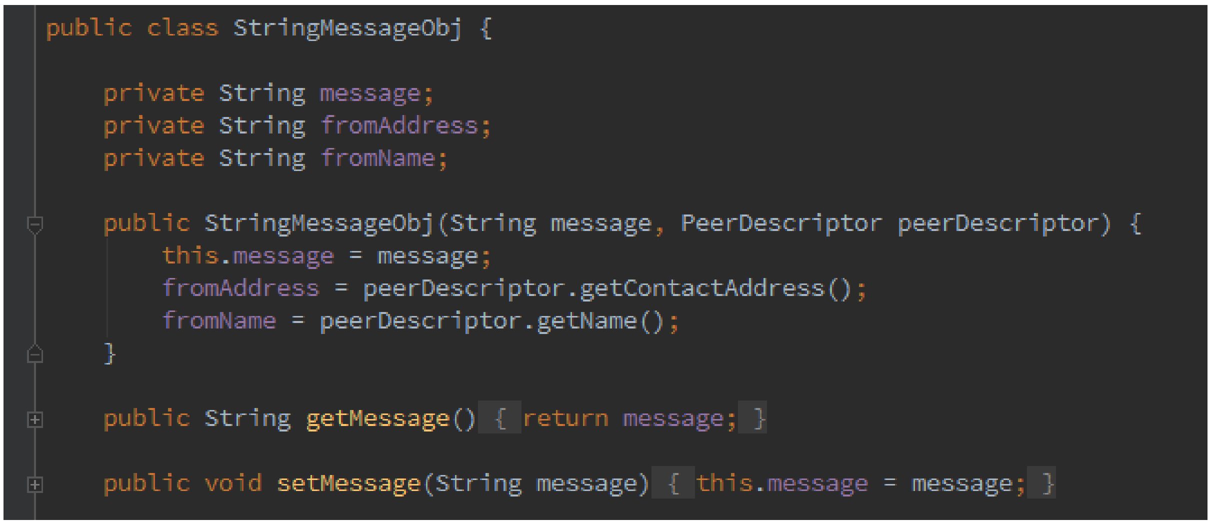
for SBC and bootstrap, Android application was written. Example Android application allowed to send pings (request IP address) between peers. Sending chat messages was implemented through JSON (JavaScript Object Notation) messages since it gives a good example on how to send more complex objects between devices.
As can be seen from Figures 9 and 10, JSON object corresponds exactly to Java object and is generated from object variable toString() methods. It is also worth mentioning that every field needs a get method for the JSON object to be generated properly.

Most of the work was done on writing the Hangman game. All the new code was added in such a way, that it is only required in game host's application. The game communicates with other peers using already created message sending methods. While



```
{
  "addresFrom": "sendersInformation",
  "addressTo": "recieversInformation",
  "message": "message"
}
```

Figure 9: JSON structure



```
public class StringMessageObj {

    private String message;
    private String fromAddress;
    private String fromName;

    public StringMessageObj(String message, PeerDescriptor peerDescriptor) {
        this.message = message;
        fromAddress = peerDescriptor.getContactAddress();
        fromName = peerDescriptor.getName();
    }

    public String getMessage() { return message; }

    public void setMessage(String message) { this.message = message; }
```

Figure 10: Java object structure

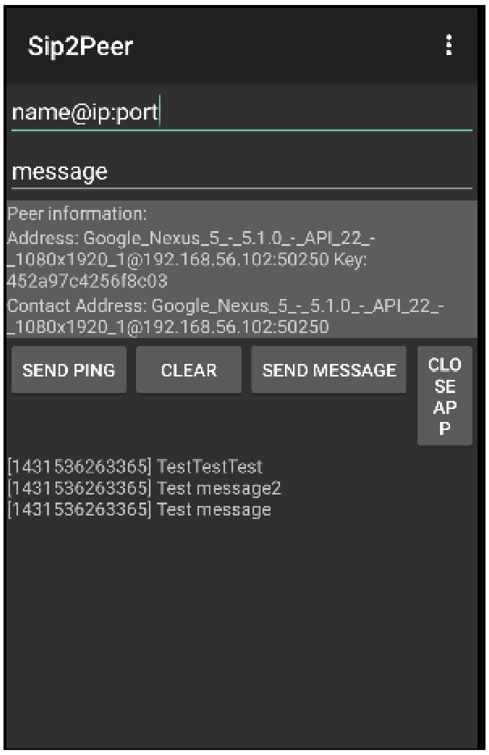
the host is hosting the game, a boolean flag is raised for him that the game has started. In such way, host knows to treat incoming messages as game messages instead of chat messages.

4.1.1 Application structure

Android application front-end consists of Activities. Each Activity can be imagined as one screen on device. Prototype application's Activities are as follows:

1. Main Activity which contains sending messages to other peers and a chat log11.
2. Configuration activity where SBC and bootstrap peer information is contained12.
3. Bootstrap Activity where peer can connect to bootstrap peer (data is automatically pulled from configuration menu)13.
4. Peer list Activity which contains other network members' name and IP addresses13.
5. Host game Activity from which Hangman game can be hosted15.

Navigation from one Activity to another can very well be illustrated with the following diagram 16.



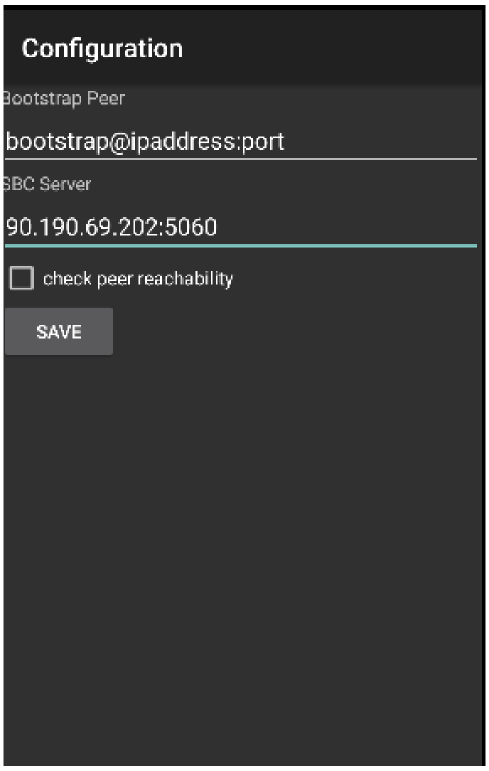
name@ip:port

message

Peer information:
Address: Google_Nexus_5_-_5.1.0_-_API_22_-_1080x1920_1@192.168.56.102:50250 Key: 452a97c4256f8c03
Contact Address: Google_Nexus_5_-_5.1.0_-_API_22_-_1080x1920_1@192.168.56.102:50250

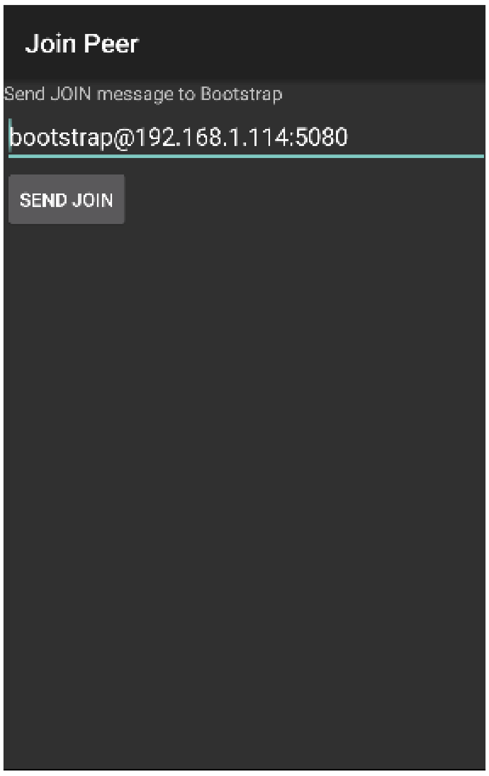
```
[1431536263365] TestTestTest
[1431536263365] Test message2
[1431536263365] Test message
```

Figure 11: Main activity



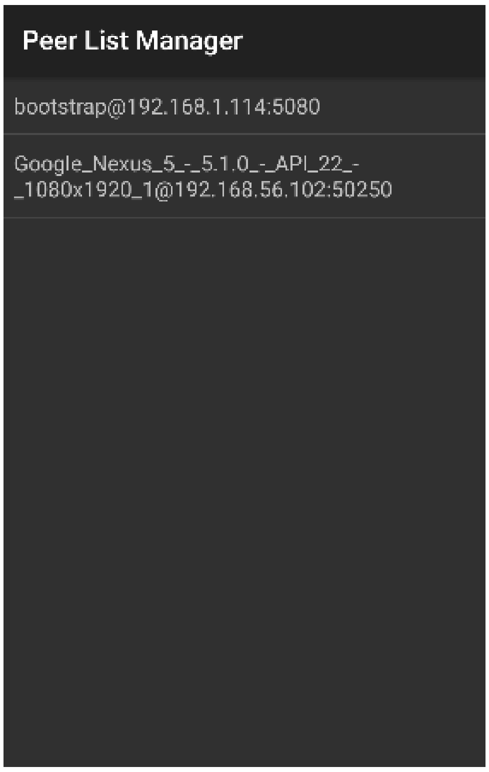
Bootstrap Peer
bootstrap@ipaddress:port
SBC Server
90.190.69.202:5060
☐ check peer reachability

Figure 12: Configuration Activity



Send JOIN message to Bootstrap
bootstrap@192.168.1.114:5080

Figure 13: Bootstrap Activity



bootstrap@192.168.1.114:5080
Google_Nexus_5_-_5.1.0_-_API_22_-_1080x1920_1@192.168.56.102:50250

Figure 14: Peer list Activity

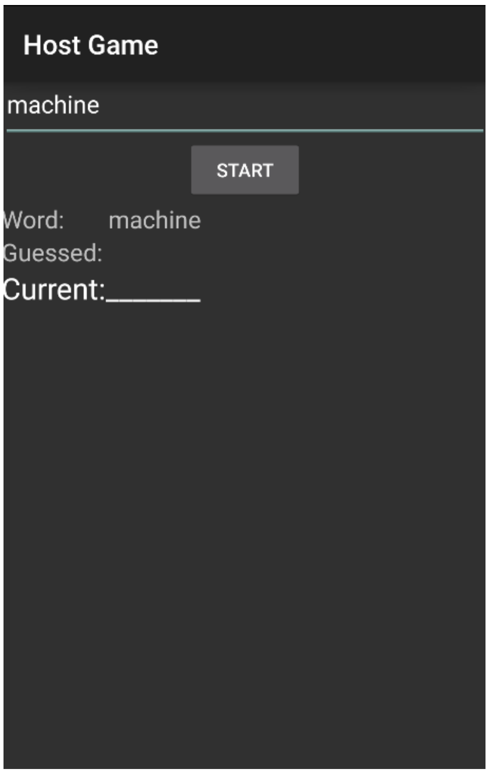


Figure 15: Host game Activity

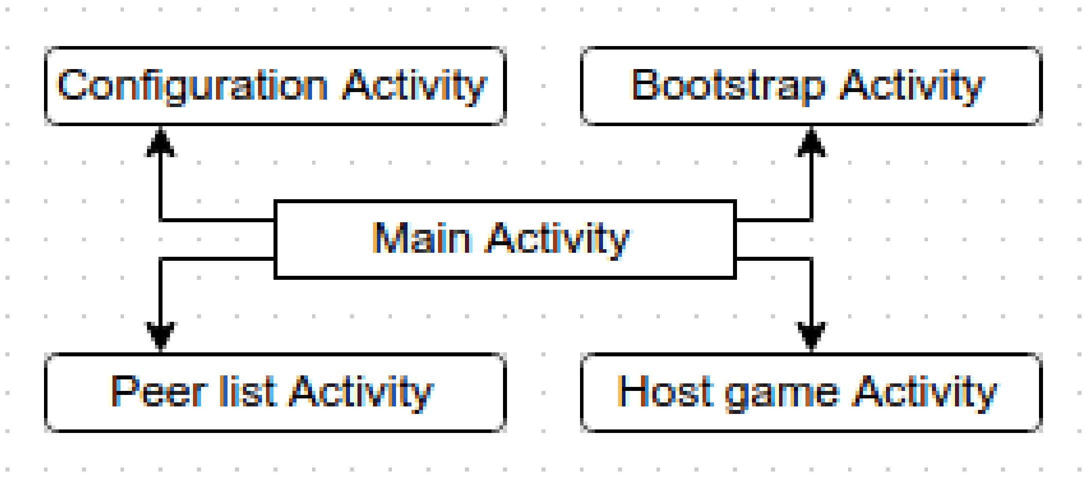


Figure 16: Navigation

4.2 Result analysis

The results of this work are divided into two parts both of which are equally informative and useful. Theoretical part will gives fairly up-to-date and summarized overview of current state of existing P2P applications and frameworks. Also the main issues with using P2P with mobile devices were brought out and possible solutions explained.
For the practical part everything also works as stated in application's specification. Since routers have different built in NATs, this application may not be successful in creating a connection with every peer. The application has very modular design and logical code structure, which means that other types of NAT Traversal techniques can be added without changing the majority of the code. All the main features of P2P data exchange were covered and an example given on how to exchange Java objects through JSON objects. Also a demonstrative Hangman game was written which shows how P2P connection could potentially be used for sharing games in a closed environment with other peers without them having the source code in their application.
Sip2Peer is not very popular library and therefore not very many projects use it. OghmaSip is built on Sip2Peer and is a very similar application for media streaming [Ege11]. It uses OpenId for identifying peers rather than Android device ID which is used in this thesis' application.

4.3 Running the system

Prerequisites:

- Android device(s)
- Connection with public IP address
- Java 1.8
- S2PBootstrap project
- Sip2PeerSBC project
- Android APK

Bootstrap peer S2PBootstrap should be opened with any development environment (e.g. IDEA, Eclipse, NetBeans etc.). Configuring the Bootstrap peer (should not be necessary) can be done from /config/bs.cfg file. Starting the peer is done from package it.unipr.ce.dsg.s2p.example.peer with class BootstrapPeer main method. After starting the method, an IP address will be printed out which can be used by other peers. The default port is 5080

SBC server Sip2PeerSBC should be opened in a similar manner. For configuring /config/sbc.cfg should be used. Starting the SBC server is done from it.unipr.ce.dsg.s2p.sip.sbc package from class SessionBorderController main method. SBC server IP address will be printed out.

Mobile application Application can be run by transferring the .apk file to mobile storage and installing it. It can be also run by opening the project in any of the Android supporting IDEs and running the application from there.

5 Conclusion

This thesis showed that although the use of P2P has declined in past years, there are still many use cases for peer-to-peer networks. Given the fast speed of modern connections, file synchronization using P2P seems to become more popular (BitTorrent Sync) because it virtually has no limits. NAT and NAT Traversal issues were also looked upon. Currently switching to IPv6 protocol is undergoing which means soon NAT should not be an issue anymore and this could mean a potentially huge spike in P2P usage. Sending direct data directly from peer to peer is also very secure and when using proper encryption is very hard to tamper with. Piracy has been a big reason why Internet service providers are limiting P2P traffic but different market models have been developed to reduce piracy. This work provides excellent means to start developing P2P applications for Android which is a huge market and needs more peer-to-peer applications. A modular design application was developed which can act as a template for basically every program which needs P2P communication.

5.1 Future Work

This application is currently in a prototype state and needs changes for real world use. Currently it provides information on how to do certain things.
One possible use case would be to develop a very abstract interface for receiving and displaying game data. This could be used for simple card games or text based games with a beautiful graphical user interface. Even a small program language for this can be developed where so that any application owner can write its own game and share play it with others without sharing the source code of the game.
Other use case would be a file sharing application where people can share different files from their devices between a network of peers. If multiple people have the same files, the speeds will become much faster. It can also be used for synchronizing for example tablet and mobile storage. The whole application could be updated to support IPv6 IP addresses to keep up with current Internet protocol standards.

References

[Bar02] David Barkai. An introduction to peer-to-peer computing. page 7, February 2002. [confirmed on 14.05.2015] `http://www2.it.lut.fi/wiki/lib/exe/fetch.php/courses/ct30a6900/p2p_barkai.pdf`.

[BS] David Schwartz Baruch Sterman, Ph.D. Nat traversal in sip. page 17. [confirmed on 14.05.2015] `http://startrinity.com/VoIP/Resources/sip26.pdf`.

[Dee98] Stephen E Deering. Internet protocol, version 6 (ipv6) specification. 1998.

[Ege11] Raimund Ege. Oghmasip: Peer-to-peer multimedia for mobile devices. In *MOBILITY 2011, The First International Conference on Mobile Services, Resources, and Users*, pages 1–6, 2011.

[Fie14] Seth Fiegerman. The slow decline of peer-to-peer file sharing. page 1, May 2014. [confirmed on 14.05.2015] `http://mashable.com/2014/05/14/file-sharing-decline/`.

[Mar00] John Markoff. Cyberspace programmers confront copyright laws. page 10, May 2000. [confirmed on 14.05.2015] `http://www.nytimes.com/2000/05/10/business/cyberspace-programmers-confront-copyright-laws.html`.

[Pet14] Noah Petherbridge. Skype switched to the msn messenger protocol. page 9, December 2014. [confirmed on 14.05.2015] `https://www.kirsle.net/blog/entry/skype-switched-to-the-msn-messenger-protocol`.

[SFK08] Pyda Srisuresh, Bryan Ford, and Dan Kegel. State of peer-to-peer (p2p) communication across network address translators (nats). *Internet Engineering Task ForceâĂŤRequest for Comments*, 5128:1–32, 2008.

6 Appendices

6.1 Appendix A: Prototype Application

The prototype application is located in GitHub repository
`https://github.com/hannesss81/PrototypeApplication`

6.2 Appendix B: S2PBootstrap project

The prototype application is located in GitHub repository
`https://github.com/hannesss81/S2PBootstrap`

6.3 Appendix C: Sip2PeerSBC project

The prototype application is located in GitHub repository
`https://github.com/hannesss81/Sip2PeerSBC`

6.4 Appendix C: License