# The implementation of peer-to-peer in flash crowds
## *A case study of bandwidth and cost reduction for over-the-air updates*

Artur Matulaniec
dat11ama@student.lu.se

Sony Mobile Communications AB

Advisors:
Magnus Thuresson, magnus.thuresson@sony.com
Jens A. Andersson, jens_a.andersson@eit.lth.se

June 21, 2017

Printed in Sweden
E-huset, Lund, 2017

# Abstract

This thesis will examine the plausibility of exchanging protocol from a normal server-client to a peer-to-peer solution for over-the-air updates, using a single case study for Sony Mobile Communications AB in Lund, Sweden. It will examine the effort, the risks, and what the actual gain may be comparing to a client-server approach for over-the-air software updates. Sony Mobile today uses a content delivery network which creates a peak upon new releases of software and this thesis will examine how efficient the peer-to-peer approach is for flash crowds in terms of saved data from the content delivery network. Continuing on the problem there are several global network infrastructure limitations, such as NAT44, NAT444 and firewalls, which proves to decrease the overall connectability of peers in peer-to-peer networks. Live experiments using real mobile devices from Sony Mobile will be used to examine the gain after a comprehensive literature study to explore what impact such network limitations has on peer-to-peer networks. As a consequence showing that even with limitations there is a potential gain of changing the approach to a peer-to-peer model. The live experiments prove that with few steps there can be a potential data saving gain up to 20%, without any interference with the global infrastructure. The findings will also prove the feasibility of additional hole punching techniques which may further push the gain towards higher numbers.

# Acknowledgements

This thesis marks an end to a road which began in autumn 2011. It has been a bumpy ride with both high valleys and deep rivers but with the support of family and friends it has been quite enjoyable. I want to thank you all for the support, cheering words, and the great feedback received along the way. Without you it would have been hard to progress.

A special acknowledge is pointed towards my supervisors of this thesis, Magnus and Jens. Without the support and enlightenment of certain topics I would have still wondered around in a dream world. Your comments and input along the way have been very valuable for the outcome of this thesis and I highly appreciate all of your support.

A special thank you to Lars Bergman for giving me the opportunity to conduct this thesis in his organization. It has been a pleasure to work with all the new colleagues I have had around me. A lot of laughter, a bit of work, and some stress can summarize the experience but I would not imagine having it any other way.

*All good things must come to an end* as the saying goes and I look forward on the new adventures that life will present itself with.

# Popular Science Summary

As from ancient history, sharing a burden have always ensured that a task can be completed much quicker than doing it entirely by yourself. Columbus would not have made it to the West Indies without the help of his crew, and the same applies for over-the-air software updates to actively reduce data flow from a server, and by doing so reducing costs by up to 40%!

Peer-to-peer, a solution which can be easily described with a pub environment, where all the pub-people speak out and everyone listens at the same time. Information is shared to all without a central entity in between. Over-the-air software updates are spread globally whenever a mobile receives a fresh version of the operating system. This is usually followed by a craving of having the latest by the end-user which results in a flash crowds. Flash crowds, as a phenomenon, resembles ants going after sugar. This flash crowd puts a normal server structure under a lot of strain. It is as well costly as all the downloads must be paid for by the manufacturer. By distributing the effort to the mobile devices themselves, the data flow reduces by 17%, which actively translates to a cost reduction of the same magnitude. This is just by changing the way of communication, from a monologue to a dialogue for the mobiles. However, there are some drawbacks. In the global Internet, public IP-addresses are being used in extent and to many devices are today connected. Consequently, now even IP-addresses are shared among many devices.

Returning to the pub scenario, imagine that you need to talk to a stranger which has your hat, because you throw it in the air when your favorite song came on. However, to reach this stranger you must firstly pass a bouncer, as the stranger is within a restricted area (private network). The bouncer acts as a network address translator (NAT) and tells you whether you may pass or not, if you know this stranger you can tell the bouncer this and he will let you through. If you do not, then you will be blocked and the bouncer will just throw away your request.

To deal with such bouncers on the Internet, hole punching techniques have been introduced. A hole punching technique is, with common predetermined rules in the pub and with the bouncer's attention, you passing the line and entering the restricted area to the stranger and successfully retrieving your hat. This mutual agreement between you (mobile device), the pub (the Internet), and the bouncer (NAT) represents the hole punching technique Universal Plug and Play, or just

UPnP. Consequently, with UPnP the 17% reduction can increase and reach levels towards 40%, which is the ideal scenario where all the pub-people can freely speak to anyone without restricted areas. Actively increasing the amount of friendly faces willing to share the software updates with you.

The plausibility of transitioning towards a peer-to-peer model from a normal client-server approach not a large effort for any mobile manufacturer. The conclusion is that even with obstacles present around the world, just as with the restricted areas in the pub, it is with little effort of implementation a cost reduction can be obtained when new versions are released online. Any manufacturer trying to reduce costs should look no further than towards peer-to-peer with additional hole punching techniques. Such a transition means very little effect on the mobile devices but actively reducing costs and releases strain on any server structure which might be in place for any manufacturer.

# Table of Contents

# List of Figures

# List of Tables

# Chapter 1
# Introduction

This thesis will introduce and examine the field of content delivery networks and peer-to-peer focusing on over-the-air updates on behalf of Sony Mobile Communications AB in Lund, Sweden. Sony Mobile and Sony will be used as abbreviations throughout the thesis. This chapter will describe the overall outline of the conduction of the thesis, methodology, and way of proceeding towards the primary objectives.

## 1.1 Primary objectives

The thesis will primarily focus on three main objectives. These objectives will be the main motivation as of choosing direction and topics for the thesis. They will be explored, answered, and tackled using primarily an extensive literature review and further on experiments using a live setup with mobile devices from Sony.

- *Will a peer-to-peer protocol, like BitTorrent, help improve the performance of OTA updates, when there is a need for a bigger network capacity during major software releases.*
- *Will this approach allow a manufacturer to reduce network bandwidth from the server to the devices effectively reducing costs for the company.*
- *What are the risks, the costs and the effort needed to change the current approach into a new one.*

## 1.2 Context and motivation

The motivation of the thesis is to propose and examine whether a peer-to-peer approach will reduce the bandwidth for a manufacturer's over-the-air updates when new versions of applications or software updates are being released for commercial use. Similarly to many other fields of study, information sharing is a continuous evolving field of study. Not only are more and more people globally being connected to the Internet but also more and more items are becoming part of the Internet. Consequently, all of the connected devices and people create a lot of data flowing around the globe. As such, increasingly more clients must be ensured a stable and fast connection. This thesis will examine how a connection towards

other clients might help reduce connection delays and create a more efficient ad-hoc network structure, as the commonly known fact of peer-to-peer networks present.

Information sharing has historically been built on a client-server approach. A server serves multiple client requests concurrently. Due to the recent exponential growth of the Internet, such an approach puts a lot of strain on maintaining stable servers. A normal client-server approach might lack the scalability if many more clients would requests information at the same time. As a result, content delivery networks have evolved. Content delivery networks still rely on the same client-server structure but with server nodes distributed around the globe. This global distribution creates shorter distances for clients when requesting data from a server and the request will be directed towards a closer node.

As of recent times, when mobile devices are given greater computational power, content delivery networks are being reworked with an overlying peer-to-peer network. As a result, they may help reduce the bandwidth of content delivery networks even further. Although this field of study is still very young this thesis will assist with covering the performance and possibly the financial gain such network topology may have. This thesis will also look into the performance benefits of such a solution. Peer-to-peer networks do have the benefits of scalability and creating ad-hoc networks expanding the range of networks. Consequently, countries with a worse network infrastructure may be beneficiary of peer-to-peer networks.

## 1.3 Approach

The thesis will be initiated by an extensive literature review to ensure the full coverage to conduct any further research and to understand where the lack of performance may be examined. This means to choose the correct methodology and to emphasize any already covered concerns. Moreover, the literature review will be specific towards the nature of the thesis and its primary goals.

The next steps of the thesis will include further analysis of collected data, a proposition of a solution, and how to tackle the steps towards the solution. The thesis will be completed with a summary of the findings and discussion around the overall progress and if any specific topics have to be explained in further details.

The thesis will use a live implementation of a torrent client for mobile devices to highlight the effect of peer-to-peer networks. A live implementation will cover real time aspects of the mobile devices as many situations caused by user interaction may cause issues with the update. The live implementation will also show the protocol change impact of the mobile devices as simulations may not always show such physical related issues.

## 1.4 Thesis division

The thesis will be outlined in the following manner. Chapter 2 will cover, in extent, the background of the problem as well as including the literature review and any other related work. Chapter 3 will describe network address translation and how this technique impacts peer-to-peer environments. Chapter 4 will describe the approach and all the environments used throughout the thesis including the tools

and their respective setup. Chapter 5 will evaluate the approach and explain the test setup to ensure the full transparency of the results. Chapter 6 will conclude the findings and explanation from the results. Chapter 7 will include the final verdict of the findings and answer the main goals for the thesis given in Section 1.1. As well, the full analyze of the results presented in Chapter 6. Chapter 8 will present the final recommendation and future works.

# Chapter 2
# Background

This chapter will, for this single case study, describe the current structure, the current setup of the system, and overall describing how it interacts between the server and the mobile devices requesting an update. This chapter will also cover already conducted related work including the literature review.

## 2.1 Problem description of this single case study

Sony Mobile today are experiencing a major load on their content delivery network when new software is released. The released over-the-air software updates are being pushed out to a larger amount of mobile devices but with caution to not overflow the update servers [1]. This flash crowd of devices downloading at the same time causes a peak of downloads which only lasts for a short while, a day or so. Figure 2.1 shows the graph of the peak which the peer-to-peer approach hopefully can reduce. Emphasizing the fact of flash crowds, the thesis will examine the effectiveness of peer-to-peer solutions and their actual reduce in traffic load of servers.

When new software versions are made available and released as over-the-air update packages, a push notification from the internal server structure is sent out to the affected mobile devices. Downloads automatically starts if the mobile devices are well powered and if they are connected to a WiFi network. Thus this load causes high costs for Sony [1] and a peer-to-peer approach may introduce a tool to reduce this peek and endorse any manufacturer, with a similar situation, into transitioning towards a peer-to-peer approach.

### 2.1.1 Current configuration setup

To understand the thesis and the proposed solution for this case study, it is important to understand how the current configuration is designed. There are two main parts in the architecture, a client side and a server side. Both of the parts will be described in further details in the forthcoming sections. Even though the main focus for this thesis will be the server side infrastructure and implementation and not the entire chain of updates which means excluding the installation process internally on mobile devices.

For a simplified server architecture refer to Figure 2.2.

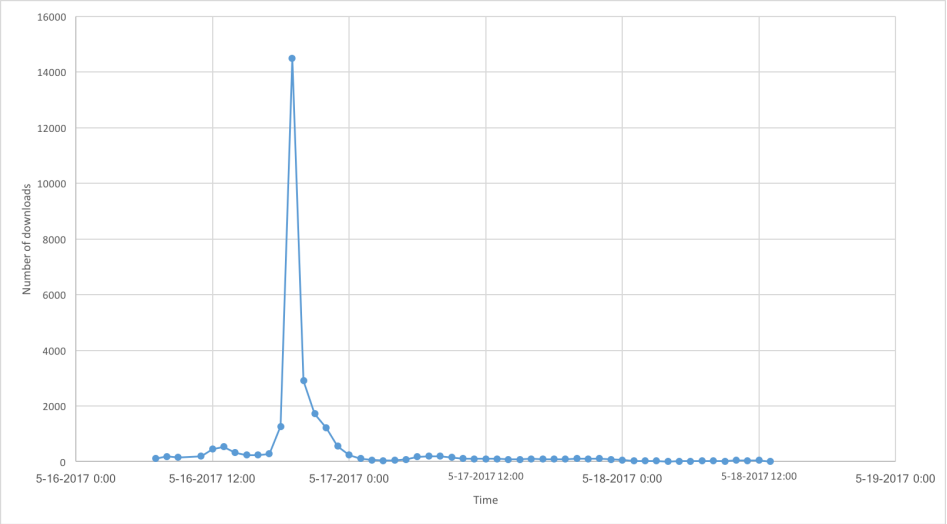


**Figure 2.1:** Presenting the graphical representation of the peak which is introduced due to the flash crowd of devices which potentially a peer-to-peer approach can reduce.

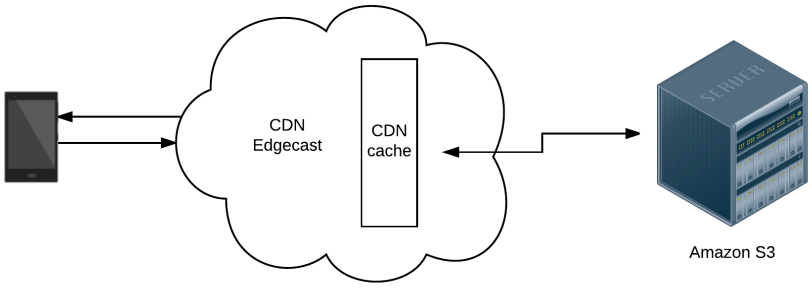


**Figure 2.2:** Simplified server structure of the current update setup.

### 2.1.2 Server

Sony is using Amazon S3 as the file server [2] which hosts all the new released software binaries, both for new application releases and for Android software releases [1]. The content delivery network is hosted by EdgeCast, a Verizon copyright trademark [3]. The distribution of the content delivery network is as of today 95 nodes worldwide [4].

### 2.1.3 Client

The client will refer to any Sony Mobile device, including mobile phones and tablets, but may as well be any other Android powered mobile device. Throughout the thesis the client, together with mobile device, will be exchangeable. The clients are continuously searching for updates through the application Software update, which is installed on the mobile devices, and always included. The graphical user interface of the application can be seen in Figure 2.3. When new requests are being made from the clients they receive a manifest file, which includes an URL to the binary file, an URL to the release notes, and other metadata which are necessary for the completion of the updates and to confirm that there is a newer version available or not.

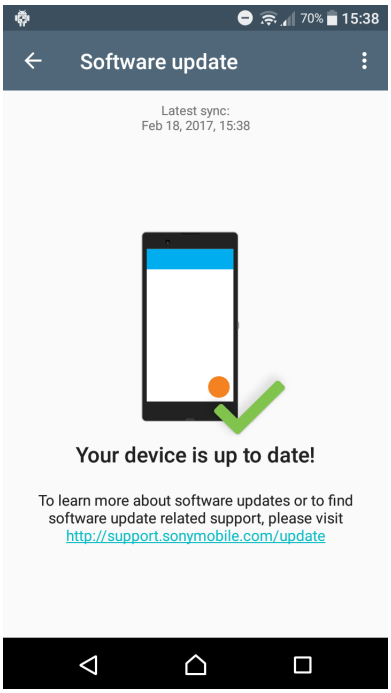


**Figure 2.3:** A screenshot from the Software update application in a Sony mobile device showing the graphical user interface.

#### New update available

Considering the scenario when there is a new available update for a mobile device. Upon receiving an HTTP request from the client the server responds with a manifest file. Upon receiving the manifest file, the client request the binary file and also the release notes from their respective URL specified in the manifest. The EdgeCast content delivery network handles the request by firstly checking its cache, since the binary file might already have been downloaded. If it is not present in the cache, the request is forwarded to Amazon S3 which is then downloaded and stored in the cache for a certain amount of time [1]. This design decision is made to decrease the request time for clients. The release notes, which are also requested by the client, may potentially include pictures, videos, and other information which regards the specific update.

### 2.1.4 Economic consequence

The thesis will examine how the transfer to a peer-to-peer model, like BitTorrent, may change the economic consequences for a manufacturer. As for this case study, Sony today does not have a flat rate price with Verizon EdgeCast and are paying per downloaded gigabyte from the content delivery network [1]. Which, as a consequence, varies in price depending on the size of the binaries present on a file server. Hence, for any manufacturer as for Sony in this case study, the larger the binary the higher the costs if not a flat rate contract is signed.

## 2.2 Protocols and infrastructure

This section will describe any specific protocols that are currently used today or will be used in the thesis. As well introduce the reader to certain network infrastructure which will be used in the thesis.

### 2.2.1 BitTorrent

The BitTorrent protocol was first introduced in 2001 by the programmer Bram Cohen [5] and is today one of the most widely used protocols for implementing peer-to-peer networks [6]. The conceptual idea behind BitTorrent is that peers, or clients, can communicate directly with each other sharing information between each other without any interference of a central entity. Consequently letting peers gain higher download speeds if many are interconnected peers [5].

#### Parts of BitTorrent

To be able to understand the architecture proposed in the thesis an introduction to the different parts of BitTorrent will be described in this section.

**.torrent-file** The .torrent-file is a metadata file which contains information which will be used by the client. Some of the information found in the .torrent-file are,

- The address to the tracker,
- Piece length,
- SHA-1 hashes of the pieces,
- Information about the files,

Which enables the clients to find the correct information and validate the data [7].

**Leech** A leech is a client which only have a partial part of the file or have just join the swarm to begin download. A leech works both as an *uploader* and a *downloader*. This means that what partial parts the leech has downloaded until a certain time the leech is also able to share with other peers connected to the swarm [6].

**Seed** A seed is a client which has fully downloaded the entire copy of the file. For BitTorrent to work there must be at least one (1) seed in the swarm. As regards to the thesis the Amazon S3 file server works as a stable seed at all times. Without this seed other peers do not have any chance of downloading the entire file [6].

**Swarm** A swarm is the set of peers, both seeders and leechers which participate in the peer-to-peer BitTorrent network [6].

**Tracker** The tracker is the bookkeeper in the protocol. The trackers responsibility is to track the progress of all the peers in the swarm and make sure that new joiners receives a list of peers they may connect to. As such the tracker is the only way for peers to find each other. The tracker is also responsible to keep track of certain information about the download progress for the entire swarm and also to ensure that the peer list is up to date [8][9].

**File division** For BitTorrent to work as intended the original file is divided into small pieces, approximately into 256kB pieces or *chunks* [8][6]. All of those pieces are hashed with SHA-1 and added into the .torrent-file as information. Upon completing downloading a piece the peer performs a check of the hash comparing it to the information in the .torrent-file. When this is completed and the integrity is confirmed the peer reports the finished download to the tracker [8].

**Choking** As BitTorrent is a peer-to-peer protocol where peers share chunks of data between each other sometimes a peer stops fulfilling its purpose and stops sharing with other peers connected to it. To ensure that the protocol does not come to a stall *choking* is introduced. A choking algorithm is to ensure that peers are exchanged if they are not providing the necessary information requested by the other peer. In BitTorrent a variant of tit-for-tat is being used which is strictly based on the downloading rate of the peers [8].

**Piece selection** The basics for each downloading peer is to obtain all the pieces of a file to obtain the entire file. Consequently the peer must be able to select in what order the pieces should be downloaded in. Depending on the implementation design BitTorrent may use the following policies,

- Strict priority,
- Rarest first,
- Random first piece,
- Endgame mode,

As well as a good selected algorithm will ensure a good performance [8].

#### Communication in BitTorrent

The communication in BitTorrent is maintained by TCP/IP [8]. In order to keep the tracker informed and updated peers periodically send updates to it. The frequency of such updates are usually set to 15min and this is maintained in the tracker [6]. The tracker protocol is a simple protocol based on Hypertext Transfer protocol (HTTP) [8][6].

**Ports** BitTorrent relies on TCP as a transport protocol. Destination ports commonly used are in the range 6881-6889. Port 6969 is designated to all tracker communication [10].

### 2.2.2 HTTP

Hypertext Transfer protocol, or just HTTP, is the most common used protocol which is the Internet is built on. HTTP is widely used for requesting either binary files or web pages. For a detailed protocol description the reader is referred to [11].

### 2.2.3 Content delivery networks (CDN)

Content delivery networks expands the basic concept of the client-server architecture by expanding the server into several distributed servers. As mentioned in Section 2.1.2 Sony takes advantages of EdgeCast's content delivery network.

Referring back to the distributed part of content delivery networks they deploy many servers in strategic points, either chosen by geographical location preference or edge points on backbone networks of Internet service providers [12]. The main concept of content delivery networks is that the content of one server, which will be referred to as the *main server*, is mirrored out to other servers called *mirror servers* [12]. The mirror servers are considered as caches of the main server and emphasize the effect of close locality to the end users. The end users are in this case referred to as clients requesting content from a web server [12]. The content delivery network itself is being updated as changes occur on the main server. This is to ensure that all the servers in the network are up to date with what is placed on the main server.

Figure 2.4[1]. shows an example distribution of a main server with several mirror servers in a country. Likewise how the communication may look like between the servers to ensure quick and painless updates of the content.

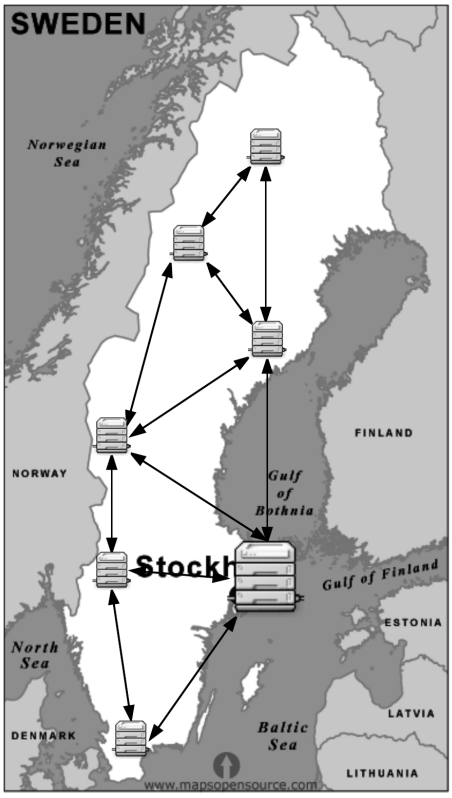


**Figure 2.4:** Figure showing how a possible CDN may look like over Sweden. One main server placed in Stockholm and mirror servers placed in different locations in the country.

[1]http://www.mapsopensource.com/sweden-capital-map-black-and-white.html, accessed 2017-04-18

## 2.3 Related work and literature review

For this thesis a literature review have been performed to gather information on the topic of content delivery and peer-to-peer networks. The literature review have also been broadened in order to cover any topics in order to understand what infrastructure issues and limitations may cause towards the performance of peer-to-peer networks. The literature review have also been directed towards the other primary objectives of the thesis as to gather any necessary knowledge and proceedings of others to explore the fact of the possible solutions others have already completed.

### 2.3.1 Security and privacy

The security of peer-to-peer protocols and WiFi have been directed towards sniffing network data and identifying how user behavior in mobile devices may be used to identify your identity. Atkinson et al. [13] explained how an external party may sniff a channel between a mobile device connected to an access point and with great accuracy identify the user of the mobile device. Atkinson et al. focused on anonymous traffic, which includes encrypted traffic between two entities. Their methodology included selecting several applications and creating user profiles. By observing user patterns they could identify an anonymous user and assess that certain user to one of the user profiles with high accuracy.

Continuing on the user patterns, Arvidsson et al. [14] studied a great amount of data of YouTube users in order to determine whether caching might help reduce traffic within a smaller geographical area. The idea Arvidsson et al. showed was that a YouTube clip requested once by a user will be requested once again by either the same or another user within the same geographical area. A geographical area could be a university campus or the same country. The results from this study shows that users in the same geographical area tend to watch the same YouTube clips as their neighbors.

Faath et al. [15] further focused on the topic of broadcast data and what it may tell a sniffer of such network characteristics. Their methodology was to sniff a large campus network over several months and by storing this data trying to learn as much as possible from this data set. The focus was both to gather information about the devices and the users behind them. One of the key findings they identified was that many of the users do not change the name of their device but continue to use the standard device name which might include both language preferences and their first or last name. They did also find that Dropbox, as an example, sends an unique identifier in broadcast. The key showings of their work included that even with technical countermeasures it is with relative ease from an attacker to identify an unique user just by looking at the broadcasted data from its mobile device.

### 2.3.2 Energy consumption

The field of energy consumption in mobile peer-to-peer networks focuses mainly on how the peer-to-peer quality and traffic affects the battery life. Nurminen and Nöyränen [16] focus solely on how peer-to-peer network behavior affects the

battery life of mobile devices. They performed real time tests showing that a mobile BitTorrent client is more vulnerable to greater power consumption if the network it is connected to is of bad quality. Their conclusion in the paper stated that higher bit rate networks, such as a WiFi connection, will be better for mobile devices in peer-to-peer networks in terms of battery consumption. What also was interesting was the finding that mobile devices should, to maximize their battery efficiency, only act as full peers until their download is completed. Kelényi et al. [17] also confirms the findings by Nurminen and Nöyränen. The battery efficiency is dependent on the time spent sharing rather than the amount of data. Consequently, higher bit rate networks can send data quicker comparing to slower bit rates thus consuming less battery time as both downloading and sharing takes less time.

Further on the topic, Kouyoumdjieva and Karlsson [18] focused on ad-hoc opportunistic networks in urban environments. Opportunistic networks are very similar to peer-to-peer networks but focuses more on device-to-device networking. Their study focused on whenever a mobile operator trying to push out an update to see if data offloading could be utilized. The data offload would be utilized using mobile devices in an urban area, for example a street. They distinguished between data fulfilled nodes and data seeking nodes as to examine the energy consumption. Their findings do however state that in a crowded environment data fulfilled devices still help in the data offloading consequently decreasing the time data seeking nodes are active in search for a device to download from. Their findings can be summarized to whenever there are many devices in an area, the better the battery performance as the bit rate of the network increases. This is the same conclusion Nurminen and Nöyränen [16] got in their study.

### 2.3.3 Cross-traffic and infrastructure decisions

Another topic which was highlighted in the literature review was inter Internet service provider (inter-ISP) traffic. Cho et al. [19] highlight the issue in which inter-ISP traffic creates a lot of economic loss for Internet service providers. They highlight that since BitTorrent uses swarming and choses peers randomly it emphasizes the need to select peers as with content delivery networks servers. The peers selection must be smarter to avoid crossing inter-ISP traffic. What they propose is a solution of *iCodes*. Which means that each Internet service provider provides a token to the peers within it and when peers are to be selected, the ones with the same iCode are chosen firstly.

Bindal et al. [20] emphasizes the same economic lost as Cho et al. and do also propose a solution for better choosing peers instead of randomly selecting, as per native BitTorrent behavior. Although, they do highlight that other studies has shown that randomly selected peers do keep the performance of the network at almost optimal levels. However, from their own simulation findings they do get almost the same level of optimality as with a random selection. Consequently, there is an economic gain from a Internet service providers perspective to adapt a smarter peer selection algorithm while still maintaining the same level of network quality. Streit and da Silva Rodrigues [21] do also propose another policy of peer selection. Their proposal is *Quota-Based Peer Selection* (QBPS). However, their

policy focuses on the user experience. The QBPS policy builds on that lower bit rate peers receives a greater amount of opportunities to connect to more sources. Their findings shows an increase in performance when using QBPS in comparison with the regular random peer selection policy.

### 2.3.4 Live streaming and sharing

Streaming of media have transformed the shape of how we as society digest media. Many of the major streaming services, such as Spotify and Netflix, uses either a peer-to-peer network or content delivery network architecture. Kreitz and Niemelä [22] describes the architecture in Spotify and how the client behaves when streaming the live media. Ellis et al. [23] further analysis the Spotify peer-to-peer architecture. They go in depth with the interaction between other peers and how the overlaying server-client co-operates with it. Goldmann and Kreitz [24] focus on the same topic but expand their research into exploring user patterns. Goldmann and Kreitz examine whether the network behavior differs between weekdays and weekends. They do find discrepancies but smaller than what they expected.

Adhikari et al. [25] study the architecture of Netflix. They find that Netflix utilizes several content delivery networks depending on what streaming quality the user is capable of receiving. Likewise, as with Sony's infrastructure [1], Netflix outsources many of their services to the Amazon cloud service, Amazon Web Service [26]. To be able to maintain a high quality of experience for the end-users Netflix uses several content delivery networks which are working together. However, the effort is to delivery highest possible quality of experience to the end-user. Due to the several content delivery networks, Netflix uses a trade-off which uses static choosing the content delivery network depending on geographical location of the end-user. As such, it does not focus entirely on the quality of experience for the end-user [25]. Netflix, just as Sony, are using manifest files to deliver the correct content to the end-user [1][25]. The manifest file includes several important information fields in order for the client to be able to connect and stream the correct file.

Hammami et al. [27] continue on the topic and describe the drawbacks of BitTorrent in terms of initial startup delay and how pieces are selected. Next Hammami et al. focuses on the peer selection. They highlight the inefficiency in the pure BitTorrent protocol in the peer selection phases but do consider another algorithm called Give-to-Get [27]. Farahbakhsh et al. [9] examined the popularity of the content and the diversity of the different types of multimedia content available. Consequently, video content was the most popular multimedia category. One of the key aspects which Farahbakhsh et al. explained is a direct consequence of the multimedia contents popularity. If all other multimedia categories would be deleted of a tracker, which roughly was half of the content, the download capacity would not be affected [9].

Abdelhalim et al. [28] continue to examine and describe the efficiency with peer-to-peer networks, such as BitTorrent, to deliver and stream large multimedia files. Abdelhalim et al. focuses mainly on Scalable Video Coding (SVC) together with BitTorrent to allow the media streams be encoded into several substreams. The substreams are encoded into different qualities and be distributed concurrently

into a peer-to-peer network [28]. Abdelhalim et al. also highlights the issues as Hammami et al. considering the poor native piece selection algorithm. Abdelhalim et al. expands the selection by utilizing a sliding window. This sliding window is used both as a buffer and to guarantee that the right pieces are requested at the right time. To tackle the native peer selection Abdelhalim et al. adapts part of the tracker to ensure that peers closest to the requesting peer are selected to avoid any unnecessary network delays [28].

Continuing on streaming, Kouyoumdjieva and Karlsson [29] assumes that by year 2020 75% of mobile traffic will be real-time data. Their paper focuses on opportunistic communication to enable offloading of the mobile network. Opportunistic communication is working without any overlying infrastructure and enables two or more devices share information between each other [29]. In relation to this thesis, Kouyoumdjieva and Karlsson states that software update is one of the topics which could be offloaded using device-to-device communication. This is a direct consequence of the nature of software updates. Software updates are delay-tolerant comparing to a live streaming song. Kouyoumdjieva and Karlsson continues describing the interaction between two devices. As with BitTorrent, a device must show interest in a certain file or other media. A key finding in Kouyoumdjieva and Karlsson's paper is the nature of how well opportunistic communication works. For it to be efficient at all, it all boils down to node density within a close radius of the devices [29].

### 2.3.5 Content delivery and peer-to-peer networks hybrids

Content delivery networks have recently grown in popularity to ensure high quality delivery to end-users independently of their location. Just as Sony with the distribution of software updates. A field closely related to this thesis is the use of hybrid content delivery networks with an overlying structure using a peer-to-peer network. Lu et al. [30] examine the purpose, the implementation, and the gaining of such an approach. Lu et al. begin with describing any related work but pinpoint that this hybrid model is a combination where non of the content delivery network or the peer-to-peer network is more important than the other. The main implementation is that each mirror server in the content delivery network has its own peer-to-peer network [30]. Another key aspect from the paper is the comparison between content delivery network and peer-to-peer solutions. Lu et al. did find many common points as to being able to compare the advantages and disadvantages between the two solutions to enable file sharing for many end-users.

Further in the paper Lu et al. presents two methods for which the content delivery network may use the peer-to-peer. The two different approaches are *peer-aided content deliver peer-to-peer network* (PAC) or *content delivery network aided peer-to-peer network* (CAP) [30]. The latter is the more popular one today as the content delivery network works as a back-up if peers are starving in the peer-to-peer network. As to compare the latter solution to an actual implementation a similar approach is deployed and implemented by Spotify. This can be seen and further analyzed in [22], [23], and [24]

Mondal et al. [31] continue with the same assumptions as Lu et al. in which content delivery networks may become more efficient with an overlying peer-to-

peer network. Mondal et al. focuses on decreasing the inter-ISP traffic in which peer-to-peer networks tend to create a lot thus implicating an economic loss for the Internet service providers [19][20]. Their paper further emphasizes the importance of a well defined and well working peer selection algorithm. Thus, emphasizing to refrain from using the native random peer selection present in the BitTorrent protocol. Comparing to Lu et al. Mondal et al. focuses more on the geographical location of peers [31].

### 2.3.6 Mobile data increasing constantly

Cisco [32] released it newest VNI forecast for 2016-2021 explaining that the mobile traffic will continue to grow over the course of the next few years. According to the forecast approximately 12 billion mobile devices will by 2021 be connected to the mobile networks globally. Cisco do also point out the connection speed of mobile networks will continue to increase as well as the amount of data produced by these devices.

One other key fact presented by Cisco [32] is the decrease of wired connections. The trend presented by Cisco point towards a steady increase of wireless connections for the networks. As well the traffic of video streaming will continue to increase being almost 75% of the total network traffic.

Chapter 3

# Network Address Translation

This chapter will describe and introduce Network Address Translation (NAT) and how NAT devices affects the overall feasibility to change from a client-server model into a peer-to-peer model for over-the-air updates. NAT's, on all levels, impose a blockage on the original end-to-end principle of which Internet was designed around. This chapter will describe the issues which arise from NAT's and also how to tackle them with hole punching techniques to enable the end-to-end principle between peers. This chapter will also highlight the overall impact on peer-to-peer networks in terms of number of neighbors and performance of peers behind a NAT device in the infrastructure.

### 3.0.1 Network Address Translation – NAT44

#### Description

Networks address translation allows an end-user to use several devices through only one public reachable IP-address. Figure 3.1 shows how the NAT device, in most cases a home router, divides the public IP-address and translates them into private IP-addresses.

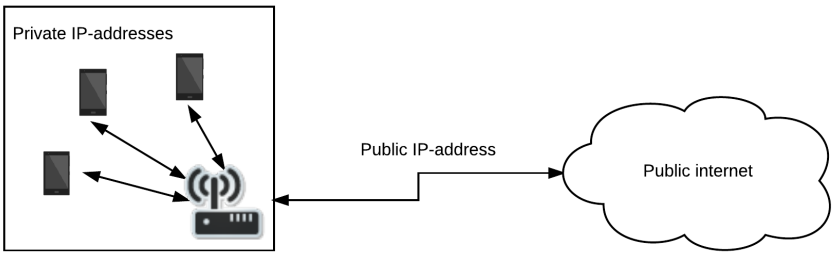


**Figure 3.1:** Figure showing a possible solution using NAT44 in a home environment.

### Potential drawbacks and advantages

NAT44 ruins the natural end-to-end principle of how the Internet was originally implemented. This is due to the translation of one public IP-address into several private IP-addresses as presented in Figure 3.1. On the other hand, as NAT44 usually is deployed within a home environment, not allowing end-to-end connections into the private network creates a natural firewall and security for the end-users within this private network. Consequently, keeping the private network more secure comparing to a native end-to-end connectivity. Another advantage with NAT44 is that end-users have the possibility to alter with their own private network. This option creates the possibility to change the behavior accordingly to the users needs.

## 3.0.2 Carrier-grade Network Address Translation – NAT444

### Description

As with NAT44, described in Section 3.0.1, carrier-grade network address translation allows one public IP-address to be shared and distributed between several users. However, as opposite to NAT44, NAT444 distributes a pool of public IP-addresses which are distributed and used by several home routers. Consequently, the setup using NAT444 creates an additional layer of NAT. Figure 3.2 shows a potential implementation. A considerable advantage for NAT44 is that the public IP-address is not changing when establishing new sessions whereas within NAT444 a client is assigned an IP-address from the shared pool within the specific NAT444 [33].

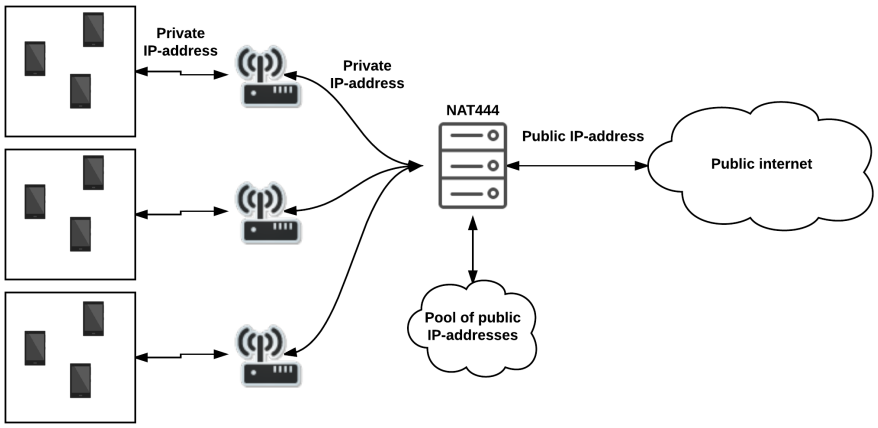


**Figure 3.2:** Figure showing a possible solution using NAT444 for a carrier and how it is connected to the private a home environments.

#### Potential drawbacks and advantages

NAT444, or also known as CGN, shares the same drawbacks as NAT44 including the broken end-to-end principle. In this thesis NAT444 will be used. However, due to the double layer of NAT the situation becomes even worse. Due to the nature of NAT444 creating an additional layer, the end-users lose the possibility to alter anything on the second level NAT. This issue could be considered as a limitation for the end-users free movement on the Internet. As NAT444 may use a pool of public IP-addresses, which are shared within an Internet service provider, one user can never be sure to receive the same IP-address for two different sessions. Upon entering the public Internet, as specified in Figure 3.2, the NAT444 assigns an available public IP-address to the requester in its private network.

### 3.0.3 Avoiding double NAT situations

Network address translation is nothing which, by any means, can be avoided in the current situation with IPv4 IP-addresses still being used excessively [34]. Although, end-to-end solutions and protocols, such as the peer-to-peer protocol BitTorrent, are still available. Even with the presence of NAT devices connected to the peer-to-peer swarm, they are still functioning behind the NAT device due to the existence of techniques for hole punching and NAT traversal [34].

In the report *Report on the Implications of Carrier Grade Network Address Translators* [35] the authors emphasized the fact that with NAT444 the overall performance must be shared with more devices comparing to a solution without a NAT444. As certain applications requires several concurrent sessions, the presence of a NAT444 may cause the number of allocated ports towards each of the devices to decrease drastically. The authors also emphasize that with the presence of a NAT444, certain VPN applications may be broken and malfunction. This is due to the fact that NAT444 assigns IP addresses randomly and for tunnel sessions this pipeline may be broken more easily comparing to a single level NAT.

Port forwarding also becomes an issue in the presence of a multilevel NAT situation. The authors of [35] highlight the issue of not being able to configure the NAT444, hence, port forwarding for applications becomes a major fault. This is however possible for NAT44. As shown by Figure 3.2, many of the first level NAT's (NAT44) are reachable and in most cases integrated into home network routers [33]. Although, with NAT444 those configurations are off limits for end-users and Internet service providers deploying a NAT444 are reluctant to set those configurations free to alter for the end-users. The authors continued to emphasize that certain applications will require much additional configuration to make them work. One of them specifically is the peer-to-peer protocol BitTorrent, which eventually will work but new traversal techniques and configurations must be presented to cope with the multilevel situation. In Section 3.0.4 more techniques will be presented.

### 3.0.4 NAT traversal and hole punching techniques

To deal with the issues of firewalls and NAT's on all levels, hole punching techniques have been presented. The common ground for the hole punching techniques

is to *enter into a locked door without user involvement*. As one of the main drawbacks for hole punching techniques, presented by different papers, is not being able to find a common technique. This is due to the not standardized nature of NAT's and firewalls. Müller et al. [36] not only mentioned the difference in manufacturers but also model-wise within the same manufacturer. Consequently, creating issues with trying to establish a standardized solution for NAT traversal.

### 3.0.5 Feasible techniques

This section will introduce the most common ways of hole punching and their respective implementation for a work around of the NAT and firewall issues.

#### Universal Plug and Play – UPnP

Universal Plug and Play (UPnP) was created as a set of protocols to try to standardize the way of opening holes of firewalls and routers to enable a peer-to-peer connection between two devices [37]. Consequently, UPnP allows a device to co-operate with the NAT device in order to punch a hole through it [38]. This co-operation is based on previously agreed terms which are present in the protocol itself. This agreement allows the device to automatically negotiate and open holes in the NAT [39] without the need of end-user interference. However, as the name *Universal* emphasizes, UPnP can be used in other environments as well and not only in NAT's and firewalls.

Grimmett and O'Neill [40] in their article states that UPnP, as a protocol, allows UPnP enabled devices to tell others about what services and configurations it is subscripted to. The UPnP device itself becomes exposed to the rest of the Internet and will allow certain configuration being made on it by other UPnP enabled devices. They do emphasize that UPnP is not well scalable due to the multicast messages being sent. As such, UPnP enabled devices usually are within a home network. They did also highlight that there might be a certain security risk in enabling UPnP for the device, as it might be tampered with.

One protocol in particular to highlight, which is based on UPnP, is the Internet Gateway Device protocol (IGD protocol). NAT's and firewall may, to the Internet, expose themselves as Internet Gateway Devices. As they are usually placed at the end points of LAN networks they will allow other devices to receive information and allowing them to perform port forwarding on the announced IGD device [41][40].

#### Session Traversal Utilities for NAT – STUN

Session traversal utilities for NAT (STUN) is a protocol that enables the devices connecting to the NAT to identify how a certain communication path looks like along the way to the NAT device itself [38]. This identification is specially interesting for situation where multilevel NAT's (see Section 3.0.2) arises [38]. Although, not being very accurate [38] it can sometimes help in the guessing of the behavior of the NAT. STUN makes this guessing possible due to the structured nature of NAT's. When a NAT is to be considered as *symmetric*, it assigns port numbers in a certain logical way and not ad-hoc randomly. STUN allows the devices connecting to the NAT to explore this logic and then predict the end-points. This

structured way makes the guessing very efficient [38]. Ford et al. [38] showed that 80% of the UDP sessions were successful using STUN. They do also emphasize that many of the hole punching techniques presented and used today are having a high successful rate. Much more in one level NAT situations rather than in multilevel situations, making it an effective technique to pass NAT devices.

Richter et al. [42] argues that a very permissive approach of STUN will produce a result closest to the actual behavior of the NAT444. The NAT traversal using STUN allows the devices to guess the behavior of the NAT further down the road. Which means that predictions on multilevel NAT's may be bypassed as well.

Fu et al. [43] emphasized the fact that STUN is an explicit protocol of which a peer, most likely using another peer without interference of a third-party, explores the network. From that guessing the two peers uses hole punching to establish a connection between themselves.

STUN, as protocol, may be programmed to query an external STUN server as well. Lutu et al. [44], when deploying their own client, used an external STUN server to query for the public IP-addresses of the peer. Consequently, making STUN a very versatile hole punching technique.

### Third-party relay

To ensure peer-to-peer functionality each of the peers in a swarm must know all the others end-points in their respective network to be able to establish a connection between them. As NAT's, both NAT44 and NAT444, creates walls between the peers a trusted third-party relay may be used [38]. This trusted third-party, which for example may be a server, will ensure that the two peers wanting to establish a connection between them may properly do so. The part the third-party plays is to basically ensure that the two connections are mapped to each other, as they both share their end-points with the common server. Figure 3.3 shows the logic of a third-party relay. Although a third-party relay will ensure the end-to-end connectivity of peers it results in a performance drop, higher delays of communication, and creates an additional party which must be maintained [38]. However, in terms of durability and robustness, using a third-party relay will ensure the highest quality of service for the peers in the swarm [38]. A third-party relay will ensure that most of the connections will be successful.

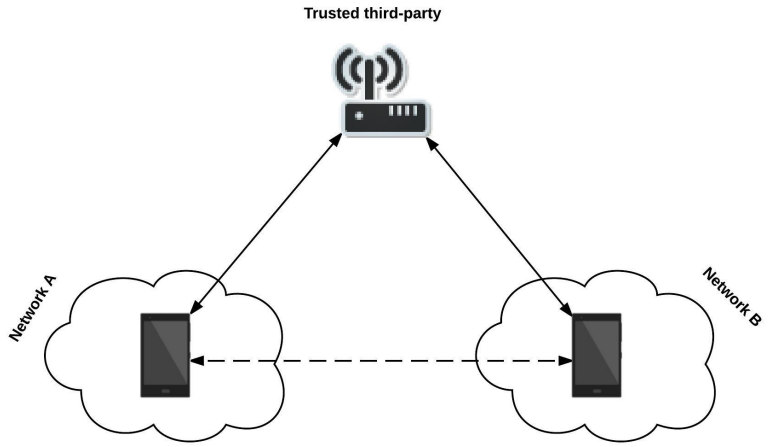


**Figure 3.3:** Figure showing how a trusted third-party allows two peers behind NAT's to find the respective peers end-point.

**Traversal Using Relays around NAT – TURN** The traversal using relays around NAT (TURN) protocol is also used together with a third-party relay. This is also an effort to standardize the protocol for third-party relays. As stated in RFC5766[45], it creates additional costs and complexity for the party deploying the TURN relay server. As with any trusted third-party relay, it sets high expectations on the server itself not to produce any delays for the peers.

### NAT Loopback – Hairpinning

In RFC 5128 [46], a solution called *Hairpinning* is presented. Hairpinning allows two devices behind the same NAT, to connect directly to each other. The NAT device will know for a fact that they are connected to the same private network. What occurs when a NAT recognizes a package, it loop backs it to the known private destination IP-address, as it is a part of the same pool of connected peers [42]. Figure 3.4 shows the logic of hairpinning.

However, as also mentioned in the RFC and by Ford et al. [38], not all NAT devices implement this functionality. Although, Ford et al. highlight that this type of functionality is becoming more and more popular.

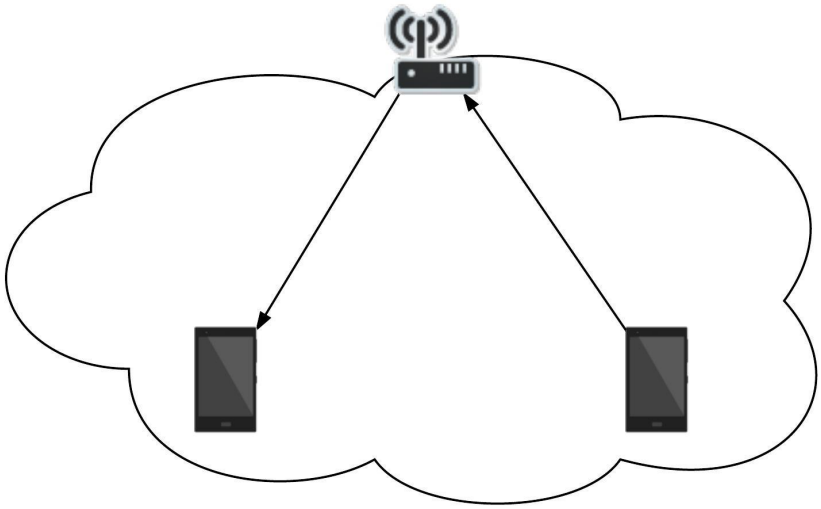

**Figure 3.4:** Figure showing how hairpinning works in general practice.

## 3.0.6 Transition to IPv6

Another topic which arises along the infrastructure of the Internet is the transition towards IPv6. Internet today uses, in extent, IPv4 as the main addressing protocol. According to Internet Assigned Numbers Authority (IANA) the last available IPv4 address space was assigned in 2011 [47][48]. Consequently, forcing IPv6 to be successful to ensure unique global addresses for devices connected to the Internet. In May 2014 [49] Internet Corporation For Assigned Names and Numbers (ICANN) announced that the final allocated block of IPv4 IP-addresses started to be assigned and even further emphasizing the need to adjust and move towards IPv6.

### 3.0.7 IPv4 to IPv6 – transition with the help of NAT444

One of the tactics for a smooth transition of IPv4 to IPv6 is to use dual stacks network, as Levin and Schmidt [48] are presenting in their study. During the transition phase the two existing infrastructures will co-exist. This is to allow the full usage of IPv6 until the overall network infrastructure has been fully implemented in all parties. Levin and Schmidt predicts that such an approach will be available for a longer time until the change has been fully made [48]. Cisco [50], as one of the leading global infrastructure providers, pinpoints the same approach as Levin and Schmidt. However, Cisco in addition adds the complexity of a NAT444 to deal with the lack of IPv4 IP-addresses [51]. In 2008 Russ Housley, the former chairman of IETF, also confirmed that NAT's in general are necessary for a smooth transition from IPv4 to IPv6 [52]. From [52] a table presenting IETF's approach for the transition to IPv6 included a dual stack network with the use of NAT444 for Internet service providers.

Beeharry and Nowbutsing [53] showed that the transition towards IPv6 is slow but steadily increasing. Dell et al. [54] did an analysis for the progress in Australia and they highlighted some issues in terms of where to begin the transition. The question they asked was to whether the transition should begin in the core network and work out towards the edge or the opposite. They concluded that the transition should begin with the edges and work towards the core network. They did also highlight the fact that NAT444 is useful to help with the transition. One of their key findings is that there is a resistance of organizations being *the first* to move to IPv6. Although, still emphasizing that some governments around the Asia-Pacific (APAC) region are starting to push towards a faster transition to IPv6, as a result, the momentum of the transition is gaining.

#### IPv4 addresses – Market target

Interestingly, due to the slow pace of movement towards IPv6 [53], IPv4 address spaces has become a target for organizations and Internet service providers to start selling public IPv4 addresses. Akplogan [55] also emphasized the fact that IPv4 IP-addresses have been fully allocated, but due to the original allocation, some parts of the world still have unallocated addresses available. Aklogan also emphasized that ICANN does allow transfers of IP-addresses, however, due to the unstructured and uncontrolled nature prices may differ which creates a market for IPv4 address spaces.

Howard [56] did an extensive comparison up to what price IPv4 IP-addresses may rise before deploying a NAT444 becomes useful for an Internet service provider. Also, considering the top price of what subscribers are willing to pay before naturally the need to move to IPv6 becomes the case [56].

### 3.0.8 NAT444 not causing any significant performance drop for users

As a NAT444 must ensure to examine all incoming and outgoing packages some delays may be expected. Bocchi et al. [57] in extent tried several targets to examine the impact of deploying a NAT444 for an Internet service provider. Specially targeting the end-users and their experience of quality decrease. Bocchi et al.

tried to compare public IP-addresses versus private IP-addresses and comparing them to each other. One of their key findings in the paper was that the point of significant impact was the number of hops a package had to make before ending up at the other side. However, the conclusion they made was that NAT444 is a mature tool and do not affect the overall performance for end-users. Thus, is terms of deploying a NAT444 environment, they concluded that for an Internet service provider, the overall user-experience will not be impacted by a NAT444.

### 3.0.9 NAT's decreases the number of neighbors for peers in BitTorrent swarms

In this section references to *unconnectable* peers refer to devices behind a NAT or firewall which do not have a public routable IP-address. References to *connectable* peers will refer to the right opposite, thus to devices which have a public routable IP-address.

One of the key impacts of deploying a NAT is the division of many devices sharing only one, or several, public routable IP addresses. This fact has a significant impact on the unconnectable peers when it comes to public swarms in peer-to-peer protocols like BitTorrent [58]. Yoshida and Nakao [58] continued their analysis and their methodology to measure the swarm sizes which concluded that over 90% of the swarms have less than 20% connectable peers. They highlighted in their study that the general number of connectable in a swarm is just over 8%. Liu et al. [34] emphasizes the same number of approximately 20% of the peers being connectable in swarms in general.

Due to the level of unconnectable peers in swarms many studies have tried to establish if conectable peers actually benefit from those unconnectable peers. D'Acunto et al. [59] showed that connectable peers do benefit of the larger sphere of unconnectable peers. This benefit, which arises for the connectable peers, is that they do not have to share the bandwidth with more peers than needed. However, they did also show that after crossing a threshold of having more unconnectable peers comparing to connectable will in fact have a major impact on the upload to the swarm. The unconnectable peers will have a very hard time to reach the limit of any share ratio which has been set by the site or organization.

However, Liu et al. [34] showed the opposite as well. Concluding that connectable peers do not benefit the presence of unconnectable peers due to the nature of peer-to-peer protocols, specifically BitTorrent. The logic behind BitTorrent functionality is that peers within a swarm share bits and pieces among each other. Liu et al. showed that having to many peers behind NAT's and firewalls causes the natural sharing to diminish, thus, creating inefficient swarms. However, non of the mentioned articles provided any actual numbers in terms of how worse the efficiency actually becomes.

#### Download performance of NAT devices decreases

Liu et al. [34] continued to argue that the performance of unconnectable peers is decreasing due to the fact that they are not chosen by other peers. As stated in Section 2.2.1, BitTorrent uses tit-for-tat methods to chose the right peers to

connect to, even if the peer selection is randomized. In the most general way, the peers are selected depending on their performance towards others and peers not sharing will be chocked more often then connectable peers. Due to this reason unconnectable devices will take longer time to download pieces from the swarms as non of the other peers will want to connect to them. Their results also emphasize the fact that the fewer neighbors in a swarm the worse performance in terms of download speed. This is due to the fact that peers must be able to connect to each other to establish an end-to-end connection.

# Chapter 4
# Approach

This chapter will describe the approach of how the thesis was conducted and executed. This chapter will also describe the different parts and a proposition of solution will be given.

## 4.1 Methodology

The literature review has been used to gather a thorough understanding of already tested approaches. The literature review has been used as an information source to examine any drawbacks which others stepped onto. Those issues are described in Section 2.3. The literature review has presented tools and useful statistics which has been used in this thesis as well. Those findings has been summarized for understanding.

Next a conceptual architecture has been created as to understand the direction of the implementation. The architecture worked as a template from which an analysis was made and to be used for persuasion of the problem. The main purpose of the conceptual model is to give the reader the understanding of what was lacking in the literature review and which will be covered in the experiments.

Finally the experiments were conducted. The experiments were conducted to tests the drawbacks highlighted in the literature review and comparing them to the current setup. The main idea was to see how a peer-to-peer model, using for example BitTorrent as the protocol, did help reduce the peak load of flash crowds upon releasing a new software update. The experiments are presented and introduced in Section 5.1. The experiments were created to examine different performances in different setups. This was to ensure a valid data set to further analysis of the results.

## 4.2 Proposed solution of the test environment

This sections will describe in details what the proposed solution for the test environment looked like. The idea is to describe the different parts and how this was implemented to conduct the thesis.

### 4.2.1 Requirements

As a part of the initialization phase of the thesis some common requirements were gathered to ensure the scope of the thesis was as expected. The requirements are presented in the forthcoming list. The requirements were used to ensure the scope and approach was as expected from the organization in this singular study case and to deal with the primary objectives of the thesis. The primary objectives in terms of implementation and statistics gathering.

- Start, pause, and stop download over WiFi.
- Resume downloads if paused manually.
- Resume downloads if WiFi connection is lost.
- The file should be seeded for at least a factor X before being removed.
- The file should be seeded for at least the time before installation is complete.
- If reusing code, explore what licenses the code is under.

### 4.2.2 Proposed implementation

The ensure the functionality was met an entire change of architecture must occur. The proposed implementation must meet the requirements from the BitTorrent protocol, see Section 2.2.1 for details. Thus, both the client and server must be edited and also ensuring a tracker functionality.

The proposal for this thesis was to ensure,

- Implementing or using a third-party BitTorrent client on Android mobile devices.
- Using Amazon S3 file server as both the storage and use the integrated tracker functionality.
- Explicitly not using the ordinary Update Center application to avoid any disruptions.

As such, keeping things to the minimum in terms of complexity as the objective of the thesis was to look into the effort of the implementation and the possible gain due to the protocol change from a client-server model to a peer-to-peer based model instead.

#### BitTorrent client for Android mobile devices

The BitTorrent client was decided to be very simple and installed as a third party application and not to run as a background task. The reason for this is that having it as an application makes its easier to ensure that it will not interrupt any other background services which were running on the Android mobile devices. The other parts of the BitTorrent client is to have a simple GUI to start, pause, and stop any ongoing downloads. The GUI will also work to gather statistics from the application in order to be able to compare with the current setup and the possible

gain. The third-party torrent client decided to be used in the experiments was DrTorrent, as it has been used by others when conducting live implementations of a BitTorrent setup (see Section 4.3 for details about DrTorent and its related articles).

#### Amazon S3

As one of the most important parts of the BitTorrent protocol is to have a bookkeeper controlling all the peers connected to the swarm. Amazon S3, which is the service used to store the binary files in this case study, offers a tracker functionality. The other advantage using Amazon S3 is that the service also becomes a steady seed. This means that there is always at least one seed with the full content. To cope with the stability of the change this is a key fact. There are also no specific costs related to use the tracker-functionality but rather it is included in the subscription.

The usage of the tracker and permanent seed in Amazon S3 will both keep the new configuration simple and strengthen the maintainability for future works.

#### Software update application

Software update application is the application which is used today by Sony to download the software updates. The decision to not use this in the thesis is that it is complex and dependent to many parts which are not in scope for this work. An example is the dependency to the installation manager. As such, the decision to avoid changing in the implementation will ensure that the focus remains where it should be.

The avoidance also affects the implementation positively. This is due to the fact that many internal dependencies to the download manager creates internal complexity which is not preferable to change for this work. However, the conceptual conviction for the concept have still been met even if the client is not exactly the same. The conceptual understanding will be therefore later transfered into the real Software update application depending on the outcome of the results. For this thesis this has been one key fact to keep out of scope.

## 4.3 Tools

This section will describe the tools used in the thesis.

### 4.3.1 Android studio

To develop the android client the official Android Studio was used. As it allowed a full feature android environment this was mutually agreed to be the most efficient tool to use. Android Studio did also come with many examples and free available code to reuse and most importantly to learn from.

### 4.3.2 Lucidchart.com

Lucidchart.com is an online tool for creating flowcharts. Some of the figures in this thesis were created using this online tool.

### 4.3.3 DrTorrent

DrTorrent is a third-party client written by Bori András and is available through Google Play [60]. DrTorrent has been used by others, e.g. [61] and [62], and has served as a starting point for any live tests conducted throughout this thesis.

### 4.3.4 Eclipse

Eclipse is an open-source tool and has become one of the most popular tools for programmers. It has been used throughout the thesis for any necessary programming.

### 4.3.5 Internal tools

Many of the internal statistics were gathered using internal tools used by the architect, developers, and managers. Not all will be described in this subsection as the relevancy to do so is not in scope with the thesis.

# Chapter 5

# Live experiments and evaluation of scenarios

This chapter will evaluate and describe the experiments conducted in the thesis and their respective purpose. After completing the literature review, some areas were found where there was a need for further evaluation. This chapter will introduce the different test scenarios together with an description of their respective area of solution and what the focus of each experiment was. The results from each of the experiments are presented in Chapter 6.

## 5.1 Live experiments using mobile devices

This section will present the live test scenarios. The introduction will include both the purpose behind the experiments and also their respective setup.

### 5.1.1 Aim with the live experiments

The aim of the live experiments is to further develop the general knowledge of peer-to-peer protocols being efficient even with certain network limitations. However, comparing to other conducted research, the aim will be to examine how badly the peer-to-peer networks work comparing to an ideal scenario without any network infrastructures limitations. Since prior conducted experiments used BitTorrent as the peer-to-peer protocol this thesis will continue to do so as well.

From the literature review, it was found that the focus of those experiments, in regards to any infrastructure limitations, focused on the download speeds, the amount of neighboring peers, and how unconnectable peers affect the swarms themselves. However, the actual gain in terms of how much peer-to-peer networks reduce a flash crowd peak was not mentioned and will in this thesis be extended using three different scenarios, which will be described in further details in the forthcoming section.

#### Division of live experiments

The purpose of the division was to create a dataset which could expand the well-know effect of peer-to-peer and emphasize any already conducted research, as

to tying the hypothesis that fewer amount of unconnectable peers creates worse efficieny for BitTorrent swarms. As stated in Section 3.0.9 the impact of NAT presence in the global networks infrastructure pushes the ratio of connectable peers in BitTorrent swarms down to only 20%. However, the lack of numbers proving the real effect of such bad ratio, in terms of actual data saving for flash crowds, is missing.

Keeping this in mind the experiment division is presented in the forthcoming list.

1. An ideal-case swarm – All peers connectable.
2. A worst-case swarm – All peers unconnectable.
3. A general-case swarm – 20% peers connectable.

### 5.1.2 Experiment execution

Returning back to Section 2.1, in which the feasibility of decreasing the peak occurred by a flash crowd of devices downloading over-the-air updates. In order to cope and relate to this fact the experiments were divided into two parts, as per the forthcoming list.

- Client-server test
- Peer-to-peer test

The decision to divide the test scenarios into two parts was to compare downloads using the current client-server setup as the reference point. As already stated, the experiments were meant to simulate a flash crowd to replicate a live situation when new software is released for over-the-air updates and the most devices connect at the same time. Hence, emphasizing that when the experiments were performed, all of the mobile devices were cleaned form older downloads and all did initiate the download of the file at the same time. This way of proceeding included both the client-server and the peer-to-peer test. The mobile devices were connected to a WiFi home network. The detailed test setup is presented in forthcoming sections. Before any of the peer-to-peer experiment the .torrent-file was initialized by Amazon S3, as presented in Section 5.1.3.

#### An ideal-case swarm – All peers connectable

The ideal-case swarm experiment was the starting point of the analysis and as well used as the starting point of the experiments. All ports were open and an ideal end-to-end principle was maintained. The detailed setup for this scenario is presented in Table 5.2 and using the aforementioned execution methodology.

#### A worst-case swarm – All peers unconnectable

The worst-case swarm experiment was the second experiment conducted. The worst-case swarm was conducted to evaluate how NAT devices affect the overall

performance of the offloading. All the ports were closed to ensure that the end-to-end principle was disrupted. The detailed setup is presented in Table 5.3 and the execution occurred with the aforementioned methodology.

### A general-case swarm – 20% peers connectable

The general-case swarm experiment was the third experiment conducted. One peer had an open port and the rest were closed. The decision for this experiment was to emphasize other conducted research and to examine the most general case of connectable peers in swarms. This general number was provided by the literature review. The detailed setup is presented in Table 5.4 and was executed using the aforementioned methodology.

### Test setup

In order ensure full transparency and reproducibility of any of the experiments the test setup will be presented and described. All of the experiments used the general setup presented in Table 5.1. Figure 5.1 shows a graphical representation of how the private network looked like. The deviations will be highlighted for each of the experiments respectively. In order to explain the port forwarding made in the home router, an *open* port made the device connectable and a *closed* port made the device unconnectable. The devices which were chosen for the experiments were Sony Mobile smartphones running on different Android versions. The reason for this was that a decision was made as to diverse the operating system, showing that the peer-to-peer solution works on a wider population of Android operating systems.

**Table 5.1:** Standard setup for peer-to-peer testing and for client-server model in terms of execution.

| | |
|---|---|
| Number of devices | 5 |
| DrTorrent version | 1.3.7 |
| WiFi connection | 2.4GHz |
| Number of experiments per scenario | 17 |
| File size | 129241752bytes |
| Experiment frequency | 2 experiments/h |

All of the mobile devices were installed with DrTorrent, a third-party BitTorrent client, with standard settings. UPnP was deactivated, on both the access point and in the clients, to maintain full control over the ports and not allowing the mobile devices to open ports as pleased.

The frequency of the experiments was two experiment an hour, one client-server experiment and then one peer-to-peer experiment. This was due to the granularity of the billing report taken from Amazon S3 with the data. As well, to ensure that no other data traffic affected the data sets, a new bucket on Amazon S3 was created.

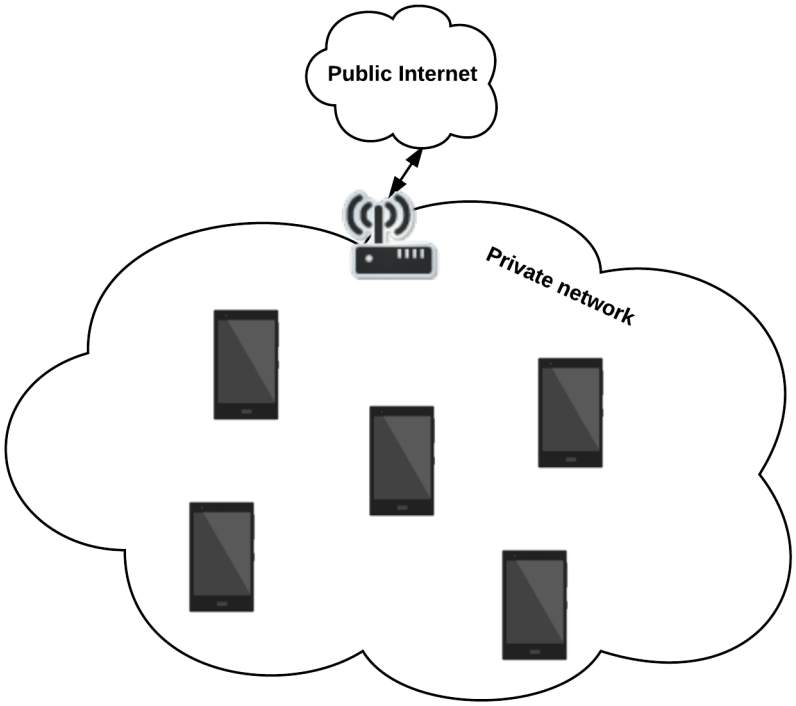


**Figure 5.1:** The figure presenting the test environment. All the mobile devices for the test scenarios were connected to the same access point over WiFi 2.4Ghz creating a private network.

### Setup deviation – An ideal-case swarm

Table 5.2 presents the deviated test configuration for the ideal-case scenario.

**Table 5.2:** Setup for the ideal-case swarm which was used both for the client-server and the peer-to-peer test.

| Device 1 | |
|---|---|
| Hardware/Model | F5321 |
| Android version | 7.1.1 |
| Software build | 34.3.A.0.129 |
| Port | 6886 (open) |
| Device 2 | |
| Hardware/Model | G8141 |
| Android version | 7.1.1 |
| Software build | 45.0.A.1.206 |
| Port | 6886 (open) |
| Device 3 | |
| Hardware/Model | E6553 |
| Android version | 5.0.2 |
| Software build | 28.0.A.8.251 |
| Port | 6886 (open) |
| Device 4 | |
| Hardware/Model | C5503 |
| Android version | 4.4.4 |
| Software build | 10.5.1.A.0.292 |
| Port | 6886 (open) |
| Device 5 | |
| Hardware/Model | E6603 |
| Android version | 7.1.1 |
| Software build | 32.4.A.0.72 |
| Port | 6886 (open) |

### Setup deviation – A worst-case swarm

Table 5.3 presents the deviated test configuration for the worst-case scenario.

**Table 5.3:** Setup for the worst-case swarm which was used both for the client-server and the peer-to-peer test.

| Device 1 | |
|---|---|
| Hardware/Model | F5321 |
| Android version | 7.1.1 |
| Software build | 34.3.A.0.129 |
| Port | 5612 (closed) |
| Device 2 | |
| Hardware/Model | G8141 |
| Android version | 7.1.1 |
| Software build | 45.0.A.1.206 |
| Port | 6984 (closed) |
| Device 3 | |
| Hardware/Model | E6553 |
| Android version | 5.0.2 |
| Software build | 28.0.A.8.251 |
| Port | 5478 (closed) |
| Device 4 | |
| Hardware/Model | C5503 |
| Android version | 4.4.4 |
| Software build | 10.5.1.A.0.292 |
| Port | 7654 (closed) |
| Device 5 | |
| Hardware/Model | E6603 |
| Android version | 7.1.1 |
| Software build | 32.4.A.0.72 |
| Port | 5586 (closed) |

### Setup deviation – A general-case swarm

Table 5.4 presents the deviated test configuration for the general-case scenario.

**Table 5.4:** Setup for the general-case swarm which was used both for the client-server and the peer-to-peer test.

| Device 1 | |
|---|---|
| Hardware/Model | F5321 |
| Android version | 7.1.1 |
| Software build | 34.3.A.0.129 |
| Port | 6886 (open) |
| Device 2 | |
| Hardware/Model | G8141 |
| Android version | 7.1.1 |
| Software build | 45.0.A.1.206 |
| Port | 6984 (closed) |
| Device 3 | |
| Hardware/Model | E6553 |
| Android version | 5.0.2 |
| Software build | 28.0.A.8.251 |
| Port | 5478 (closed) |
| Device 4 | |
| Hardware/Model | C5503 |
| Android version | 4.4.4 |
| Software build | 10.5.1.A.0.292 |
| Port | 7654 (closed) |
| Device 5 | |
| Hardware/Model | E6603 |
| Android version | 7.1.1 |
| Software build | 32.4.A.0.72 |
| Port | 5586 (closed) |

### 5.1.3 Initial live test – proof of concept

The first live experiment was conducted with *proof of concept* in mind. The setup for it can be seen in the forthcoming list. The main reason for this initial live test was to confirm the functionality in Amazon web services and as well in Amazon S3. This test was used to resolve any uncertainties which were still present to create a smoother path for the other experiments.

- Third-party mobile torrent client.
- An open-source video file.
- A .torrent-file created and hosted by Amazon S3.
- Some additional peers, computers and other mobile devices.

#### Initial phase

The initiation phase began with creating and initiating a .torrent-file in Amazon S3 using Amazon's guide of creating one [63]. Amazon S3 is just a normal file storage environment but the internal structure of folders are called buckets [64]. All data is stored within buckets instead of folders. For the sake of the test a bucket was created and an open-source video file [65] was uploaded into the newly created bucket. To create the .torrent-file a request must be made by a client. The .torrent-file is automatically generated upon this request. The actual link is presented in Link 5.1, however, some information is removed due to company secrets.

`https://.../thesisbittorrent/cscw94_10_m1.mpg?torrent` (5.1)

The .torrent-file is generated by Amazon S3 upon receiving the flag `?torrent` after the `GET HTTP` request in the link.

Upon successfully generating the .torrent-file, it was opened in the decided third-party mobile torrent client, DrTorrent. To prove the concept of how BitTorrent works with the Amazon tracker, the original mobile device, which was the only peer downloading the full copy from Amazon using peer-to-peer instead of client-server. The mobile device continued to be present in the swarm and acted as a seed. When the original mobile device continued to seed the rest of the devices mentioned in the aforesaid list connected to the swarm.

# Chapter 6
# Results

This chapter will present the findings divided per each test scenario described in Section 5.1.1. Small comments will be mentioned but an overall discussion and verdict will be given in Chapter 7.

## 6.1 Results from the live experiments

This section will be divided for each of the test scenario presenting the data in both tables and figures also introducing the calculations used to calculate the numbers.

### Calculations

Data downloaded with the client-server model is presented as $D_{cs}$ and data downloaded through the peer-to-peer model is presented as $D_{p2p}$. $G$ is the gain in bytes moving from a client-server model to a peer-to-peer model and is calculated according to (6.1)

$$G = D_{cs} - D_{p2p} \tag{6.1}$$

The actual gain in percentage, $W$, was calculated according to (6.2).

$$W = \frac{G}{D_{cs}} \tag{6.2}$$

The min and max points $\overline{min_{gain}}$ and $\overline{max_{gain}}$ are given by calculation of the average gain from the results in the ideal-case scenario and worst-case scenario.

### Results – An ideal-case swarm

This section will present the data collected from the ideal-case swarm scenario. Table 6.1 shows the collected data from each of the test which was executed. Due to the division explained in Section 5.1.2, the downloaded bytes from both the client-server experiment and peer-to-peer are presented separately, likewise the actual gain for each of the tests. Further, the average of the dataset was calculated together with the standard deviation for the entire dataset.

**Table 6.1:** Table shows the results in terms of downloaded data from the ideal-case scenario including the standard deviation for all examined parts.

| Test # | Client-server data, $D_{cs}$ (bytes) | P2P data, $D_{p2p}$ (bytes) | Actual gain, $W$ (%) |
|---|---|---|---|
| 1 | 884726431 | 577600749 | 34,71 |
| 2 | 761562484 | 509850156 | 33,05 |
| 3 | 741940321 | 423160302 | 42,97 |
| 4 | 728391364 | 323839653 | 55,54 |
| 5 | 710099876 | 544573733 | 23,31 |
| 6 | 715494040 | 362082134 | 49,39 |
| 7 | 709859689 | 474326044 | 33,18 |
| 8 | 720465286 | 338592105 | 53,00 |
| 9 | 827842779 | 425837304 | 48,56 |
| 10 | 710681681 | 454582557 | 36,04 |
| 11 | 697225464 | 415773012 | 40,37 |
| 12 | 720666942 | 383869073 | 46,73 |
| 13 | 697939307 | 348246410 | 50,10 |
| 14 | 835891507 | 449903676 | 46,76 |
| 15 | 700502106 | 383857202 | 45,20 |
| 16 | 706134025 | 548292478 | 22,35 |
| 17 | 687817321 | 481974617 | 29,93 |
| Average | 738661213 | 437732229 | 40,66 |
| Standard deviation | 55027794 | 74835786 | 9,77 |

Figure 6.1 shows the graphical representation of the collected dataset. The figure shows the downloaded data, both from the client-server and peer-to-peer experiments, the average gain, and the gain for each of the data points. The standard deviation for the gain is presented for each measurement point.

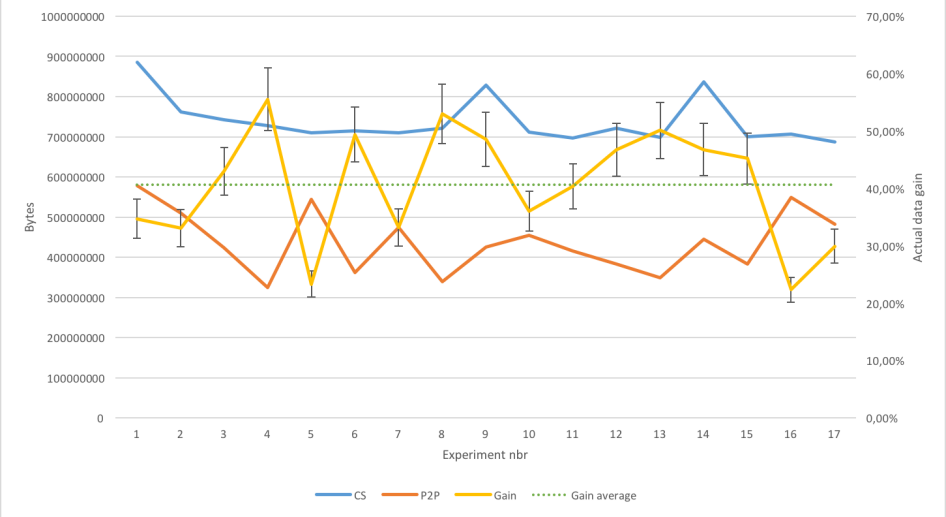


**Figure 6.1:** The figure is presenting the data set from the ideal-case scenario.

What can be seen from the collected dataset, for the ideal-case swarm, is that the average gain is 40,66% but the results vary with almost a 10% standard deviation. The results shows that even with distributed effort there is a maximum point of how much can be saved for flash crowds.

### Results – A worst-case swarm

This section will present the data collected from the worst-case swarm scenario. Table 6.2 shows the collected data from each of the tests which was executed. Due to the division explained in Section 5.1.2, the downloaded bytes from both the client-server experiment and peer-to-peer are presented separately, likewise the actual gain for each of the test. Further, the average of the dataset was calculated together with the standard deviation for the entire dataset.

**Table 6.2:** Table shows the results in terms of downloaded data from the worst-case scenario including the standard deviation for all examined parts.

| Test # | Client-server data, $D_{cs}$ (bytes) | P2P data, $D_{p2p}$ (bytes) | Actual gain, $W$ (%) |
|---|---|---|---|
| 1 | 686699544 | 662736678 | 3,49 |
| 2 | 724445808 | 661782708 | 8,65 |
| 3 | 680023068 | 662212685 | 2,62 |
| 4 | 681309333 | 659639651 | 3,18 |
| 5 | 693343289 | 660939127 | 4,67 |
| 6 | 688830985 | 660733349 | 4,08 |
| 7 | 689145319 | 658258600 | 4,48 |
| 8 | 678729274 | 658505869 | 2,98 |
| 9 | 675502120 | 663737229 | 1,74 |
| 10 | 696261753 | 658835282 | 5,38 |
| 11 | 672598559 | 661878408 | 1,59 |
| 12 | 666567935 | 660789526 | 0,87 |
| 13 | 668957039 | 663279914 | 0,85 |
| 14 | 675377542 | 663091431 | 1,82 |
| 15 | 668957039 | 661229307 | 1,16 |
| 16 | 672407967 | 663354430 | 1,35 |
| 17 | 674553943 | 659046948 | 2,30 |
| Average | 681982972 | 661179479 | 3,01 |
| Standard deviation | 13653053 | 1752137 | 1,95 |

Figure 6.2 shows the graphical representation of the collected dataset. The figure shows the downloaded data, both from the client-server and peer-to-peer experiments, the average gain, and the gain for each of the data points. The standard deviation for the gain is presented for each measurement point.

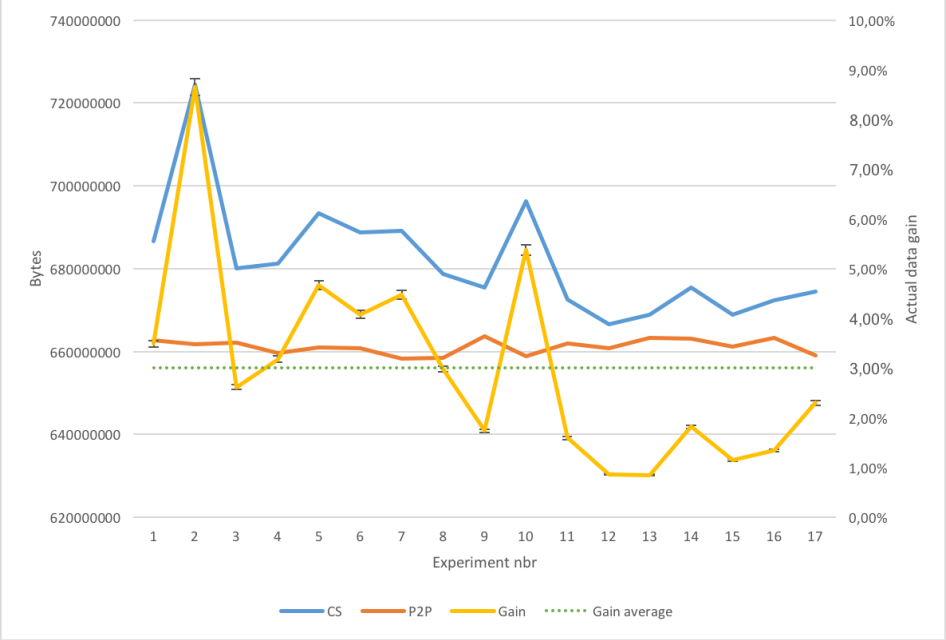


**Figure 6.2:** The figure is presenting the data set from the worst-case scenario.

The worst-case showed that there is a small potential gain if all the peers are unconnectable. However, the standard deviation was almost 2% which, together with the average gain of 3,01%, makes an unconnectable swarm just as efficient as the original client-server approach when it comes to flash crowds. The results should have been 0%, but the billing report from Amazon Web Services did point towards a decrease of data from the server.

### Results – A general-case swarm

This section will present the data collected from the general-case swarm scenario. Table 6.3 shows the collected data from each of the tests which was executed. Due to the division explained in Section 5.1.2, the downloaded bytes from both the client-server experiment and peer-to-peer are presented separately, likewise the actual gain for each of the test. Further, the average of the dataset was calculated together with the standard deviation for the entire dataset.

**Table 6.3:** Table shows the results in terms of downloaded data from the general-case scenario including the standard deviation for all examined parts.

| Test # | Client-server data, $D_{cs}$ (bytes) | P2P data, $D_{p2p}$ (bytes) | Actual gain, $W$ (%) |
|---|---|---|---|
| 1 | 671487008 | 534038483 | 20,47 |
| 2 | 670284319 | 553946845 | 17,36 |
| 3 | 692465596 | 560240678 | 19,09 |
| 4 | 669429724 | 553794293 | 17,27 |
| 5 | 672407967 | 579568861 | 13,81 |
| 6 | 668566992 | 525168157 | 21,45 |
| 7 | 672607059 | 505486159 | 24,85 |
| 8 | 667231575 | 579405504 | 13,16 |
| 9 | 666982393 | 562450077 | 15,67 |
| 10 | 665372555 | 596948645 | 10,28 |
| 11 | 672705424 | 566555086 | 15,78 |
| 12 | 673568348 | 568060848 | 15,66 |
| 13 | 674274296 | 572563231 | 15,08 |
| 14 | 672894843 | 572809516 | 14,87 |
| 15 | 668408087 | 573996706 | 14,12 |
| 16 | 672298527 | 470595860 | 30,00 |
| 17 | 674344311 | 529759994 | 21,44 |
| Average | 672078178 | 553258173 | 17,67 |
| Standard deviation | 5740080 | 30510932 | 4,67 |

Figure 6.3 shows the graphical representation of the collected dataset. The figure shows the downloaded data, both from the client-server and peer-to-peer experiments, the average gain, and the gain for each of the data points. The standard deviation for the gain is presented for each measurement point.

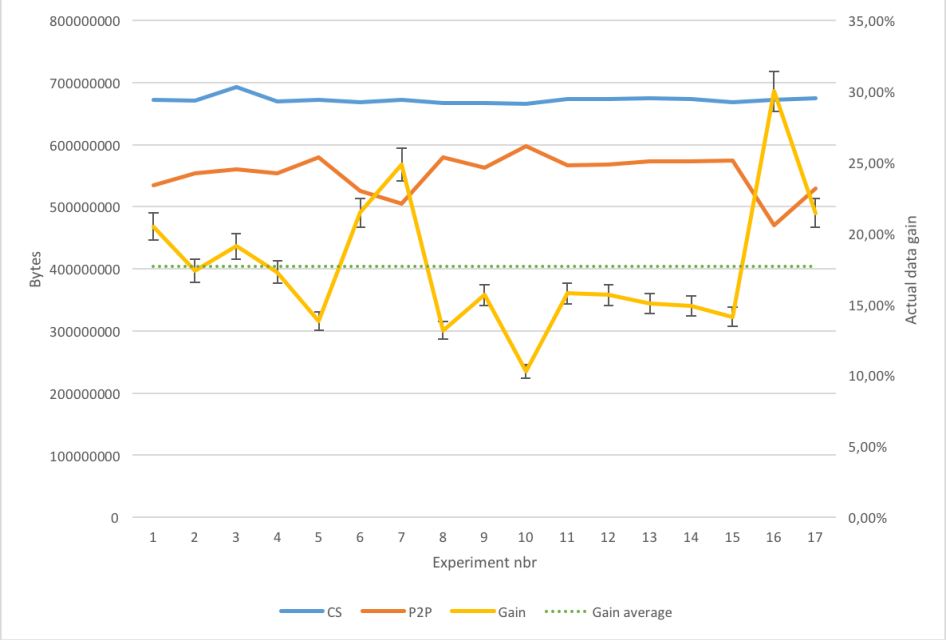


**Figure 6.3:** The figure is presenting the data set from the general-case scenario.

From the dataset in the general-case scenario it can be concluded that the average gain is following the amount of connectable peers in the swarm. As the scenario included 20% connectable peers, the average gain almost reached 20%. The actual average gain was 17,67%.

### Combined view of the data sets

Figure 6.4 shows the average of the tests and how the gain proceeds from the minimum $\overline{min_{gain}} = 3,01\%$ towards the maximum $\overline{max_{gain}} = 40,66\%$. The figure also shows how the gain tends to becomes more and more unstable and fluctuates with more connectable peers in the swarm.

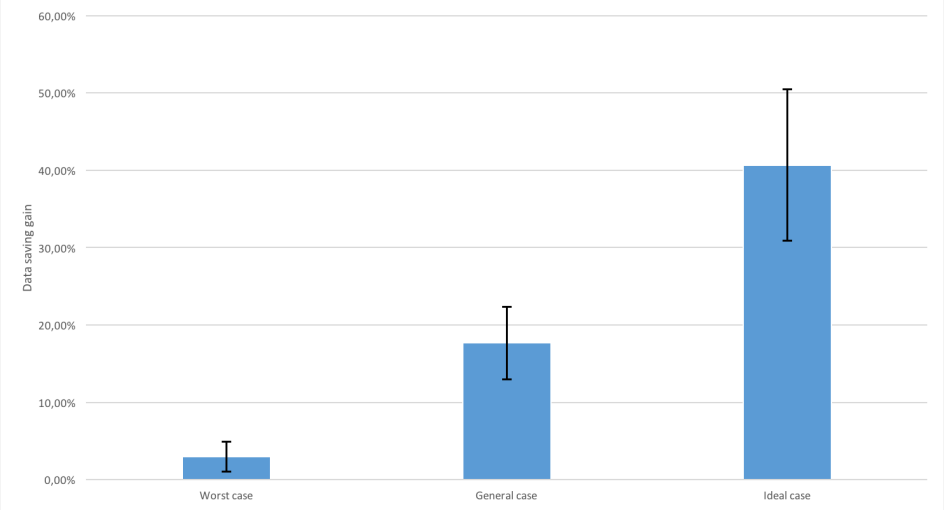


**Figure 6.4:** The figure is presenting the combined data set from all of the scenarios together with the increasing standard deviation towards the ideal-case scenario.

# Chapter 7

# Discussion

This chapter will present the discussion and summarize the findings and answering the primary objectives given in the beginning of this thesis. The results from Chapter 6 will be tied with the literature review to further persuade the reader of the accuracy of the findings and feasibility of changing into a peer-to-peer model from a client-server model.

## 7.1 Performance impact of NAT

In order to examine the performance of peer-to-peer networks this thesis extended the field and examined the overall performance impact of NAT devices in the global infrastructure. These issues, in the wider spectra, are impacting the feasibility of any peer-to-peer networks. Consequently, the presence of NAT's in the infrastructure, on any level, creates an additional level of penetration for any traffic flowing through the NAT's. As such, NAT presence disrupt the natural end-to-end principle but studies shows that it does not decrease the performance for end-users but limits the network modifiability for them.

### 7.1.1 Hairpinning

Hairpinning will become more and more important with the presence of NAT444 and as such will be highlighted in the discussion. As presented in Figure 3.4, it will become more and more important when the private networks, as with NAT444, becomes larger. If the private network itself is large the more gain there will be to provide hairpinning functionality as more peers will be able to identify each other in their respective private network. Although, Grundemann [66] stated that the impact of hairpinning in NAT can cause performance issues, hence the selection of not implementing it in the NAT devices. Ford et al. [38] continued on the topic of hairpinning and stating that not all devices implement hairpinning due to the increased level of complexity. This complexity grows from the fact that NAT devices with hairpinning functionality must ensure to check both the private address pool together with the public.

### 7.1.2 Swarm sizes are affected by the presence of NAT but the performance is still maintained

One of the key findings from the literature, which were extended with the live experiments, is how badly smaller connectable swarms perform comparing to an ideal swarm scenario, where all peers were connectable. This is a direct consequence of NAT presence, as shown Section 3.0.9. Any NAT presence in the global infrastructure must be taken into account when deploying any peer-to-peer solution for over-the-air updates, or any software release mechanism.

The consequence of such network obstacles opens the field of deploying hole punching techniques whom can traverse and open up ports in order to establish new connections. Section 3.0.4 presents the most commonly deployed techniques today but also highlights the effectiveness of each of them. To ensure cost effectiveness, in terms of moving into a peer-to-peer model, the manufacturer must consider the effort of deploying certain techniques but maintaining high feasibility and cost effectiveness. Using a trusted third party relay creates an environment with the highest accuracy, in terms of successfully establishing end-to-end connections between peers. Although, deploying such a third party creates an additional layer of complexity in the manufacturers own infrastructure and, as such, they must ensure to plan additional resources for any maintenance.

Moving onto other popular solutions, one of the most common and most widely deployed techniques is UPnP. The characteristics of UPnP is that it a set of protocols. As such, implementing it into mobile devices only requires a single effort when implementing the torrent-client and confirming the functionality of the mobile devices. It will, however, require some effort of selecting the appropriate sets of protocols to suit the peer-to-peer infrastructure for the torrent-client.

Although, as presented in Section 3.0.9, unconnectable peers in swarms are badly affected by their unconnectability due to the native implementation of BitTorrent, as the peer-to-peer protocol. This is due to the fact that since the unconnectable peers cannot upload as much as the connectable ones, the unconnetable peers are not chosen as frequently as the connectable. Interestingly enough, even though unconnectable peers are not chosen as frequent, the overall performance of devices behind a NAT is not affected, as presented in Section 3.0.8. This important fact also emphasizes that NAT devices will continue to be present in the global infrastructure, as they are cheaply deployed and most of the Internet users are not widely affected. Which concludes that any manufacturer will have to continue deploying hole punching techniques to cope with such limitations. Section 3.0.3 tells that avoiding double NAT situations is preferable for the end-users, as those will loose the possibility to alter any configuration on the NAT444. This is a crucial point for the transition towards a peer-to-peer model. Even though many users are not familiar with port forwarding in their home routers a few end-users will be. More open ports will create a larger population of connectable peers. Without this possibility even power users will not be able to open any ports if such configuration is kept beyond reach.

### 7.1.3 Energy consumption not being affected negatively

One of the key objectives was to examine how a peer-to-peer swarm will affect the battery life of mobile devices. From Section 2.3.2, key findings point towards that using high bit rate networks with low latency does not negatively affect the energy consumption more than any other approach. What will have to be taken into consideration for the manufacturer is to examine how long any mobile devices should stay in swarms before they end their contribution. This is due to that time being spent in swarms is the main reason for higher energy consumption for the mobile devices. Consequently, high bit rate networks means less time spent downloading and uploading. As such, there is not native impact of peer-to-peer networks but a manufacturer must ensure to set a policy to enlighten the amount of time each mobile device must be connected to the swarm.

## 7.2 Data saving for flash crowds

Finally, mowing onto the additional key aspect of the thesis was to prove the potential economic gain of transitioning into a peer-to-peer model from a client-server model, which is tested with the singular case study. The live experiments performed in this thesis were meant to extend the common knowledge of peer-to-peer solutions being more efficient for offloading central entities comparing to a client-server but also emphasizing how much a manufacturer actually can save. As the original focus of the thesis was flash crowds for the single case study, in order to decrease the original peak when new software is released and most of the users download the packages at the same time. This economic saving is restricted, on average, from 3,01% to 40,66%, ranging from a worst-case scenario towards an ideal-case scenario. Figure 6.4 shows the actual results taken from the scenarios implemented in this thesis which concludes that there are benefits even with all peers being unconnectable. This is although uncertain due to the billing report extracted from Amazon Web Services. Amazon provides billing reports on byte level and the worst-case scenario should have shown 0%, as non of the devices had open ports. However, the extracted report showed a decrease of data even for the worst-case scenario. This fact may be caused due to discrepancies of the billing report but since Amazon are paid per byte, the reports should be highly accurate from a billing perspective. Although, emphasizing the standard deviation of almost 2%, the worst-case is performing just as a client-server model and with more tests this discrepancy might have been even further reduced. Another interesting finding from the result is that the uncertainty grows as the swarms moves towards an ideal-case scenario. The standard deviation is most stable when the amount of unconnectable peers is less. Hence, the more ideal-case swarm the more the actual result will vary from time to time and from update to update. However, as seen in Section 3.0.9, the general number of connectable peers in general swarms reach approximately 20%, hence, the actual benefits and gain for this case study is on average 17,67%.

# Chapter 8
# Conclusion

This chapter will present a final recommendation based on the results and discussion. It will also present future works which, in some cases, have been highlighted in the thesis but left uncommented due to limitations of the thesis.

## 8.1 Recommendation

Concluding the thesis, a recommendation would be that even though with all the obstacles being present in the global Internet infrastructure, a transition towards a peer-to-peer model is feasible and potentially giving almost 20% of data saving for flash crowds when releasing new over-the-air updates. For this singular case study, the effort will be to implement a torrent-client for mobile devices which implements UPnP to increase the successful end-to-end connections. Minimizing the infrastructure impact, Amazon S3 should be utilized with its tracker functionality as the service is already present. Such a setup will be cost efficient and decrease the data traffic by at least 17,66%.

Looking forward, such an approach will also be future-proof considering that a transition towards IPv6 may be plausible but is not yet implemented for a wider audience. The reason for this is that the logic itself will not be changed even though the addresses themselves will be updated. There will be consequently a smaller effort to just further enhance the mobile torrent-client for it to work as Amazon S3 does support IPv6 already, considering this singular case study. Although, this recommendation still applies for all manufacturers which want to transit to a peer-to-peer approach.

## 8.2 Future works

Due to the limitations of this thesis, open questions towards the affect of mobile devices will be left untouched. Hence, the following points may be picked up by others to evaluate in further details.

- Further look into the mobile devices. Including dependencies and limitations in Android as the operating system and further digging into the impact of mobile devices in a peer-to-peer environment. Mobile devices are constantly under development and in the future they might have even better network

capabilities and battery technology which might further open up for more complex protocols.

- Further developing BitTorrent with geographical limitations, as presented by some articles, to extend the capabilities of BitTorrent peer selecting algorithm. This point will be to explore the efficiency of BitTorrent and remove any of the randomness present in the protocol. An uprising field of study is hybrid-CDN networks, where a content delivery network is creating its own edge peer-to-peer network to further enhance the sharing between peers.
- Further look into the presented hole punching techniques to see explore the possibilities for mobile devices to efficiently utilize such techniques. As well as exploring if certain techniques may be better suited for mobile networks or whether the same applies for WiFi connectivity.
- Further look into how the transition towards IPv6 may impact over-the-air updates. The field is starting to be explored but there seems to be much more to find.

# References


[1] Magnus Thuresson. Conversation with my thesis advisor.

[2] Amazon. Amazon s3. https://aws.amazon.com/s3/. Last visited 2017-02-07.

[3] Verizon. Edgecast content delivery network service. https://www.verizondigitalmedia.com/platform/edgecast-cdn/. Last visited 2017-02-07.

[4] Verizon. Edgecast network structure. https://www.verizondigitalmedia.com/our-network/network-overview/. 2017-02-18.

[5] Unknown. Bittorrent of bram cohen. http://history-computer.com/Internet/Conquering/BitTorrent.html. Last visited 2017-02-16.

[6] Matteo Varvello, Moritz Steiner, and Koen Laevens. Understanding bittorrent: A reality check from the isp's perspective. *Computer Networks*, 56(3):1054 – 1065, 2012. (1) Complex Dynamic Networks (2) {P2P} Network Measurement.

[7] N. Gaddam and A. Potluri. Study of bittorrent for file sharing in ad hoc networks. In *2009 Fifth International Conference on Wireless Communication and Sensor Networks (WCSN)*, pages 1–6, Dec 2009.

[8] Bram Cohen. Incentives build robustness in bittorrent, 2003.

[9] R. Farahbakhsh, N. Crespi, Á. Cuevas, R. Cuevas, and R. González. Understanding the evolution of multimedia content in the internet through bittorrent glasses. *IEEE Network*, 27(6):80–88, November 2013.

[10] Wireshark. Bittorrent wiki. https://wiki.wireshark.org/BitTorrent. Last visited 2017-04-22.

[11] R. Fielding et al. Hypertext transfer protocol – http/1.1. RFC 2616, RFC Editor, June 1999.

[12] P. Hillmann, T. Uhlig, G. D. Rodosek, and O. Rose. Modeling the location selection of mirror servers in content delivery networks. In *2016 IEEE International Congress on Big Data (BigData Congress)*, pages 438–445, June 2016.

[13] John S. Atkinson, John E. Mitchell, Miguel Rio, and George Matich. Your wifi is leaking: What do your mobile apps gossip about you? *Future Generation Computer Systems*, pages –, 2016.

[14] Å. Arvidsson, M. Du, A. Aurelius, and M. Kihl. Analysis of user demand patterns and locality for youtube traffic. In *Proceedings of the 2013 25th International Teletraffic Congress (ITC)*, pages 1–9, Sept 2013.

[15] M. Faath, R. Winter, and F. Weisshaar. How broadcast data reveals your identity and social graph. In *2016 International Wireless Communications and Mobile Computing Conference (IWCMC)*, pages 357–362, Sept 2016.

[16] J. K. Nurminen and J. Noyranen. Energy-consumption in mobile peer-to-peer - quantitative results from file sharing. In *2008 5th IEEE Consumer Communications and Networking Conference*, pages 729–733, Jan 2008.

[17] I. Kelényi, Á. Ludányi, and J. K. Nurminen. Distributed bittorrent proxy for energy efficient mobile content sharing. In *2011 The 14th International Symposium on Wireless Personal Multimedia Communications (WPMC)*, pages 1–5, Oct 2011.

[18] S. T. Kouyoumdjieva and G. Karlsson. Energy-aware opportunistic mobile data offloading for users in urban environments. In *2015 IFIP Networking Conference (IFIP Networking)*, pages 1–9, May 2015.

[19] K. Cho, H. Jung, M. Lee, D. Ko, T. Kwon, and Y. Choi. How can an isp merge with a cdn? *IEEE Communications Magazine*, 49(10):156–162, Oct 2011.

[20] Ruchir Bindal, Pei Cao, William Chan, Jan Medved, George Suwala, Tony Bates, and Amy Zhang. Improving traffic locality in bittorrent via biased neighbor selection. In *Proceedings of the 26th IEEE International Conference on Distributed Computing Systems*, ICDCS '06, pages 66–, Washington, DC, USA, 2006. IEEE Computer Society.

[21] Ananda Gorck Streit and Carlo Kleber da Silva Rodrigues. Improving bittorrent's peer selection for multimedia content on-demand delivery. *CoRR*, abs/1512.03796, 2015.

[22] G. Kreitz and F. Niemela. Spotify – large scale, low latency, p2p music-on-demand streaming. In *2010 IEEE Tenth International Conference on Peer-to-Peer Computing (P2P)*, pages 1–10, Aug 2010.

[23] M. Ellis, S. D. Strowes, and C. Perkins. An experimental study of client-side spotify peering behaviour. In *2011 IEEE 36th Conference on Local Computer Networks*, pages 267–270, Oct 2011.

[24] M. Goldmann and G. Kreitz. Measurements on the spotify peer-assisted music-on-demand streaming system. In *2011 IEEE International Conference on Peer-to-Peer Computing*, pages 206–211, Aug 2011.

[25] V. K. Adhikari, Y. Guo, F. Hao, V. Hilt, Z. L. Zhang, M. Varvello, and M. Steiner. Measurement study of netflix, hulu, and a tale of three cdns. *IEEE/ACM Transactions on Networking*, 23(6):1984–1997, Dec 2015.

[26] Amazon. Amazon web services (aws) - cloud computing services. https://aws.amazon.com/. Last visited 2017-02-28.

[27] C. Hammami, A. Gazdar, I. Jemili, and A. Belghith. Study of vod streaming on bittorrent. In *2015 International Symposium on Networks, Computers and Communications (ISNCC)*, pages 1–6, May 2015.

[28] A. Abdelhalim, T. Ahmed, H. Walid-Khaled, and S. Matsuoka. Using bittorrent and svc for efficient video sharing and streaming. In *2012 IEEE Symposium on Computers and Communications (ISCC)*, pages 000537–000543, July 2012.

[29] S. T. Kouyoumdjieva and G. Karlsson. Device-to-device mobile data offloading for music streaming. In *2016 IFIP Networking Conference (IFIP Networking) and Workshops*, pages 377–385, May 2016.

[30] ZhiHui Lu, Ye Wang, and Yang Richard Yang. An analysis and comparison of cdn-p2p-hybrid content delivery system and model. *JCM*, 7(3):232–245, 2012.

[31] Amit Mondal, Ionut Trestian, Zhen Qin, and Aleksandar Kuzmanovic. P2p as a cdn: A new service model for file sharing. *Computer Networks*, 56(14):3233 – 3246, 2012.

[32] Cisco. Cisco vni global mobile data traffic forecast. 2016-2021. http://www.cisco.com/c/en/us/solutions/service-provider/visual-networking-index-vni/vni-infographic.html. Last visited 2017-04-19.

[33] Dan Wing. Nat tutorial. IETF 78, Maastricht, July 2010.

[34] Yangyang Liu, Le Chang, and Jianping Pan. On the performance and fairness of bittorrent-like data swarming systems with nat devices. *Computer Networks*, 59:197 – 212, 2014.

[35] Brian Aitken. Report on the implications of carrier grade network address translators. Technical report, InterConnect Communications, 2013.

[36] A. Müller, G. Carle, and A. Klenk. Behavior and classification of nat devices and implications for nat traversal. *IEEE Network*, 22(5):14–19, September 2008.

[37] Yangyang Liu and Jianping Pan. The impact of nat on bittorrent-like p2p systems. *2009 IEEE Ninth International Conference on Peer-to-Peer Computing, Peer-to-Peer Computing, 2009. P2P '09. IEEE Ninth International Conference on*, page 242, 2009.

[38] Bryan Ford, Pyda Srisuresh, and Dan Kegel. Peer-to-peer communication across network address translators. *USENIX Annual Technical Conference*, 2006.

[39] L. DAcunto, M. Meulpolder, R. Rahman, J.A. Pouwelse, and H.J. Sips. How do firewalls and nats affect the performance of p2p swarming systems? In *the 16th Annual Conference of the Advanced School for Computing and Imaging (ASCI'10), Veldhoven, the Netherlands*, pages 1–8. ASCI, 2010.

[40] J. Grimmett and E. ONeill. Upnp: Breaking out of the lan. In *2012 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, pages 170–174, April 2012.

[41] Prakash Iyer and Ulhas Warrier. Internetgatewaydevice:1 device template version 1.01. UPnP Forum2001.

[42] Philipp Richter, Florian Wohlfart, Narseo Vallina-Rodriguez, Mark Allman, Randy Bush, Anja Feldmann, Christian Kreibich, Nicholas Weaver, and Vern Paxson. A multi-perspective analysis of carrier-grade NAT deployment. *CoRR*, abs/1605.05606, 2016.

[43] X. Fu, M. Stiemerling, and H. Schulzrinne. Implications and control of middleboxes in the internet. *IEEE Network*, 22(5):6–7, September 2008.

[44] Andra Lutu, Marcelo Bagnulo, Amogh Dhamdhere, and K. C. Claffy. Nat revelio: Detecting nat444 in the isp. *Passive & Active Measurement (978331930504 2)*, page 149, 2016.

[45] R. Mahy, P. Matthews, and J. Rosenberg. Rfc 5766, traversal using relays around nat (turn), 2010.

[46] et al. P. Srisuresh. Rfc 5128, state of peer-to-peer (p2p) communication across network address translators (nats), 2008.

[47] Available pool of unallocated ipv4 internet addresses now completely emptied. https://www.icann.org/resources/press-material/release-2011-02-03-en. Last visited 2017-03-20.

[48] Stanford L. Levin and Stephen Schmidt. Ipv4 to ipv6: Challenges, solutions, and lessons. *Telecommunications Policy*, 38(11):1059 – 1068, 2014.

[49] Remaining ipv4 addresses to be redistributed to regional internet registries - address redistribution signals that ipv4 is nearing total exhaustion. https://www.icann.org/news/announcement-2-2014-05-20-en. Last visited 2017-03-20.

[50] Cisco systems, inc. http://www.cisco.com. Last visited 2017-03-21.

[51] Cisco. Deploy cgn to retain ipv4 addressing while transitioning to ipv6.

[52] Carolyn Duffy Marsan. Slow move to ipv6 giving nat a new life. (cover story). *Network World*, 25(28):1 – 14, 2008.

[53] J. Beeharry and B. Nowbutsing. Forecasting ipv4 exhaustion and ipv6 migration. In *2016 IEEE International Conference on Emerging Technologies and Innovative Business Practices for the Transformation of Societies (EmergiTech)*, pages 336–340, Aug 2016.

[54] Peter Dell, Christopher Kwong, and Liu Ying. Some reflections on ipv6 adoption in australia. *Info*, 10(3):3 – 9, 2008.

[55] Adiel. A. Akplogan. Ipv6: The future is now more than ever. https://www.icann.org/news/blog/ipv6-the-future-is-now-more-than-ever, September 2015. Last visited 2017-03-21.

[56] Lee Howard. Internet access pricing in a post-ipv4 runout world.

[57] E. Bocchi, A. S. Khatouni, S. Traverso, A. Finamore, V. D. Gennaro, M. Mellia, M. Munafò, and D. Rossi. Impact of carrier-grade nat on web browsing. In *2015 International Wireless Communications and Mobile Computing Conference (IWCMC)*, pages 532–537, Aug 2015.

[58] M. Yoshida and A. Nakao. Measuring bittorrent swarms beyond reach. In *2011 IEEE International Conference on Peer-to-Peer Computing*, pages 220–229, Aug 2011.

[59] L. DAcunto, M. Meulpolder, R. Rahman, J. A. Pouwelse, and H. J. Sips. Modeling and analyzing the effects of firewalls and nats in p2p swarming systems. In *2010 IEEE International Symposium on Parallel Distributed Processing, Workshops and Phd Forum (IPDPSW)*, pages 1–8, April 2010.

[60] Bori András. Drtorrent. https://play.google.com/store/apps/details?id=hu.bute.daai.amorg.drtorrent&hl=sv. Last visited 2017-03-04.

[61] K. Csorba, P. Ekler, A. Bori, and H. Charaf. Analysis of mobile bittorrent client behavior. In *2013 IEEE 4th International Conference on Cognitive Infocommunications (CogInfoCom)*, pages 613–618, Dec 2013.

[62] A. Bori and P. Ekler. The analysis of bittorrent protocol reliability in modern mobile environment. In *2013 3rd Eastern European Regional Conference on the Engineering of Computer Based Systems*, pages 120–126, Aug 2013.

[63] Amazon. Using bittorrent with amazon s3. http://docs.aws.amazon.com/AmazonS3/latest/dev/S3Torrent.html. Last visited 2017-01-19.

[64] Amazon. Working with amazon s3 buckets. http://docs.aws.amazon.com/AmazonS3/latest/dev/UsingBucket.html. Last visited 2017-03-04.

[65] Assocation for Computing Machinery (ACM). Acm cscw 1994 issue 87 - seamless media design. https://open-video.org/details.php?videoid=8246, 2004. Last visited 2017-02-20.

[66] Chris Grundemann. Carrier grade nat - observations and recommendations. In *North American IPv6 Summit*, 2012.