Simulation and data analysis of peer-to-peer traffic for live video streaming

ANTHONY SMITH
ANDREAS JOHANSSON LINDGREN

MASTER´S THESIS
DEPARTMENT OF ELECTRICAL AND INFORMATION TECHNOLOGY
FACULTY OF ENGINEERING | LTH | LUND UNIVERSITY

Simulation and data analysis of peer-to-peer traffic for live video streaming

Anthony Smith, Andreas Johansson Lindgren
`dat12as1@student.lu.se, dic12al1@student.lu.se`

Department of Electrical and Information Technology
Lund University

Supervisors: Maria Kihl, LTH
Jonny Lundell, Entecon AB
Magnus Hoem, Entecon AB

Examiner: Christian Nyberg

November 29, 2017

© 2017
Printed in Sweden
Tryckeriet i E-huset, Lund

Abstract

Evaluating and testing changes or configurations to peer-to-peer systems or even understanding their behaviour can be complicated. One approach is to simulate a large peer-to-peer system and visualise the results. In this master's thesis a study is performed to understand how an actual implementation of a hybrid peer-to-peer live video streaming system behaves and performs under different scenarios.

The behaviour and performance of a hybrid live video streaming system consisting of an unstructured mesh-pull-based P2P network and a classic content delivery network solution is studied by simulating the system with different scenarios such as flash crowds and flash disconnects. The simulation system includes a network model taking latency and bandwidth into consideration.

As expected the mesh-based system performed well under user churn. Although the system consisted of approximately 80% free-riders the utilisation of the content distribution network was reduced by 95% on average. The data analysis was successful in improving the system's overall performance. Furthermore, the visualisations and data analysis were used to understand the system's behaviour.

Keywords - P2P, Peer-to-Peer, Hybrid Live Video Streaming System, Mesh-based-pull Approach, Unstructured Network, Simulation, Visualisation.

Acknowledgements

We are very grateful to Entecon AB, and our supervisors Jonny Lundell and Magnus Hoem, for giving us the opportunity and resources to execute the project. Many thanks to Jonny Lundell providing invaluable guidance, commitment, and many good ideas during this master’s thesis. Even more praise for remaining positive and encouraging throughout the project. Praise and thanks towards Magnus Hoem are in order. Providing invaluable feedback, and helping us maintain our ambition and goals. Lastly, we would like to thank Maria Kihl for being a very flexible and friendly supervisor, and helping us with the academic process and conducting this master’s thesis.

Popular science summary

The behaviour and overall performance of peer-to-peer live video streaming systems are complex, and known to be notoriously hard to mathematically model and theorise. Peer-to-peer live video streaming systems utilise the users' resources to improve the system's scalability, robustness, and to reduce costs. In this master's thesis a peer-to-peer live video streaming system is simulated and tested in various scenarios in order to visualise the results and understand its overall performance.

Peer-to-peer (P2P) protocols such as BitTorrent share content in a more static environment where it is easier to keep track of where to find each part of the content. These parts can be obtained in any order, or at any specific instance. However, in a live environment content has a life span and then becomes obsolete. The order is also vital to avoid interruptions in playback. Users can swap channels creating a dynamic environment, imposing other problems than for the static environment.

There exists several types of P2P live video streaming systems and multiple ways of simulating them. In such streaming systems, users streaming the same media content attempt to help each other obtain it by sharing their resources. In the P2P solution studied in this master's thesis users select what media content they want to stream, and this is noted by a supervising entity. The supervising entity keeps track of which users recently began streaming, and what they are streaming. When users begin to stream the supervising entity provides a list of users streaming the same content. This list of users along with algorithms that for example determine which users to request content from, and when to request content without needlessly flooding the P2P network with requests, is all that is needed to outperform classical streaming systems.

Studies on simulating P2P live streaming systems usually take a mathematical approach and specify strict assumptions and conditions for the simulation. In this master's thesis an actual implementation of P2P solution for live streaming is used, and the simulation attempts to simulate real conditions.

The common denominators for P2P live video streaming systems are their complexity, and their enormous potential to outperform classical live video streaming systems while also reducing costs. Therefore, when developing P2P solutions for live video streaming there is a need to test and evaluate how changes and configurations affect the system. Due to the complexity of P2P solutions it can be challenging to understand how certain changes will improve the system or if they will improve it all, and testing this with real users is both expensive and time consuming.

Contents

List of Figures

List of Tables

List of Acronyms

- **CDN** - Content Distribution Network
- **CRID** - Content Reference Identifier
- **GNP** - Global Network Positioning
- **HLS** - HTTP Live Streaming
- **ISP** - Internet Service Provider
- **NAT** - Network Address Translation
- **OTT** - Over The Top
- **P2P** - Peer-to-Peer
- **QoS** - Quality of Service
- **RTT** - Round Trip Time
- **VoD** - Video on Demand

Chapter 1 Introduction

This master’s thesis will introduce and examine the field of peer-to-peer (P2P) systems for live video streaming on behalf of Voddler Sweden AB in Stockholm, Sweden. The work was performed at Entecon AB, who conduct the development for Voddler. Voddler will be used as an abbreviation throughout the report.

The motivation of the master’s thesis is to study and analyse the behaviour and overall performance of an actual implementation of a P2P live video streaming system, under different scenarios. The study is performed by simulating the system. This includes simulating network resources such as latency and bandwidth limitations. The long term goal for Voddler is to be able to use the simulation system to simulate proposed improvements and see what implications they have.

1.1 Background

One of the absolute largest parts of internet traffic today is video traffic. In 2015 video represented 70% of all traffic on the internet, and this volume is expected to grow to three times as much in 2020, and will at that time represent 83% of all traffic. [1]

Essentially all commercially available models of video distribution are based on unicast, where each viewer has its own direct connection to the back-end, placing enormous pressure on the back-end infrastructure. [2]

The operators, both over the top (OTT) providers like Netflix and operators like Ericsson, are trying to remedy this by putting servers with big discs that act as local caches as close to the customers as possible. However, this is a very expensive solution, especially in areas where population is sparse, or if the area has poor network infrastructure.

Providers of live streams have further problems with spikes in what people watch, and when they watch it. This can for example arise when users of live streaming systems watch popular sporting events, many users will begin streaming close to kickoff and then watch something else in half-time. It is expensive to purchase a lot of bandwidth and server supply to meet the demand for the occasional enormous sporting event, and then just have it unused.

To solve these dilemmas, Voddler developed a hybrid peer-to-peer system that copes well with many users in the same area watching the same streams. It enables material that is being streamed, to be peered (redistributed) to others watching the same stream.

In this thesis a hybrid live video streaming system, consisting of an unstructured P2P network and a classic content delivery network (CDN) solution, is simulated. Material being watched by many peers (users) is peered between them, and less popular material is being distributed directly from the CDN to the end user. The system uses some notable algorithms that have been created by Voddler, e.g when to get data from peers, what to get from whom, and when it is better to get data from the CDN in a live environment etc. These algorithms will be described more in detail later.

1.2 Project aims

The main goal of the thesis is to evaluate and analyse how the system performs under various scenarios through simulation. Thus, this thesis will try to answer the following questions:

1. How will the P2P network change over time in terms of: content distribution in the network, and peers' responsibilities?
2. How will the P2P network cope under different scenarios, flash crowds and bursts of sudden disconnects etc?
3. Can changes and configurations that improve the simulated system be found, and if so, are these alterations likely to improve the actual system?

1.3 Approach

How will a hybrid P2P live video streaming system perform with different scenarios? Can changes and configurations of a hybrid P2P live video streaming system be tested and evaluated by simulation? The thesis' approach is to study an implementation of said system by simulation, with different scenarios, and then visualise and analyse the results.

The first step consists of researching the field and related work.

The second step is to create a simulation system that includes a network model, and create a visualisation system that visualises the results generated in the simulations.

The third step is working with skilled developers, with great knowledge of live video streaming and P2P solutions, to accurately simulate their system.

The last steps are to state hypothesises for expected behaviour, simulate different scenarios, and perform a data analysis of how the system performed, and if configurations and changes can be found to improve its overall performance.

1.4 Limitations

There are limitations to the work conducted in this master's thesis. User behaviour was not studied, nor attempted to be fully replicated in the simulated system. Instead attempts to replicate user behaviour are based on assumptions.

Although the work in this thesis has deployed great system focus, the simulations do not consider media player behaviour, or a peer's bandwidth changing over time.

Hardware limitations are not considered. Resulting in that the simulation does not for example differentiate between an old Android device's or modern computer's RAM or CPU.

1.5 Outline

The report is outlined as follows:

- *Chapter 2 Background:* Explains the theory to understand video streaming, peer-to-peer networks, simulation systems, network modelling, and the simulated system.
- *Chapter 3 Methodology:* Explains the methodology that is being used to create the simulation system and the visualisation system, to perform the data analysis, and how to reach conclusions and recommendations.
- *Chapter 4 Implementation:* Explains how the simulation system and the visualisation system were designed and implemented, and motivation of why.
- *Chapter 5 Results:* Presents the results and visualisations from the simulation.
- *Chapter 6 Data Analysis:* Analyses the results and visualisations from the simulations.
- *Chapter 7 Discussion:* Contains a discussion about the results and data analysis, both in project aims and in general, possible sources of error and future work.
- *Chapter 8 Conclusions:* Presents the conclusions of this master's thesis project.

Chapter 2 Background

This chapter will describe concepts relevant to the thesis, related work and the live streaming system that was simulated.

2.1 Video streaming

During the early 2000s the Internet saw a booming escalation of network bandwidth. Combined with improved media compression algorithms and more powerful computer systems streaming delivery of media content became possible. The term *streaming*, the only alternative for a live scenario, refers to a method of relaying data over a computer network as a stable continuous stream, granting playback of obtained content while consecutive data is being received. This differs from *download-and-then-play*, where playback starts when all the data has been downloaded. Instant playback is the main advantage of streaming delivery as the user is no longer required to wait until the entire download is completed. [3]

2.1.1 On-demand vs live streaming

There are two cases for streaming media, on-demand streaming and live streaming. In the first case, the media has been recorded and compressed in advance. The media content is stored on a server and delivered to receivers upon request (on-demand). A great number of sites provide streaming of stored media content today, Youtube [4] being one example. However, in live streaming, the media is captured, compressed and transmitted on the fly. Live streaming is more complex in terms of requiring significant amount of computing capabilities and usually specific hardware support on the back-end. In our thesis we have solely focused on simulating live streaming of media content. [3]

2.1.2 Apple's HTTP live streaming

There are several media streaming solutions in the marketplace today. The simulated system operates the most widespread standard, Apple's HTTP live streaming (HLS) solution. HLS follows approximately the same principle as other systems, such as DASH and Smooth Streaming.

This subsection will introduce the architecture of HLS, how the technology functions, supported formats, server requirements, and what clients are available. This section borrows greatly from Apple's documentation. [5]

HTTP live streaming architecture

Theoretically, HLS can be seen as three components: the *server* component, the *distribution* component, and the *client software*.

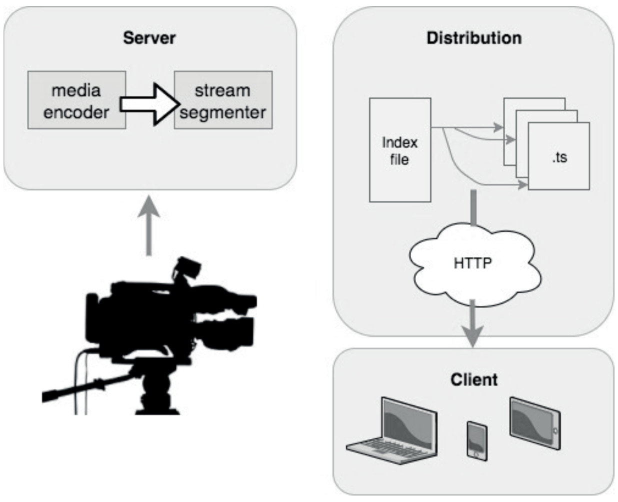

Figure 2.1: The HTTP Live Streaming Architecture

- *The server component* handles input streams of media and encodes them digitally, encapsulates them in suitable format for delivery, and prepares the media for distribution.
- *The distribution component* consists of standard web servers. They handle accepting client requests and delivering the prepared media and dedicated resources to the client.
- *The client software* determines the appropriate media to request, downloads those resources, and then reassembles them so that the media can be played back to the user in a continuous stream.

In a typical setup H.264 video and AAC audio, which is packaged in a MPEG-2 Transport Stream or a FMP4 container, is divided into short media files by a software stream segmenter. These files are placed on a web server.

An index file containing a list of the media files is generated and maintained by the segmenter. The index file's URL is published on the web server. Client software reads the index, then requests the media files that are listed in order and displays them without interruptions between segments.

An example of a simple HLS setup can be seen in Figure 2.1. The MPEG-2 stream is divided by a software stream segmenter and stored as a series of media files.

The index file is created by the segmenter and contains a list of media files and metadata, the index file is in *.M3U8* format. After the URL of the index file is accessed by clients, the indexed media files are requested in sequence. [5]

2.1.3 Using HTTP live streaming

In the case of a live session of streaming media, newly created media files are made available to the users by updating the index file. The index file, which can be seen as a sliding window, is the basis for the continuous stream. In Figure 2.2 a graphical representation of this sliding window phenom is depicted. The index file always contains the x most recent segments, this is achieved by removing the oldest segment when adding a new segment. [5]

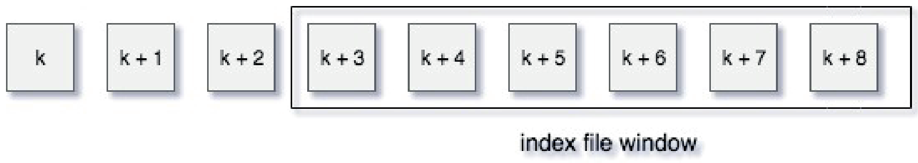

Figure 2.2: Index file of a live session, updated for every new segment.

Stream alternatives

Index files usually reference multiple streams of the same media content with varying levels of quality for different bandwidths or devices. The client software uses heuristic methods for determining what quality to stream and when to switch between the alternates. Currently the heuristic methods are based on the recent trends measured in the network throughput. [5]

2.2 Content delivery networks

Content delivery networks (CDNs) are large distributed systems of servers, which provide a geographically distributed caching infrastructure. To meet the growing demand of content, CDN providers offer an efficient and common method to provide content to end-users. *The mirror servers* copy and cache the content of *the main server*. Mirror servers are placed close to the customers to provide high quality service and convenient access. [6]

Akamai and Amazon Cloudfront are examples of CDNs.

2.3 Network address translation

Users that are part of the Internet behind network address translators (NATs) are commonly encountered in large amounts, typically around 70-80%. The presence of NATs cripples the functioning of P2P systems as they block direct communication between NATed users. This results in that NATed users, can't contribute with its resources to the system. Although there was an attempt to standardise NAT, especially to improve the support for P2P applications, not all vendors have complied to the standard. [7]

Developing P2P applications utilizing the NAT has constraints, due to that that two or more computing systems, in different private networks, are unable to send messages between them without a public address. Protocols such as NAT-PMP [8], PCP [9] and UPnP [10] employ communication protocols to configure the gateway and create a port-forwarding without the obligation of user configuration. [11]

2.4 P2P networks

Content publishing and distribution is often conducted in a costly and inefficient manner through Client/Server models. Client/Server models provide negative network externalities in that each user causes more costs by increasingly congesting the system due to consumption of scarce resources. Let us define the model as a distributed network where the *server* is the central registering unit as well as the only provider of content and service. A *client* only requests content or the execution of services, without sharing any of its own resources. [12]

During the the 1990s, the Internet consisted mainly of client/server models. In recent years, a considerable amount of new techniques have surfaced that enticed customers into desiring more than purely text and images. Several aspects such as the widespread boom of broadband Internet, improved connection reliability, better compression technology, more storage capability, more CPU power and the fact that a great amount of content resides on end users' personal computers changed how Internet was consumed. All these aspects make P2P networks, where users can share their resources, a promising solution for various areas. [12]

2.4.1 P2P network types

A P2P system is *unstructured* when users and data are positioned without certain rules and in an ad hoc manner in the network. Most unstructured systems are characterised by heavy consumption of bandwidth in terms of message traffic. However, this type of system is mostly very resilient. [12]

In *structured* networks users and data are being placed in a structured way throughout the network according to specific rules, this increases the localisation of data and the scalability. [12]

Many live streaming systems operate P2P solutions to provide users with Internet-delivered real-time multimedia content, CoolStreaming [13] and PPLive [14] being examples. These systems are constructed in an unstructured manner with information exchanges occurring in an epidemic-style, and users deciding who to interact with in an interactive manner. Unstructured systems are thought to be winning against the structured counterparts due to their operational simplicity and relative robustness to changing user populations. Structured systems require significant overhead when re-configuring their local topology due to user arrivals or departures. [15]

2.4.2 Overlay construction

P2P systems usually operate single tree, multi-tree, or mesh topology for live video streaming.

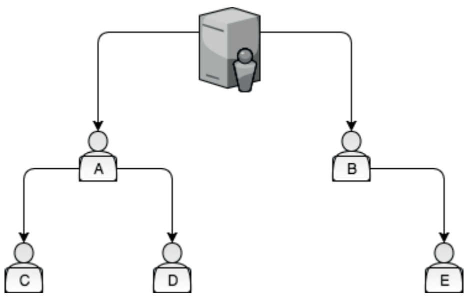

Figure 2.3: Single tree topology.

Single tree topology is a structured topology. Users participating in a live streaming session form a tree structure at the application layer, where the video source server acts as a root of the tree. Each peer receives the streamed content from its parent peer located one level above itself, and forwards the received content to its children peers located one level below. This topology focuses on minimising the depth of the tree by increasing the number of peers at each level. The reason for this being to minimise the number of hops taken by the chunks, thus reducing video delay, especially at the lower levels. Although tree topology provides a good structure for live streaming, it comes with drawbacks. Problems occur when a peers leaves the streaming session, namely its children peers and descendant peers will also leave. [16]

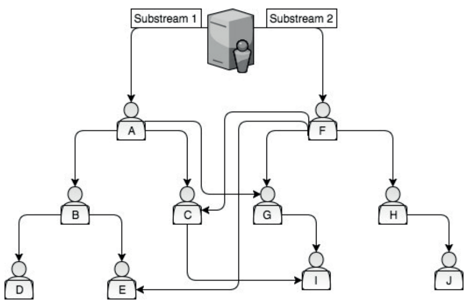

Figure 2.4: Multi-tree topology.

Multi-tree is an unstructured topology, resulting in that this type of topology consists of more than one sub-tree as opposed to one streaming tree. The server divides streaming the content into multiple sub-streams, where each sub-stream provides one of the sub-trees. Multi-tree topology intends to counteract the passive leaves that occur in single tree topology, this is achieved by a leaf in one sub-tree being a middle peer in another sub-tree. Multi-tree topology also solves the problems that occur when a peer leaves, due to its children receiving from peers in another sub-tree. However, a drawback is increasing overhead of streaming compared to single tree, and when a peers becomes a leaf in all sub-trees. [16]

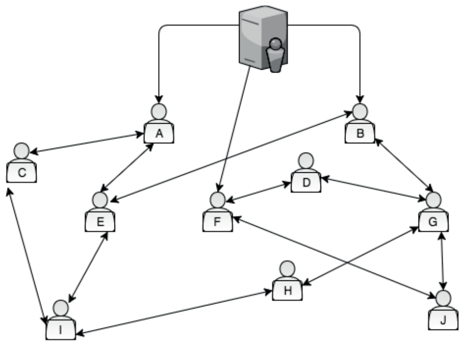

Figure 2.5: Mesh topology.

Mesh topology is an unstructured topology. In mesh topology peers download from and upload to multiple peers. Mesh topology provides robustness against peers who often change between on and off state. Although most P2P systems with mesh topology form neighbourhoods randomly, other systems implement selection rules aiming to improve the network's structure. These selection rules can be comprised of the peers' functions, resource availability, bandwidth, CPU and memory usage. Peers in mesh topology are not limited to making connections when neighbouring peers leave, they change neighbours optionally to achieve improved performance. [16]

2.4.3 Push/Pull-based P2P

Push-based P2P refers to each peer immediately forwarding all incoming data blocks to its neighbouring peers (*pushing*). Such a solution results in significant overhead, as a peer may receive multiples of the same data block by receiving it from different peers. [17]

Pull-based P2P on the other hand refers to peers requesting missing data blocks explicitly. In this scheme peers have to initiate the transmission of data blocks towards themselves (*pulling*). Although a pull-based approach prevents receiving duplicates, it comes at a cost of latencies as notifications and requests are sent back and forth. Thus, there is a trade-off between efficiency and overhead. [17]

Peers that form a mesh-shaped overlay, pushing or pulling content from each other, are either mesh-*push* or mesh-*pull* approaches. [18]

2.5 Network modelling

In order to make the simulation more accurate network limitations were taken into consideration.

2.5.1 Bandwidth

When evaluating P2P live streaming systems by simulation, it is vital to correctly model how the underlying network manages the bandwidth capacity if peers distribute large amounts of content between each other. [7]

In the simulation system in this thesis, peers are assigned download and upload capacities. When a peers downloads content, or uploads content, a share of the capacity is allocated. If a peer attempting to download content has insufficient remaining download capacity, or a peer attempting to upload content has insufficient remaining upload capacity, the content distribution request fails.

2.5.2 Internet service providers

P2P systems are becoming increasingly popular and contribute with a great amount of overall network traffic. [19] However, these systems have put Internet Service Providers (ISPs) in a dilemma. Although applications running with P2P systems have increased revenue for ISPs, P2P traffic is extremely challenging from a traffic engineering point of view. This is due to most P2P systems depending on application layer routing based on an overlay topology upon the Internet, thus depending on Internet routing. Inefficient downloads can occur if neighbours are chosen poorly by P2P traffic crossing boundaries several times. [20]

To simulate latency occurred by crossing ISP boundaries, peers are assigned to different ISPs with additional latencies if their boundaries are crossed.

2.5.3 Packet Loss

User access links are assumed as the bottlenecks in P2P networks. Many peers are home users connected with e.g. a cable modem to the Internet. Thus, a majority of the available upload bandwidths are unused resulting in low pack loss rates due to congestion. [21]

To simulate packet loss, probabilities were used from a mesh-based P2P packet loss estimation. [22]

2.5.4 Latency

Internet latency can be modelled and abstracted in various levels of detail by using: *statistical models*, *Global Network Positioning (GNP)*, or *network simulators*. [23]

Statistical models

Statistical models compress the behaviour of the Internet into a formula. Advantages are minimal computational effort, simplicity, and virtually no memory costs.

Statistical models are found to satisfactorily approximate end-to-end round trip times (RTT) with constant plus gamma distributions, given that the time interval when estimating the distribution is kept short. Statistical models are unsuitable if the tested system depends on (region dependent) jitter, or locality. [23]

Global network positioning

Global Network Positioning (GNP) [24] computes delay by using distance in some Euclidean space. *OverSim* [25] is an overlay simulation framework featuring a network model that uses geographical distance between two locations to estimate network delay. [23]

> "For this, each node is placed into a two-dimensional Euclidean space. In addition the peer is assigned to a logical access network characterized by bandwidth b_n, access delay d_n and packet loss, so that heterogeneous access networks can be simulated. The end-to-end packet delay d_e of packet P with length l_P between overlay nodes A and B is then calculated as follows where c is constant."

$$d_e = d_1 + \frac{l_P}{b_1} + c \cdot \|A - B\|_2 + d_2 + \frac{l_P}{b_2} \tag{1}$$

In this thesis peers were placed randomly into a two-dimensional Euclidean space, and equation 1 was used to determine the RTT between peers, and also considering jitter and ISP crossing latencies.

Network simulators

While statistical models mimic some statistical aspects of the Internet, network simulators model the essentials of the Internet, including backbone networks and routing protocols. A detailed model simulates behaviour of the real Internet better, however this requires considerable simulation effort and many optimisations. [23]

2.5.5 Jitter

Studies of network jitter, the variation in the delay of received packets, in large scale P2P systems have assumed *i)* the majority of peers in large scale P2P systems are comprised of computers connected to the Internet via cable modem. *ii)* The access links are bottlenecks in these end-to-end Internet connections. *iii)* The Internet congestion caused by a single user is negligible. These assumptions lead to dividing delay from end-to-end connections into two characteristic classes: [23]

1. If there is little load on the access links, the delay is characterized by the sum of the deterministic upload transmission delay, the stochastic Internet delay, and the deterministic downstream transmission delay on the receiving side.
2. If there is network load on the up-link, however, significant additional delay is caused by queuing before the up-link and the end-to-end delay is much larger.

Work on modelling jitter based on geographical location for P2P systems has observed measurement data to conclude that the inter quartile range generated by a log-normal distribution usually has values between 0-20 *ms*. [26]

This thesis simulates network jitter by using random values between 0-20 *ms*.

2.6 Peering grade

The *peering grade* is a measure of how much of the data that are being transmitted from the CDN to the peers and how much is actually transmitted between the peers. It is defined as:

$$\text{peering grade} = 1 - \frac{\text{total data downloaded from CDN}}{\text{total data in network}} \tag{2}$$

2.7 Related Work

When researching the area of mesh-based P2P systems I. Chatzidrossos' Doctoral Thesis [27] has served as a constant source of inspiration.

The following subsections mention the related work that has inspired this thesis the most.

2.7.1 On Reducing Mesh Delay for Peer-to-Peer Live Streaming

This subsection address related work that focused on reducing mesh delay for P2P live streaming. This was then simulated and evaluated. [28]

Summary

In this paper the *minimum delay problem* was formulated and shown to be NP-hard. Then a centralised heuristic based on complete knowledge was proposed, to serve as benchmark and optimal solution for all schemes under comparison. The heuristic is built around a concept called *power* (in network), given by the ratio of throughput and delay. The network achieves low delay by maximising the network power. The paper discusses a simple distributed algorithm where peers select their parents based on the power concept. Simulation results show that the algorithm performs well compared to the centralised heuristic, and outperforms traditional and modern solutions.

Mesh Delay: Simulation Setup

Software was used to generate ten different two levels top-down hierarchical topologies. Each topology consists of eight autonomous systems which has 625 routers each. peers are attached randomly to the routers. Evaluation metrics used in the analysis:

1) *Delay*: Time taken for data to travel from the streaming server to the peers measured. Both average and maximum source of delay are measured.

2) *Hop Count*: Refers to number of intermediate peers involved on the overlay path from the source to a peer. Provides an idea of the depth of the overlay.

3) *Source Workload*: Workload is defined as the amount of bandwidth that the source uses to upload the peer that directly connect to it.

2.7.2 PULSE: An Adaptive, Incentive-Based, Unstructured P2P Live Streaming System

This section address the related work *PULSE*, which has been evaluated under realistic conditions via simulation and emulation. [29]

Summary

PULSE is an unstructured mesh-based P2P system designed to support live streaming of media content on a large scale, under the arbitrary resource availability usually caused by the Internet. PULSE is a highly dynamic system, constantly optimising its mesh of data connections using feedback based peer selection rules. The paper evaluates PULSE under realistic conditions via simulation and emulation and presents the advantages of their approach, being best-effort response to resource limitations in the system and high resilience to peer churn.

PULSE: System Overview

This subsection presents the main parts of PULSE, basic terms and concepts, and details of structure and its components.

1) PULSE operates in P2P fashion. All peers are identical except the source, which differs as it is the first peer to distribute the original content. Peers can freely exchange short messages among themselves regarding the average stream reception performance. Along with this information and current measurements, peers temporarily connect to exchange data.

2) PULSE is mesh-based. Peers can easily change position and rearrange themselves based on membership changes and bandwidth capacities. This, likewise Voddler's solution, provides more flexibility.

3) PULSE is unstructured, making design and analysis more straightforward.

4) PULSE is incentive-based, incentives are used as local polices serving as global optimisation. The paper stresses that these incentives have not been introduced to achieve fairness in the system, but to optimise the overall system performance.

PULSE: Simulation Setup

PULSE was evaluated under different network sizes, access bandwidth distributions, buffer parameters, and peer arrival patterns. Mainly worst-case bandwidth availability and heterogeneity scenarios were focused, which provided meaningful insights.

The *PULSE* simulations were conducted using a simple round-based simulator. The simulated network has a single-stub topology. Peers with configurable bandwidth were connected to the stub through access links. Bandwidth allocation was performed using a slot-based mechanism. Network latencies were not taken into account.

All scenarios had the same set-up in terms of upload bandwidth, size of sliding window, and trading window.

2.7.3 LayerP2P: Using Layered Video Chunks in P2P Live Streaming

This subsection address the related work *LayerP2P*, which has been prototyped, deployed and validated. [30]

Summary

LayerP2P is a P2P live streaming system that is designed to improve three bandwidth aspects.

1) Users should contribute with bandwidth resources.

2) Provide adaptation to aggregate bandwidth availability.

3) Countermeasure bad video quality when bandwidth availability falls below bandwidth supply.

LayerP2P was implemented in C++, prototypes were deployed in PLanetLab, and performed extensive. A wide range of scenarios were examined with test-driven simulations. The results showed that the P2P system had high efficiency, provided differentiated service, adapted to different bandwidth scenarios, and eliminated free-riders.

LayerP2P combines layered video, mesh P2P distribution, and a tit-for-tat-like algorithm (where more upload contribution receives better video quality).

LayerP2P: System Overview

This section provides an overview of the design of LayerP2P.

1) LayerP2P uses *layered video*, as opposed to single-layer video which we are simulating, on chunk-based mesh P2P live streaming systems. Much like Voddler's solution, in LayerP2P a peer joining the system receives a peer-list from which it can form neighbour relationships. In LayerP2P each peers aim to maximize its upload in order to receive better video quality.

2) *Neighbour management* is conducted by each peer classifying its neighbours into imitators and receptors to treat them differently (sending different initiation messages).

3) *Tit-For-Tat with layered video* providing user incentives to maximise upload in order to receive better video quality, inspired by BitTorrent's incentive.

4) *Receivers side scheduler*, allows receiving peers to determine how to request missing data in order to maximise its received video quality.

LayerP2P: Simulation Setup

An implementation of LayerP2P following the description above was created. The LayerP2P engine, the core software peace, was implemented in C++. The engine obtains peer lists, chooses neighbours, and exchanges media content between neighbours.

The LayerP2P philosophy was validated by a large-scale experiment conducted in PlanetLab. The validation included different types of peers: 1) institutional peers; 2) residential peers; and 3) free-riders. Two scenarios were considered, an underloaded system without presence of free-riders, and an overloaded system with presence of free-riders. Flash crows were included in the simulation, peers connected during a short time interval and stayed in the system for a considerable amount of time.

2.8 Description of Voddler's P2P solution

This work simulated Voddler's system, which is a mesh-pull-based P2P solution for live video streaming and video on demand(VoD). As the scope does not consider VoD, multi bitrate, or several available live streams, this is neglected in the description of the system.

Here follow definitions for terms used throughout the description:
i) peering - distribution of content between peers
ii) stratum - value indicating a peer's contribution to the P2P network.

Voddler's solution can be quantified into three generalised components:

1. **Initialisation of a peer starting the live stream**: a peer begins streaming and finding connected peers that are watching the same content reference identifier (CRID).
2. **Selection rules**: the peers determine which neighbours to request content from.
3. **Maintaining the P2P network**: the peers regularly evaluate their contribution to self-regulate and improve the entire P2P network.

2.8.1 Initialisation of a peer starting the live stream

Peer initialisation

When a peer behind a firewall is added to the P2P network it receives *stratum* 100, while peers not behind firewalls randomly receive a *stratum* value between 20-80. These *stratum* values should ideally converge relative to the peers' download and upload resources.

Finding peers

When a peer starts a live stream, it attempts to find peers watching the same CRID. This is achieved by communicating with an entity called *Buddy Server*.

The *Buddy Server* responds with a list, referred to as *recent-list*, of peers it has been in contact with recently to the peer it is serving, it then adds the served peer to this list. The *Buddy Server* can not guarantee that the peers in its *recent-list* are available, however it is probably worth attempting to communicate with them.

The *Buddy Server's recent-list* will at some point in the beginning of the stream be empty, or contain a limited number of peers. To control how frequently each peer can communicate with the *Buddy Server* a grace period exists, avoiding the peers first connecting to the stream to spam the *Buddy Server* as they attempt to discover more peers.

When a peer receives the *recent-list*, it asks all these peers if they know any peers. The peers that respond, now known to be connected and active, answer with the peers they usually communicate with, and that also probably are connected. This leads to clusters pollinating each other with peers, resulting in that peers gain knowledge of the P2P network quickly.

2.8.2 Selection rules

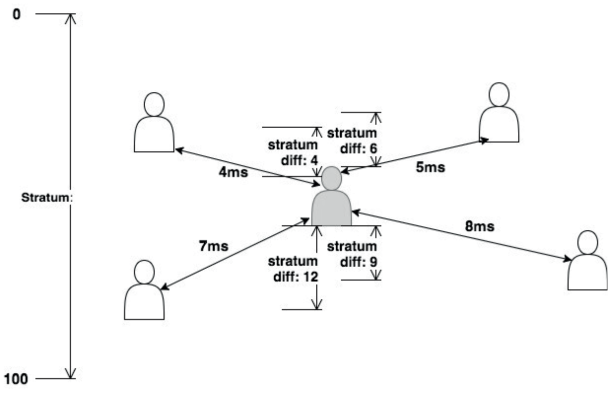

Figure 2.6: The peer in grey, and its neighbours in white. Considers latency and stratum difference when determining its best candidates.

There is no centralised process that manages the system. Instead each peer in the network has a value, referred to as *stratum*, that can be viewed as a measurement of how powerful the peer is in terms of downloading and peering the content, it is an indication of when the peer will have obtained content. Peers with low *stratum* will obtain content earlier than the rest of the P2P network, and be able to peer content better. The peers with the lowest *stratum* have the desired resources to download content from the CDN and upload it to its neighbours. Peers with high *stratum* should obtain content later than the rest of the P2P network, and probably have lesser resources to peer content to its neighbours.

Each peer calculates a *closeness* value for all its neighbours, see Figure 2.6. The *closeness* value is based on the peers difference in stratum and how close the peers are to each other in terms of latency.

Each peer then sorts its neighbours in a list referred to as *prospect-list*, based on these *closeness* values to determine which neighbours (prospects) it should request content from. This should generate a self-regulated P2P network with very efficient content flow from the CDN to the least powerful peers, as the system aims to minimise content requests from the CDN.

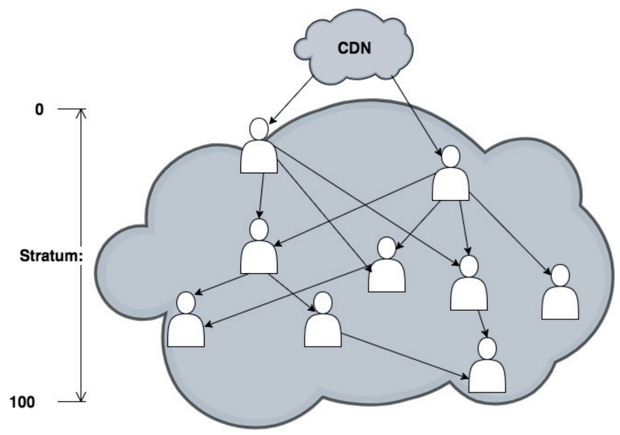

Figure 2.7: Content distribution flow from the CDN to the peer.

2.8.3 Maintaining the P2P Network

Each peer reevaluates its *stratum* periodically according to an algorithm to determine if it should decrease, increase, or remain the same. *Stratum* can also be viewed as a contract of how much content the peer should be capable of downloading and peering. The algorithm calculating the *stratum* value is based on: content requests from other peers, and how successfully the peer downloaded and peered content relative to its current contract. The algorithm considers content statistics based on samples from the last 15 minutes.

This creates an ordered network where the peers are ordered by their stratum and always request content from neighbouring peers with stratum lower than their own. This should ideally form the network in a pyramid structure where the layers will receive the content sequentially, and peers with good resources will climb further up the pyramid.

2.8.4 Network structure

One of the proposed solution's key aspects, is that every peer calculates a time to start peering content, depending on the peer's position (*stratum*), viewed as a distance from the CDN. This introduces a set latency in the network and gives the network time to distribute the data in a more controlled way. Meaning that for each time slot in the beginning of a published segment, peers with lower *stratum* are contributing their upload resources to the peers with higher *stratum* because they obtain content first. This leads to the assumption that a pyramid like structure is preferred due to 80% of all peers are behind a firewall/NAT and can not contribute, i.e. the majority of the network are free-riders. As these peers can't provide upload resources, they receive the worst *stratum* and are placed in the lowest level in the pyramid. Because the network is not trying to be an optimal, or pure P2P solution, this should lower, but not eliminate, the need for peers to request content from the CDN. To mitigate the latency and stress on the peers that can provide upload resources, from the CDN there exists a fallback mechanism that lets them collect the missing data of a segment that is stated to be obtained at a certain time in their contract. This should also minimise the propagation of latency from one layer to the next. This in turn leads to the system not being best effort in terms of distributing content as live as possible.

The users of the system will however not experience the latency difference between one another even if they have different upload capacity because the system then regulates the time that they say is live. If one peer is closer to the CDN in the network it will buffer the segment until the whole system has the content and then deliver it to the media player.

All peers in the network will watch the stream at the same time, due to the media player requesting content with a set delay. Resulting in that content does not need to be obtained strictly in sequential order, and if any content is missing a certain panic time before the player will request the content, the missing chunks are downloaded from the CDN.

Chapter 3 Methodology

This chapter will describe the methodology of how the work was conducted and executed. This chapter will also describe the different scenarios that were simulated and how the data was visualised, in order to perform the data analysis.

3.1 Approach

The aim of the thesis was to study a real implementation of a P2P system for live video streaming, and analyse its overall performance when simulating various scenarios. This was achieved by creating a simulation system, and a visualisation system to visualise and analyse the results from the simulation. The thesis consisted of four phases.

The first phase consisted of designing and implementing a simulation system in Golang [31] that simulated the P2P system. The data produced in the simulation was recorded and parsed into JSON [32] format by using Python [33] scripts.

We worked with an agile approach, implementing in three week sprints. Every third week we reconciled with Voddler and evaluated our work in the three week span, and lastly, decided what to include in the following three week sprint. We also designed scenarios that the simulation should cover, these are listed in section 3.3.

The second phase consisted of visualizing the recorded data. This was achieved by implementing a visualisation system using a JavaScript library called D3 [34].

The third phase consisted of verifying that the P2P system was simulated correctly. This was achieved by performing code reviews, explaining our solution to Voddler's senior developers who had designed and implemented the P2P solution, and by performing extensive testing to validate the simulated system.

The forth phase consisted of interpreting and performing a data analysis of the visualisations and results.

3.2 Simulation & Visualisation

This section will describe what the thesis aimed to include in the simulation and the visualisation.

3.2.1 Requirements

The first phase of the thesis included that some common requirements were gathered to ensure that the time was managed efficiently and the scope of the thesis was as expected. The requirements we decided on are:

1. The simulation system should include a network model that takes bandwidth and latency into consideration.
2. It should be possible to simulate the system with different scenarios.
3. It should be possible to alter and configure key components of the different algorithms of the system.
4. The visualisation system should visualise the simulation in different aspects.

3.2.2 Proposed implementation

The simulation system shall simulate the system in great detail allowing for the different algorithms to be altered and configured. The resulting data from the alterations can then be visualised in order to perform a data analysis. This simulation system shall be implemented in Golang [31].

The network model of the simulation system shall take bandwidth and latency implications into consideration to improve the accuracy of the results compared to the real world. Key components of the network model should be configurable.

The aim of *the visualisation system* is to visualise how content flows through the network, how the peers change responsiblities over time and what knowledge peers achieve of other peers in the P2P network, how the peering grade changes over time etc. This tool shall be implemented in JavaScript, using a library called D3 [34].

3.3 Simulation scenarios

This section defines scenarios that were deemed of great importance to simulate in the thesis, in order to evaluate the system with these scenarios.

Throughout the thesis these scenarios will be referred to as "scenario *3.3.X*", with the *X* being the number in the list below:

1. Optimal scenario: peers do not disconnect from the stream.
2. Default scenario: peers disconnect and reconnect to the stream (zapping).
3. Flash crowd: many peers connect in a short time interval.
4. Flash disconnects: many peers disconnect in a short time interval.

Chapter 4 Design & Implementation

This chapter will describe the design and implementation of the simulation system and the visualisation system used in this thesis.

4.1 Design of the simulation system

4.1.1 Considerations

As the simulation system will simulate peers in a P2P system, it must be designed in such a way that thousands of small processes can run simultaneously. Therefore, the simulation system will be implemented in Golang. The programming language is known for its modern concurrency features.

The simulation system will simulate an extensive live video streaming system, and it is to extensive to attempt simulating the entire system in detail. Therefore, the system’s multi bitrate and on demand streaming features will be omitted from the thesis scope.

4.1.2 Design

The design approach is to model each key component from the P2P solution as realistically as the time frame for the thesis allows.

The CDN is designed to have its key functions: publish new content and answer peers’ content requests. To establish that the peers will communicate with the CDN in a realistic fashion, an API will be created that specifies how the peers and the CDN should communicate with each other.

The BuddyServer is designed to have its key functions: serving peers communicating with it, and providing the peers with a list of peers it has served recently.

The peers are designed to in great detail behave as peers in the real P2P system, all key algorithms for the live streaming part will be modelled. The peers are designed to run independently and concurrently.

The NetworkModel is designed in such a way that bandwidth and latency realistically simulate network implications. The NetworkModel shall be configurable allowing for the simulation to consist of peers with different download/upload bandwidths, latencies and transmission rates.

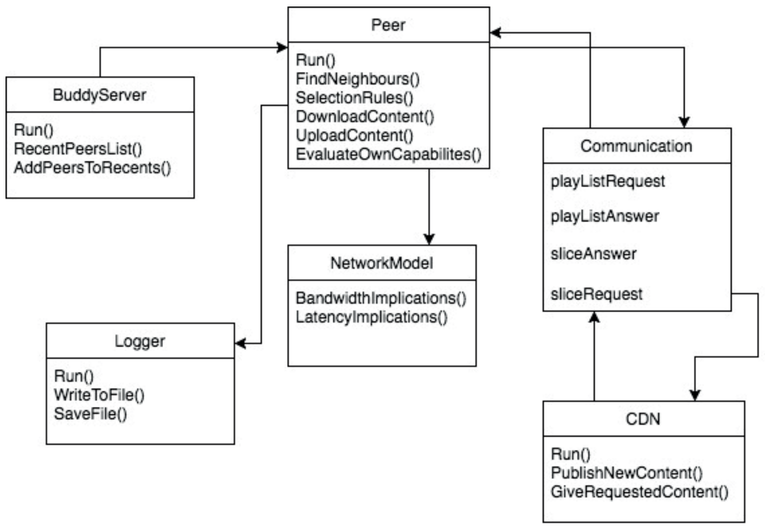

Figure 4.1: Overview of the simulation system's key components.

The Logger was designed to record and save various information of the simulation and related statistics so the simulation results could be analysed.

4.2 Design of the visualisation system

4.2.1 Considerations

Because there will be so much data and events happening all the time in the simulation it is hard to try and single out what exactly happened in graphs using average and clustering. Therefore the Visualisation could be seen as a complement to the graphs. The graphs are used to find timeslots in the simulation that are interesting where something happened. The visualisations are used to playback the events for each peer at a given time to try and understand the system's behaviour.

The logging occurs with nanosecond precision that will lead to big gaps between some events and make the graphs skew. Therefore some data for the graphs are sampled each second instead to make the data more readable.

4.2.2 Design

The *hit-per-request ratio* is the ratio between how many of the peers' requests that result in a transmission of data between peers. The purpose of this measurement is that if the peers and the network are correctly balanced almost all requests will result in a transmission because all the peers will fulfil their contract.

The *triangle* is one of the visualisations that changes over time. It replicates the topology and shows how each peer is exchanging data with other peers. This is used to understand what type of peers take on different upload responsibilities.

The changes made to the topology is in relation to what data segment is being sent.

Network topology is a visualisation that depicts the topology in a more event driven fashion that shows how the topology reconfigured itself during the stream.

4.3 Implementation of the simulation system

Golang

Golang is a free and open source programming language created in 2007 as an experiment by Google, with the aim of removing undesired traits of the most popular languages and keeping the positives. Golang is statically typed, has garbage collection, memory safety features and CSP-style concurrent programming features. Golang is suitable when many small processes run simultaneously, say thousands of peers communicating to download and upload content between each other for instance. [31]

Implementation

The simulation system was implemented in Golang version 1.8. The objective was to simulate many peers simultaneously, preferably around 10 000 peers. This requires efficient concurrent programming and data structures. In order to make it possible for a large amount of peers we utilised Golang's concurrency constructs *goroutines* and *channels*.

Golang's primary concurrency construct is the goroutine which is a type of light-weight process. A function call prefixed with the *go* keyword starts a function in a new goroutine. Goroutines are multiplexed into a smaller set of operating system threads. [35]

While Golang's standard library package features most of the classical concurrency control structures (mutex locks, etc.), idiomatic concurrent programs prefer *channels* which send messages between goroutines. The special syntax "$\leftarrow$ *channel* :" forces the executing goroutine to block until it receives on the channel *chan*.

```
select {
    case contentRequest := <- channel:
         // receive on channel, do something with input
        processContentRequest(contentRequest)
    }
```

4.4 Implementation of the visualisation system

The visualisation system consists of a web page programmed in JavaScript that displays graphs and visualisations created with the JavaScript library D3.js. [34]

D3

D3.js is a JavaScript library that allows for visualisation of documents based on data. D3 uses HTML, SVG, and CSS to visualise data.

Chapter 5
Results

In this chapter, the results are presented by means of graphs, and various visualisations.

5.1 Simulation setup

The simulations were visualised in order to fully understand the results and the system's behaviour. The results and visualisations presented in this chapter are based on simulations generated with the *Network configurations* and the *Peer configurations* in Table 5.1 and 5.2 respectively.

It was decided to simulate the different scenarios with 100, and 1000 peers to understand how the system performed with different P2P network sizes. It is also easier to analyse and study the behaviour of the system when simulating fewer peers. As multi bitrate and several streams are neglected in the scope of this work, it was concluded that running the different scenarios for ten minutes would be sufficient to analyse the results.

Network configurations	
streaming bitrate	4000 kbit/s
segment size	~ 3000 – 3300 KB
chunk size	128 KB
NAT probability	80%
packet loss probability	rand(0.005 - 0.1 %)
number of ISPs	3
latency for crossing ISP	25 ms (one-way)
jitter	rand(0-20 ms)

Table 5.1: The network configurations used in the simulations.

Peer configurations	
Ethernet 100 mbit/s quantity	8%
download rates d_r	30 + rand (0-15) [mbit/s]
upload rates	assigned $d_r \cdot 0.80$
minimum network specific latency	2 ms
LTE 12 mbit/s quantity	2%
download rates (d_r)	8 + rand(0-4) [mbit/s]
upload rates	assigned $d_r \cdot 0.40$
minimum network specific latency	50 ms
CUSTOM_SLOW quantity	30%
download rates (d_r)	8 + rand(0-4) [mbit/s]
upload rates	assigned $d_r \cdot 0.33$
minimum network specific latency	10 ms
CUSTOM_MEDIUM quantity	30%
download rates (d_r)	15 + rand(0-7.5) [mbit/s]
upload rates	assigned $d_r \cdot 0.45$
minimum network specific latency	6 ms
CUSTOM_FAST quantity	30%
download rates (d_r)	25 + rand(0-12.5) [mbit/s]
upload rates	assigned $d_r \cdot 0.60$
minimum network specific latency	3 ms

Table 5.2: The peer configurations used in the simulations.

5.2 Hypothesises

5.2.1 Optimal scenario hypothesis

The hypothesis for the optimal scenario is that the lack of peers disconnecting from the stream will provide desirable P2P circumstances for good peering grade. As peers with different *stratum* have different responsibilities, they will differ in when obtaining content. Therefore, we expect the P2P network to form a pyramid structure with 4-5 layers. The reason that a pyramid is expected, and not a square, is that the peers with low *stratum* will obtain content from the CDN, and then the content will be peered between peers (lower *stratum* providing data to higher *stratum*). Ideally, few peers should obtain content from the CDN, and the layers below will consist of an increasing numbers of peers, relative to the previous layer. As the peers can obtain *stratum* values 0-100, and they primarily attempt requesting content from peers in a range of 20 *stratum* less than themselves, layers are expected to occur in 0-5, 20-25, 40-45, 60-65, 80-85, and the NATed peers at 100, in *stratum*. The P2P network will not immediately take this form, it will occur over time and as there are no disconnects it will probably occur quickly. As the peers reevaluate their *stratum* and can change their stratum ± 8 every five seconds (being randomly assigned between 20-80), we expect the network to stabilise after approximately one minute. The system is expected to have a good hit-per-request ratio when requesting content in this scenario.

5.2.2 Default scenario hypothesis

The hypothesis for the default scenario is that it will behave similarly as stated in the optimal hypothesis. However, in the default scenario peers will randomly disconnect, with increasing probability towards the end of the stream, and randomly reconnect. Therefore, it is expected to have a slightly lower peering grade than in the optimal scenario, especially in the end of the stream. It is also expected to form a fairly stable pyramid structure, and that more peers are required to obtain content from the CDN. As the peers constantly gather knowledge of the P2P network, such as connected/disconnected status and ping times of the other peers by pinging them and assuming that peers are disconnected when not receiving a response within three seconds, disconnects are expected to not affect the results too severely.

5.2.3 Flash crowd scenario hypothesis

The hypothesis for the flash crowd scenario is that the system will deal well with new peers connecting, in terms of peering grade and hit-per-request ratio decreasing and then recovering. Having worked with the system, there are concerns that the established NATed peers will not gain knowledge of the newly connected peers. The reasoning for this is that knowledge mainly spreads in network by ping, content and other type of requests. As peers do not send requests to NATed peers (as they do not have knowledge of them, nor can the requests be answered), the established NATed peers probably will not gain knowledge of the newly connected peers and thus could lose out on potential resources, resulting in clusters where no established NATed peers request content from the newly connected peers.

5.2.4 Flash disconnects scenario hypothesis

The hypothesis for the flash disconnects scenario is that the system will deal well with peers flash disconnecting, in terms of peering grade decreasing and then recovering. The hit-per-request ratio is expected to decrease, as peers will most likely have disconnected peers in their *prospect-list*, which will influence the peering grade at first. As knowledge of the P2P network is acquired fast, and echoes through the network, the peers are expected to update their *prospect-lists* and request content from peers that are connected. The factor expected to influence the recovery time the most, is the content sample interval (a factor when considering content statistics in the reevaluation algorithm). A long sample interval should generate low *stratum* oscillation in the network, and long recovery time (from for example flash disconnects or crowds). This phenomenon is expected as recent changes in the P2P network barely make in impact as the content statistics have been gathered over such a significant amount of time (15 minutes). In contrast if the system was to use a short sample interval, this should generate high *stratum* oscillation in the network, and shorter recovery time.

5.3 Optimal scenario

This section presents the results generated when simulating an *optimal* scenario, i.e. a stable streaming session where none of the peers disconnect from the stream. The purpose of simulating this scenario is to investigate if the system behaves as expected and stated in the hypothesis. The results in this scenario can also serve as a benchmark for the results in the other scenarios.

Optimal configurations	
stream session	100 segments
segment length	6s
number of peers (n)	1000
peers' inter arrival time first 10 segments	$\frac{1}{0.05 \cdot n} + rand(0 - 0.005)$ [s]
peers' inter arrival time after 10 segments	$\frac{1}{0.01 \cdot n} + rand(0 - 0.005)$ [s]
disconnect probability all segments	0%

Table 5.3: The optimal configurations.

The peering grade, see Equation 2, is shown for every segment for 100 and 1000 peers in the optimal scenario in Figure 5.1.

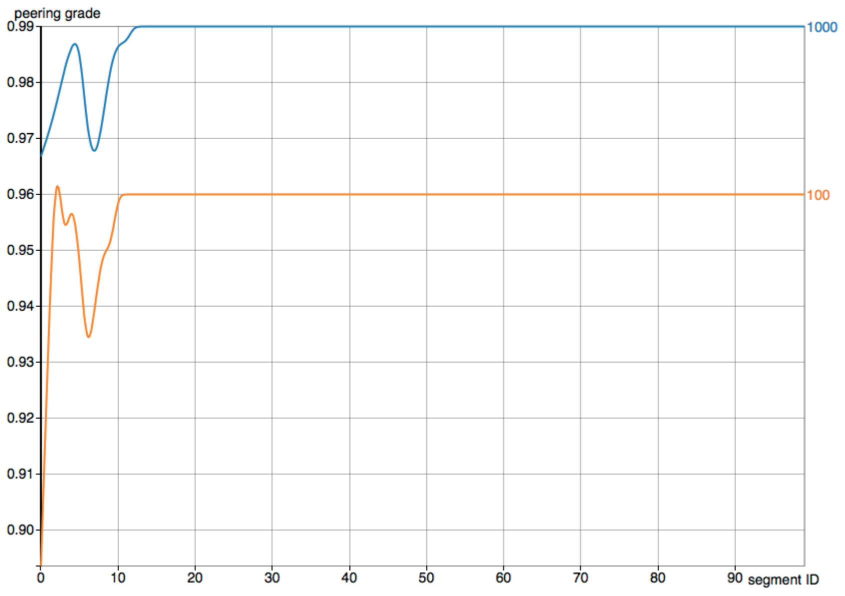

Figure 5.1: Peering grade optimal scenario: results for 100 peers plotted in orange, and 1000 peers plotted in blue.

The hit-per-request ratio (successful requests) in the optimal scenario is visualised for 100 and 1000 peers, see Figure 5.2 and Figure 5.3. In these visualisations, peers are categorised into buckets. Each bucket contains continuous values, for example *stratum* bucket 20 contains all peers with *stratum* 0-20, bucket 40 contains all peers with *stratum* 21-40, and so on.

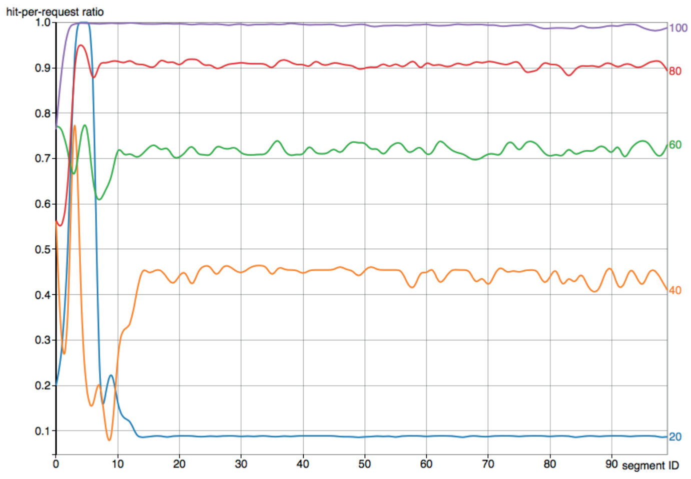

Figure 5.2: Optimal scenario: hit/request-ratio for different stratum buckets (0-20 blue, 21-40 orange, 41-60 green, 61-80 red, 81-100 purple) when simulating 100 peers.

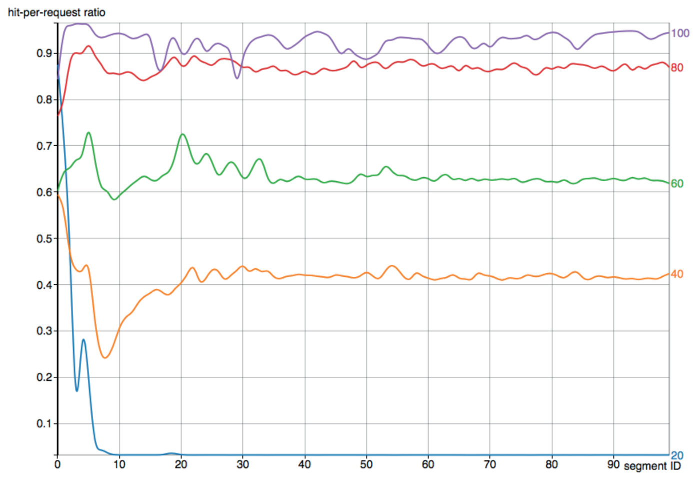

Figure 5.3: Optimal scenario: hit/request-ratio for different stratum buckets (0-20 blue, 21-40 orange, 41-60 green, 61-80 red, 81-100 purple) when simulating 1000 peers.

The network structures (in terms of *stratum* distribution) for the optimal scenario, see Figure 5.4 and Figure 5.5, show the network structures for 100 and 1000 peers.

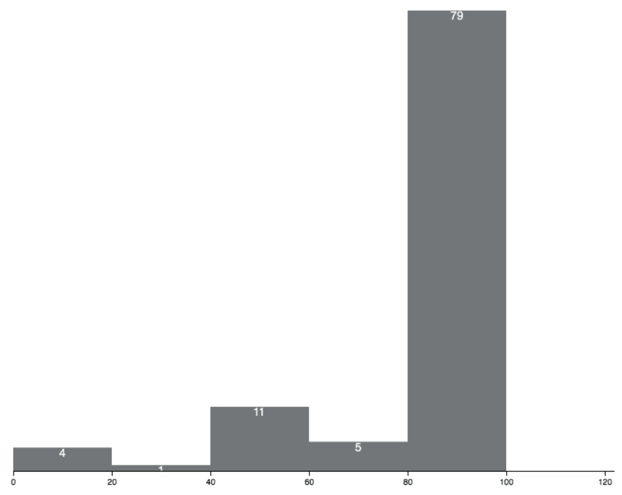

Figure 5.4: Optimal scenario: histogram for the number of peers in the different stratum buckets at the 50th segment, when simulating 100 peers.

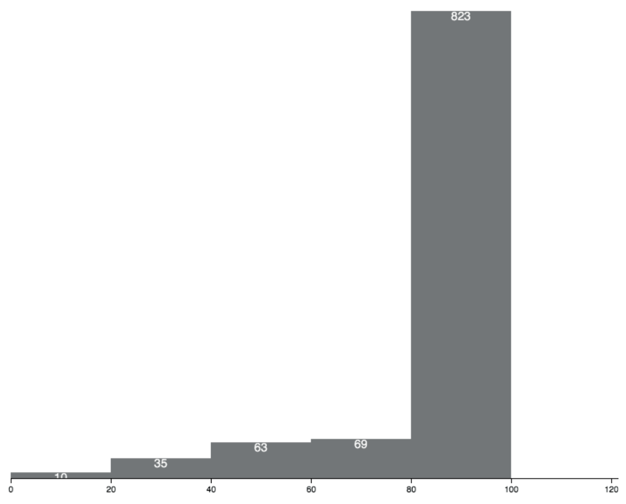

Figure 5.5: Optimal scenario: histogram for the number of peers in the different stratum buckets at the 50th segment, when simulating 1000 peers.

5.4 Default scenario

This section presents the results generated when simulating a *default* scenario, i.e. a streaming session where the peers disconnect from and reconnect to the stream. The purpose of simulating this scenario is to investigate if the system behaves as expected and stated in the hypothesis.

Default configurations	
stream session	100 segments
Segment length	6s
number of peers (n)	100, 1000
peers' inter arrival time first 10 segments	$\frac{1}{0.05 \cdot n} + rand(0 - 0.005)$ [s]
peers' inter arrival time after 10 segments	$\frac{1}{0.01 \cdot n} + rand(0 - 0.005)$ [s]
disconnect probability (P_d) first 85 segments, every 30s	1% 1
P_d after 85 segments, every 30s	previous $P_d + 3\%$
reconnect probability (P_r), every 30s	5%

Table 5.4: The default configurations.

The peering grade, see Equation 2, is shown for every segment for 100 and 1000 peers in the default scenario in Figure 5.6.

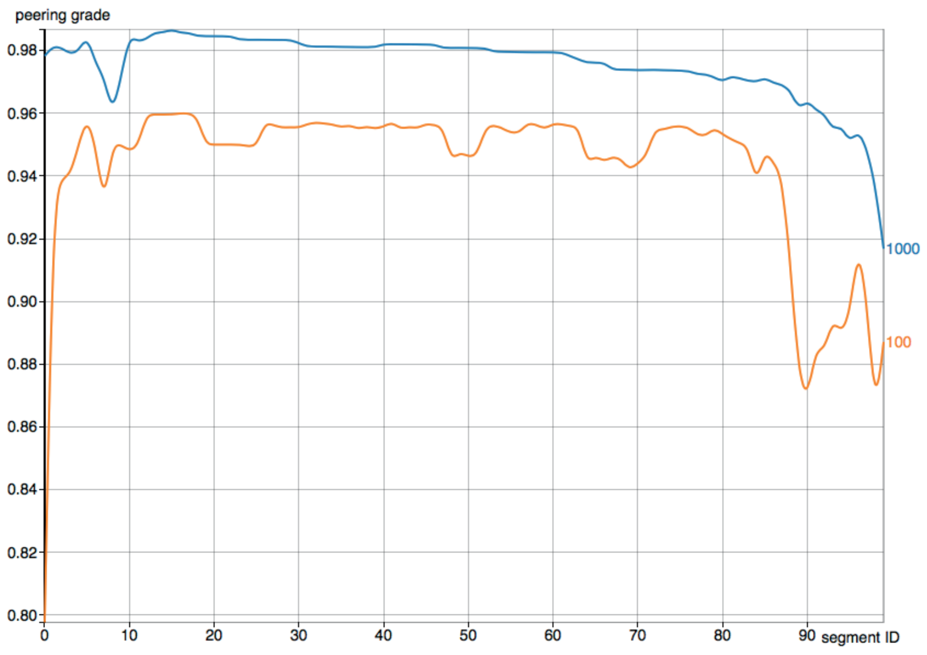

Figure 5.6: Peering grade default scenario: results for 100 peers plotted in orange, and 1000 peers plotted in blue.

The hit-per-request ratio (successful requests) in the default scenario is visualised for 100 and 1000 peers, see Figure 5.7 and Figure 5.8. In these visualisations, peers are categorised into buckets. Each bucket contains continuous values, for example *stratum* bucket 20 contains all peers with *stratum* 0-20, bucket 40 contains all peers with *stratum* 21-40, and so on.

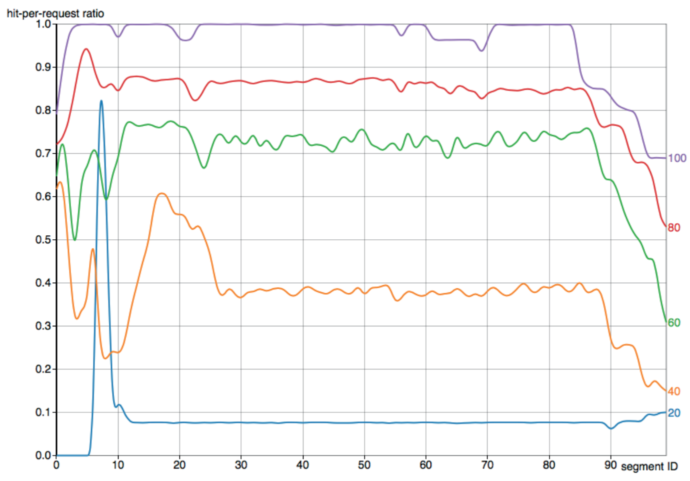

Figure 5.7: Default scenario: hit/request-ratio for different stratum buckets (0-20 blue, 21-40 orange, 41-60 green, 61-80 red, 81-100 purple) when simulating 100 peers.

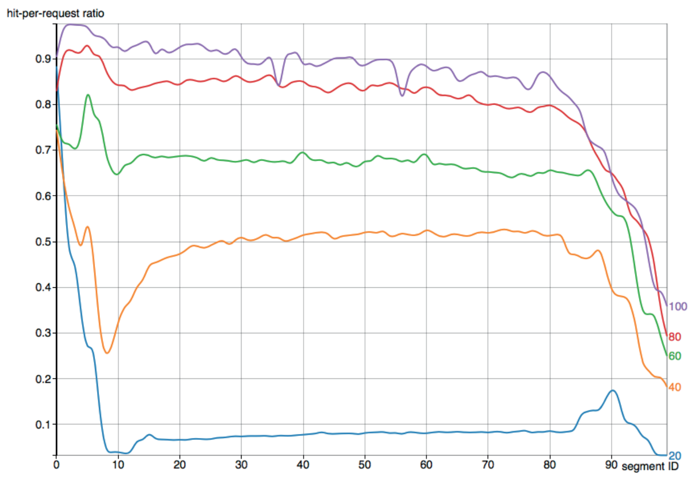

Figure 5.8: Default scenario: hit/request-ratio for different stratum buckets (0-20 blue, 21-40 orange, 41-60 green, 61-80 red, 81-100 purple) when simulating 1000 peers.

The network structures (in terms of *stratum* distribution) for the default scenario, see Figure 5.9 and Figure 5.10, show the network structures for 100 and 1000 peers.

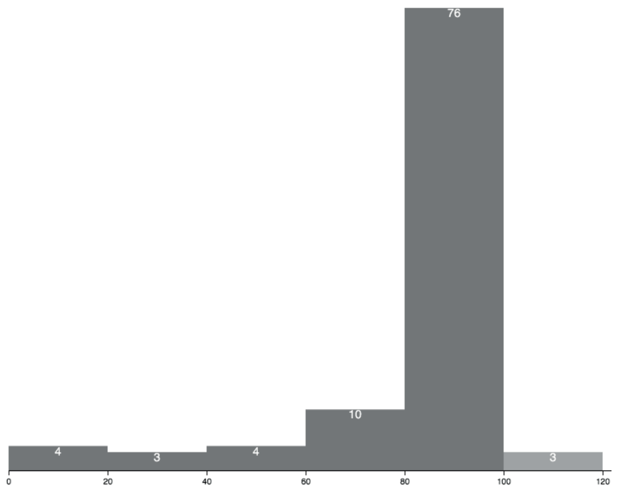

Figure 5.9: Default scenario: histogram for the number of peers in the different stratum buckets (101-120 are disconnected peers) at the 50th segment, when simulating 100 peers.

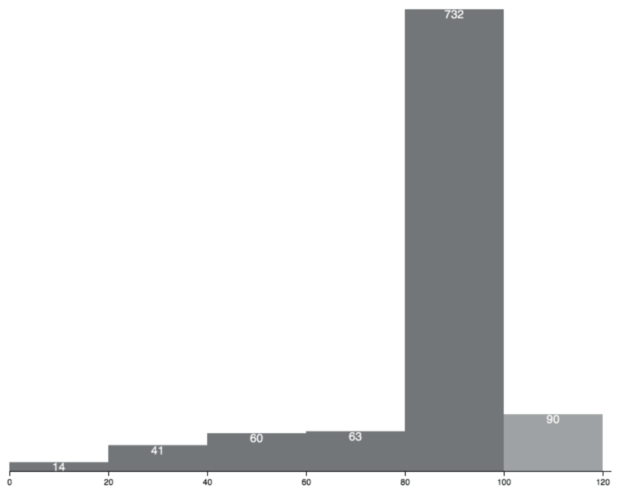

Figure 5.10: Default scenario: histogram for the number of peers in the different stratum buckets (101-120 are disconnected peers) at the 50th segment, when simulating 1000 peers.

5.5 Flash crowd scenario

This section presents the results generated when simulating a *flash crowd* scenario, i.e. a streaming session where the peers disconnect from and reconnect to the stream, and a large amount of new peers suddenly connecting to the stream. The purpose of simulating this scenario is to investigate if the system behaves as expected and stated in the hypothesis.

Flash crowd configurations	
stream session	100 segments
segment length	6s
number of peers (n)	100, 1000
peers' inter arrival time first 10 segments	$\frac{1}{0.05 \cdot n} + rand(0 - 0.005)$ [s]
peers' inter arrival time after 10 segments	$\frac{1}{0.01 \cdot n} + rand(0 - 0.005)$ [s]
number of peers flash connecting	50, 500
specific flash connect time t_{fc}	rand(0-10s)
flash connect occurrence	35th segment + t_{fc}
disconnect probability (P_d) first 85 segments, every 30s	1%
P_d after 85 segments, every 30s	previous $P_d + 3\%$
reconnect probability (P_r), every 30s	5%

Table 5.5: The network configurations used in the simulations.

The peering grade, see Equation 2, is shown for every segment for 100 and 1000 peers in the flash crowd scenario in Figure 5.11.

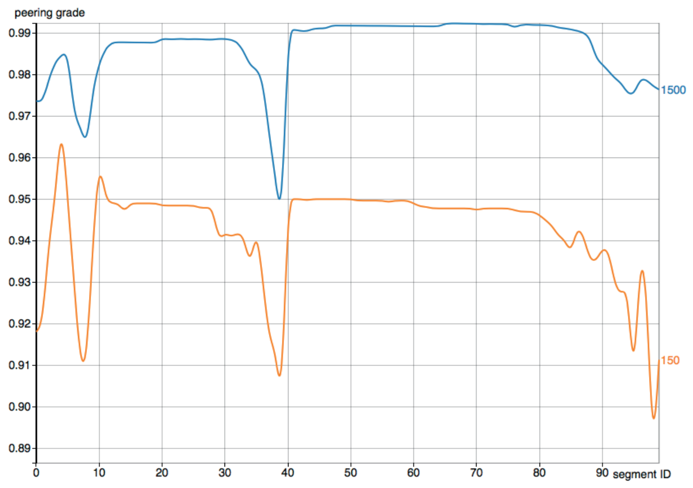

Figure 5.11: Peering grade flash crowd scenario: results for 100 peers plotted in orange, and 1000 peers plotted in blue.

The hit-per-request ratio (successful requests) in the flash crowd scenario is visualised for 100 and 1000 peers, see Figure 5.12 and Figure 5.13. In these visualisations, peers are categorised into buckets. Each bucket contains continuous values, for example *stratum* bucket 20 contains all peers with *stratum* 0-20, bucket 40 contains all peers with *stratum* 21-40, and so on.

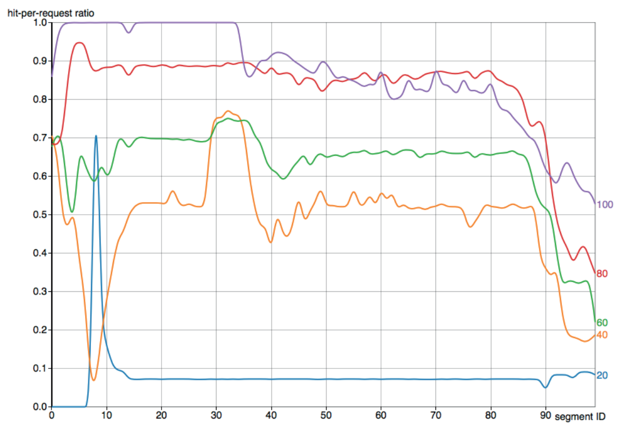

Figure 5.12: Flash crowd scenario: hit/request-ratio for different stratum buckets (0-20 blue, 21-40 orange, 41-60 green, 61-80 red, 81-100 purple) when simulating 100 peers, and 50 peers flash connecting at the 40th segment.

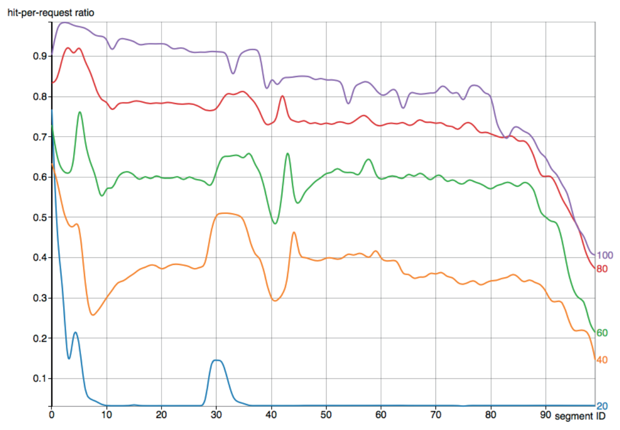

Figure 5.13: Flash crowd scenario: hit/request-ratio for different stratum buckets (0-20 blue, 21-40 orange, 41-60 green, 61-80 red, 81-100 purple) when simulating 1000 peers, and 500 peers flash connecting at the 40th segment.

The network structures (in terms of *stratum* distribution) for the flash crowd scenario show the network structures right before and three minutes after the flash crowd for 150 and 1500 peers, see Figures 5.14, 5.15, 5.16 and 5.17.

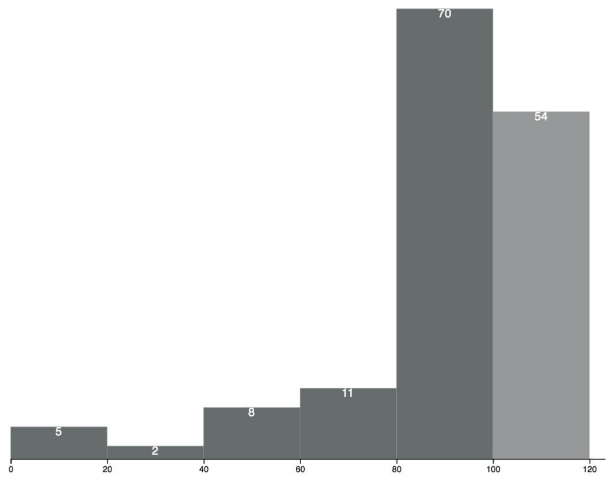

Figure 5.14: Flash crowd scenario: histogram for the number of peers in the different stratum buckets (101-120 are disconnected peers) right before the flash crowd, when simulating 150 peers.

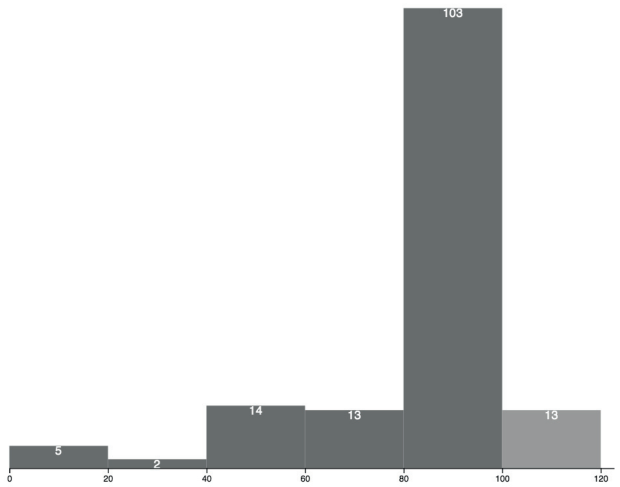

Figure 5.15: Flash crowd scenario: histogram for the number of peers in the different stratum buckets (101-120 are disconnected peers) 3 minutes after the flash crowd, when simulating 150 peers.

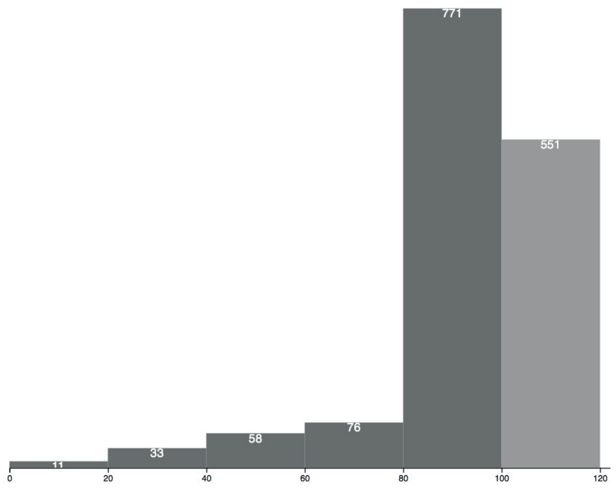

Figure 5.16: Flash crowd scenario: histogram for the number of peers in the different stratum buckets (connected peers in blue, disconnected peers placed in 101-120 in red) right before the flash crowd, when simulating 1500 peers.

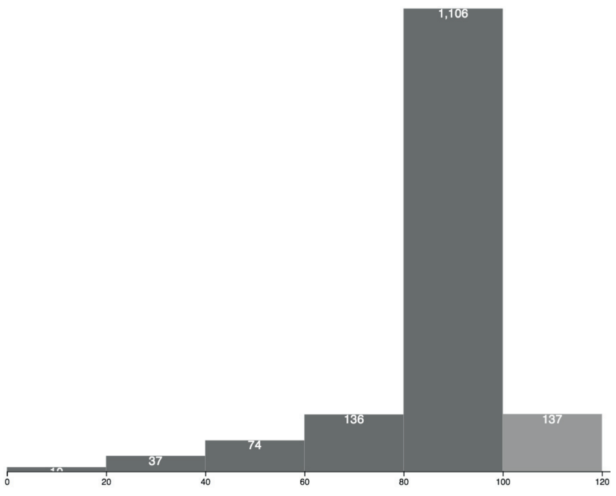

Figure 5.17: Flash crowd scenario: histogram for the number of peers in the different stratum buckets (101-120 are disconnected peers) 3 minutes after the flash crowd, when simulating 1500 peers.

5.6 Flash disconnects scenario

This section presents the results generated when simulating a *flash disconnects* scenario, i.e. a streaming session where the peers disconnect from and reconnect to the stream, and a large amount of new peers suddenly disconnecting from the stream. The purpose of simulating this scenario is to investigate if the system behaves as expected and stated in the hypothesis.

Flash disconnects configurations	
stream session	100 segments
segment length	6s
number of peers (n)	100, 1000
peers' inter arrival time first 10 segments	$\frac{1}{0.05 \cdot n} + rand(0 - 0.005)$ [s]
peers' inter arrival time after 10 segments	$\frac{1}{0.01 \cdot n} + rand(0 - 0.005)$ [s]
flash disconnect quantity	0.5
specific flash disconnect time t_{fc}	rand(0-10s)
flash connect occurrence	35th segment + t_{fc}
disconnect probability (P_d) first 85 segments, every 30s	1%
P_d after 85 segments, every 30s	previous $P_d + 3\%$
reconnect probability (P_r), every 30s	5%

Table 5.6: The flash disconnects configurations.

The peering grade, see Equation 2, is shown for every segment for 100 and 1000 peers in the flash disconnects scenario in Figure 5.18.

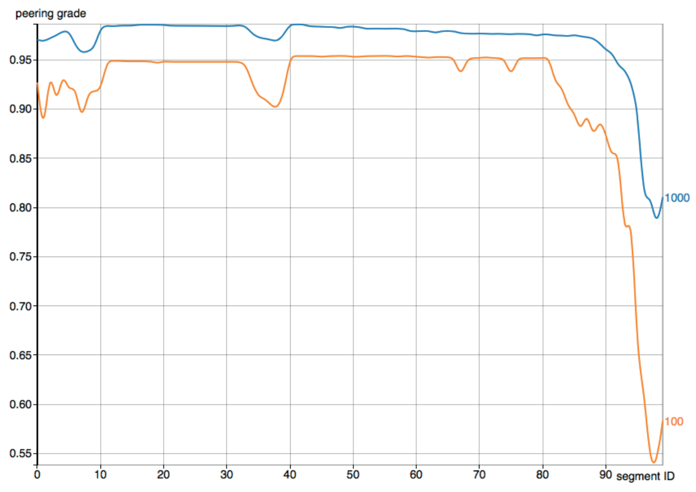

Figure 5.18: Peering grade flash disconnects scenario: results for 100 peers plotted in orange, and 1000 peers plotted in blue.

The hit-per-request ratio (successful requests) in the flash disconnects scenario is visualised for 100 and 1000 peers, see Figure 5.19 and Figure 5.3. In these visualisations, peers are categorised into buckets. Each bucket contains continuous values, for example *stratum* bucket 20 contains all peers with *stratum* 0-20, bucket 40 contains all peers with *stratum* 21-40, and so on.

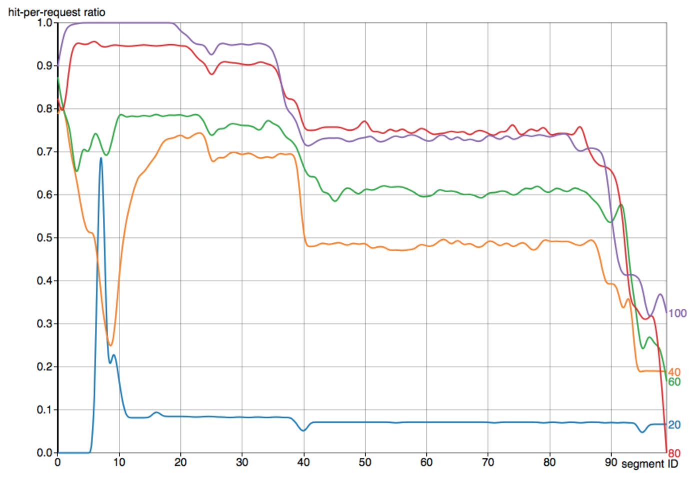

Figure 5.19: Flash disconnect scenario: hit/request-ratio for different stratum buckets (0-20 blue, 21-40 orange, 41-60 green, 61-80 red, 81-100 purple) when simulating 100 peers.

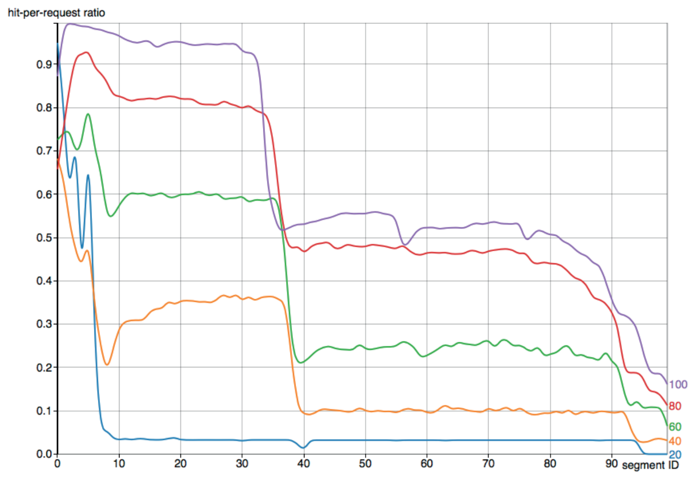

Figure 5.20: Flash disconnect scenario: hit/request-ratio for different stratum buckets (0-20 blue, 21-40 orange, 41-60 green, 61-80 red, 81-100 purple) when simulating 1000 peers.

The network structures (in terms of *stratum* distribution) for the flash disconnects scenario show the network structures right before and three minutes after the flash disconnects for 150 and 1500 peers, see Figures 5.21, 5.22, 5.23 and 5.24.

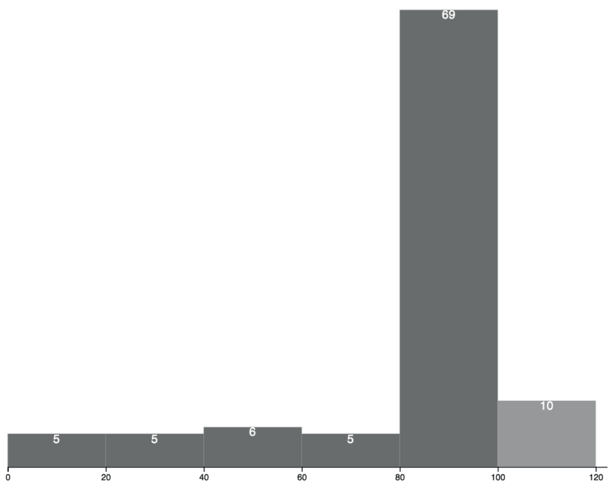

Figure 5.21: Flash disconnects scenario: histogram for the number of peers in the different stratum buckets (101-120 are disconnected peers) right before the flash disconnects, when simulating 100 peers.

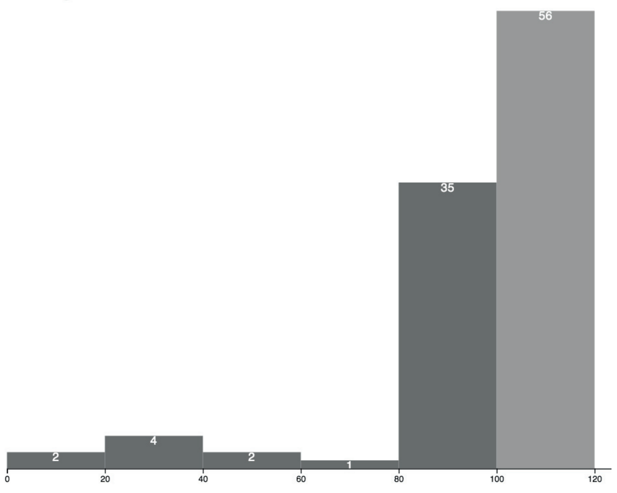

Figure 5.22: Flash disconnects scenario: histogram for the number of peers in the different stratum buckets (101-120 are disconnected peers) 3 minutes after the flash disconnects, when simulating 100 peers.

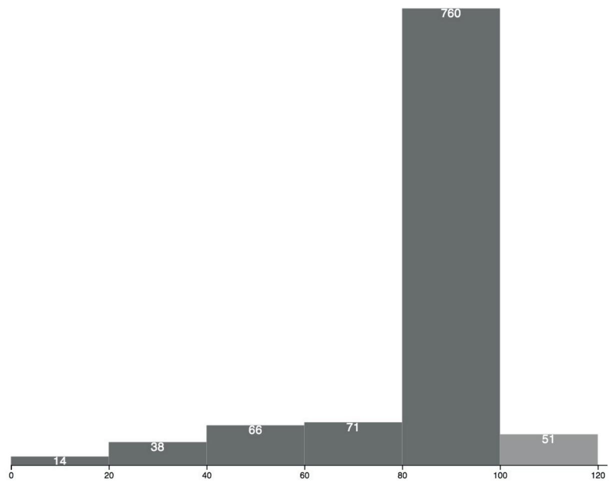

Figure 5.23: Flash disconnects scenario: histogram for the number of peers in the different stratum buckets (101-120 are disconnected peers) right before the flash disconnects, when simulating 1000 peers.

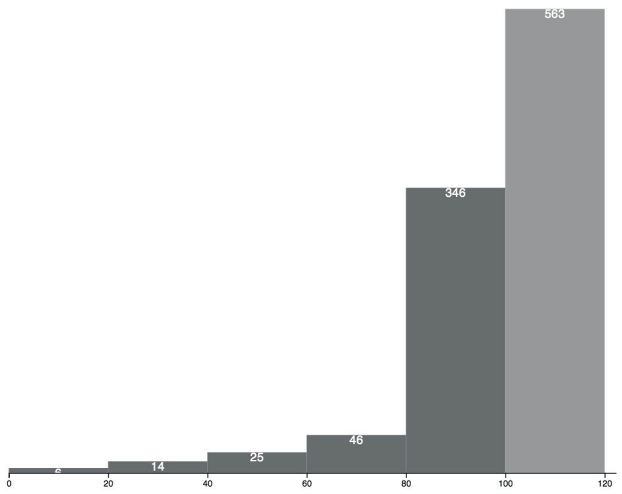

Figure 5.24: Flash disconnects scenario: histogram for the number of peers in the different stratum buckets (101-120 are disconnected peers) 3 minutes after the flash disconnects, when simulating 1000 peers.

Chapter 6
Data Analysis

In this chapter, the results are analysed and discussed, and the system's performance is evaluated for the different scenarios.

6.1 Optimal scenario

The simulations showed that the peering grade is slightly better for 1000 peers, compared to 100 peers, see Figure 5.1. This is expected for mesh-based solutions as they have good scalability. Due to the static environment of this scenario where no peers disconnect, bandwidth is stable over time, and the network structure does not change, the peering grade stabilises. When simulating 100 peers only 4 peers utilised the fallback mechanism resulting in a peering grade of 96%, see Figure A.1 in Appendix A. When simulating 1000 peers, 10 peers utilised the fallback mechanism when resulting in a peering grade of 99%, see Figure A.2. Our hypothesis expected a pyramid like structure, which occurs when simulating 1000 peers. However, this is less evident when simulating 100 peers, see Figures 5.4 and 5.5. Although simulating 100 peers did not generate the expected pyramid like network structure, the system's overall performance was good due to peers requesting content from multiple layers, see Figure A.1. Independent of the network's structure peers with higher *stratum* wait longer before requesting content from peers, which should lead to higher *stratum* having better hit-per-request ratios. This occurs when simulating both 100 and 1000 peers confirming our hypothesis, see Figures 5.2 and 5.3.

6.2 Default scenario

The simulations showed that the peering grade is slightly better for 1000 peers, compared to 100 peers, see Figure 5.6. As in the previous scenario, this is expected for mesh-based solutions. As expected the peering grade is similar, yet slightly lower and does not stabilise over time, compared to the optimal scenario. As more peers disconnect than reconnect, the peering grade decreases slightly over time. The hit-per-request ratios are similar, see Figures 5.7 and 5.3 compared to the optimal scenario. Surprisingly the hit-per-request ratios are slightly higher in the default scenario compared to the optimal scenario (until the end of the stream when peers increasingly disconnect). The network structures is more pyramid-like when simulating 1000 compared to 100 peers, see Figures 5.10 and 5.9, the same tendency as in the optimal scenario.

6.3 Flash crowd scenario

The simulations showed that the peering grade decreased and then recovered (even increasing), at the occurance of the flash crowds, see Figure 5.11. The simulations showed that 1000 peers receiving 500 flash connected peers, outperformed 100 peers receiving 50 peers, in terms of peering grade. The system deals well with flash crowds, this is expected due to mesh-based solutions' robustness. Although the peering grade recovers, the hit-per-request ratio does not, see Figures 5.12 and 5.13. This could be linked to the hypothesised clustering behaviour which occurs, see Figure A.3 in Appendix A. This visualisation shows that no established peers obtain content from the newly connected peers, thus more peers request content without utilising all resources and this results in lower hit-per-request ratio as more peers compete for bandwidth. The network structures are similar right before the flash crowd and three minutes after the flash crowd suggesting that network reconfigures itself, see Figures 5.14, 5.15, 5.16 and 5.17. Like in the previous scenarios the network takes a more pyramid like structure when simulating 1000 peers compared to 100.

6.4 Flash disconnects scenario

The simulations showed that the peering grade decreased slightly and then recovered, at the occurance of flash disconnects, see Figure 5.18. The simulations showed that the peering grade is slightly better for 1000 peers, compared to 100. The system deals well with flash disconnects, this is expected for mesh-based solutions. Like for the flash crowd scenario the peering grade recovers, while the hit-per-request ratio does not, see Figures 5.12 and 5.13. Furthermore, the hit-per-request ratio recovers less compared to the flash crowds scenario. This can be explained by that resources are increased in the flash crowd scenario, while resources are lost in the flash disconnects scenario. The network structures are similar right before the flash disconnects and three minutes after the flash disconnects suggesting that network reconfigures itself, see Figures 5.21, 5.22, 5.23 and 5.24. Like in the previous scenarios the network takes a more pyramid-like structure when simulating 1000 peers compared to 100.

6.5 Improvements

From the data analysis it was concluded that the system had problems towards the end of the streaming session when faced with flash disconnects. This is thought to occur as the system has already lost a lot of resources in the flash disconnects. As stated in our hypothesis, and discussed in section 6.4, it was possible to increase the system's responsiveness and robustness by decreasing the sample interval (a factor when considering content statistics in the reevaluation algorithm) for the content statistics. When decreasing the sample interval from 15 minutes to 1 minute, the system considered more recent content statistics and was better equipped to adapt to changes. The system performed better towards the end of the stream in terms of peering grade when peers increasingly disconnected, see Figures 6.1 and 6.2. However, utilising shorter content sample intervals could lead to oscillations in *stratum*, and thus the network will reconfigure itself constantly.

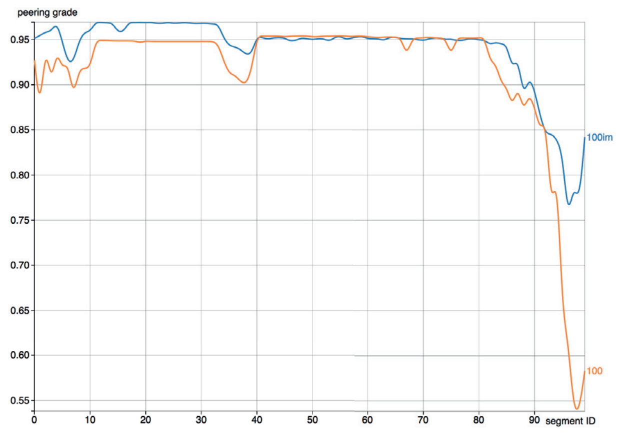

Figure 6.1: Peering grade: 100 peers with a content sample interval of 15 minutes (orange, label: 1000), and 100 peers with a content sample interval of 1 minute (blue, label: 100im).

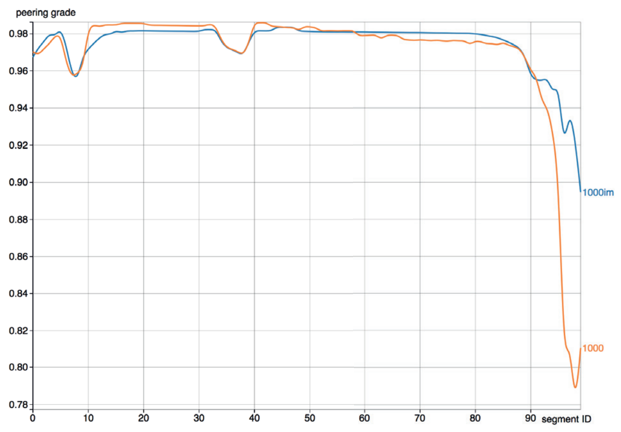

Figure 6.2: Peering grade: 1000 peers with a content sample interval of 15 minutes (orange, label: 1000), and 1000 peers with a content sample interval of 1 minute (blue, label: 1000im).

Chapter 7 Discussion

In this chapter the results and data analysis are discussed, and includes possible sources of error and future work.

7.1 Simulation environment

Simulating few peers was a conscious choice, as it is easier to analyse and visualise the behaviour of a smaller P2P network. When simulating relatively small P2P networks, the simulations are affected more by randomness (say a peer with high upload contribution disconnecting, compared to a peer with low, or none, upload contribution disconnecting). Repeated simulation of the same scenarios behaved very similarly when simulating a 1000 peers, and varied slightly when simulating a 100 peers. The simulations showed the same behaviour when repeating the scenarios.

The simulation environment consisted of assigning X and Y coordinates to the peers so that distance between peers would influence the latency. In the simulations the peers are distributed in an area approximately twice the size of Lund, Sweden. It could be interesting to simulate peers having much greater distances in order to analyse the latency's implication. As the simulation environment is limited to assigning the peers to one of three ISPs, this could influence the results generated in the simulations. Simulating with more available ISPs, and different boarder crossing latencies between the different ISPs, could improve the accuracy of the results. Further developing the ISP and latency modelling could for example simulate peers watching the same stream and distributing content between each other, although being positioned at different parts of the world.

7.2 System specific findings

A finding from the literature indicated that a way to regulate the admittance of free-riders is to use algorithms potentially blocking these peers from connecting to the network. The algorithms mentioned in [27], make use of heuristics in the peers to estimate the capacity of the network. Our results are based on an approach accepting all peers (free-riders are welcome). Although free-riders are unable to contribute their resources to the system, the simulated system accepts them and assigns them the lowest priority. The reason being that all content that can be provided to them from peers, and not the fallback mechanism, reduces the cost. Although the system admitted around 80% free-riders, and attempting to minimise data obtained from the CDN, our experiments indicates that approximately 95% of all data exchanges between peers.

An interesting artefact is that although the network has a complex structure, and no centralised entity controlling the content flow, the system quickly structures and stabilises itself, allowing for good peering grade.

Our findings show that for a live streaming service, where the majority of the peers are not able to contribute resources to the system, the utilisation of the CDN can be reduced without a significant impact for the end users.

The results show that although most content is distributed by the P2P network, there is a lot to gain by using the CDN as a fallback mechanism. Newly connected peers can utilise the fallback mechanism while finding neighbouring peers and while the P2P network is reconfiguring. The fallback mechanism functions as a safety measure so the peers can fulfil their (content) contract, thus increasing the system's scalability and robustness. CDN data transfers were registered at around 5% in a live environment, with 80% of the network consisting of free-riders. Indicating that this is a stable solution, as simulation of the scenarios showed that the mesh properties of the network dealt with user churn sufficiently.

When performing the data analysis it was concluded that reducing the content sample interval, and structuring the network based on more recent content statistics, the system's adaptiveness improved when simulating the flash disconnects scenario. This improvement should be further analysed due to its interesting side effect. When resetting the content statistics more frequently, *stratum* oscillations were introduced, see Figure B.1 in Appendix B. Peers frequently changing *stratum* could complicate the selection process when determining which neighbouring peers to request content from, possibly introducing a ripple effect through the network.

The simulations and visualisations made it is easier to understand key miss concepts, faults in the network on a topology level, and provided a basis to find configurations and alterations of the network, that could further improve the system.

7.3 Sources of error

A main aspect of P2P networks, is how data is transmitted and received over the Internet. This is very complex and hard to model, and in parts of our network modelling assumptions and cut-offs could be factors for errors. Instead of creating the whole network stack and simulating how lower network layers influence the transmission, the simulation is based on assumptions. In the master's thesis, components of the P2P system were generalised. Allowing to study the system when simulating different scenarios, and to find tendencies of its overall performance. However, as the system and the network model are generalised, the results could potentially be influenced by unintentionally neglecting important artefacts. Generalising the network modelling in terms of simulating with static bandwidth and a generous bandwidth model, and accurately modelling user behaviour, could be a factor in the simulations' high and stable peering grade.

It is of course in the realm of possibility, that bugs slipped through the screening process and code-review conducted by ourselves, and Voddler.

We have seen that most trends scale quite well from 100 to 1000 peers, but we can not say for certain that it will scale well beyond that. We theorise that a network of 1000 peers would be enough to find most trends on a larger scale.

7.4 Future work

We suggest creating a more advanced, or somehow utilise, a more advanced network model. The generalisation of the system of course plays a part in this, however it is our opinion that the network model is extremely important in order to obtain more accurate results.

In the scope of this master's thesis we neglected multi bitrate (adaptive streaming). It would be interesting to introduce multi bitrate and see its results when simulating our scenarios. Studying multi bitrate could result in new findings, potentially sub clusters for the different bitrates. As peers with good resources can download additional bitrates to the one it is streaming, it could help the other clusters with their peering. This is a potential bridge between the different clusters?

Are P2P networks wanted in a environment with mobile phones, what are the effects in terms of energy consummation and data transfers, should they be freeloaders?

We have seen studies that point out the problems of P2P networks operating across ISP boundaries, in terms of latency and clustering effects and the problems this imposes. In this thesis the boundary crossing is taken into account, but no further analysis is done. This could be interesting to study to see how different configurations impact the performance and behaviour.

Chapter 8 Conclusions

This master’s thesis has investigated and studied how an actual implementation of a hybrid P2P live video streaming system performs under different scenarios by simulation. The study showed that the mesh-based P2P solution performed well when simulating under user churn. When simulating with approximately 80% free-riders the CDN utilisation was reduced with 95% on average. The system dealt better with the flash crowd scenario than the flash disconnects scenario, in terms of peering grade and hit-per-request ratio recovering.

When performing the data analysis, it was determined that the P2P network adopted the expected structure when simulating larger networks. Although simulating small networks resulted in less of a pyramid like structure, good peering grade was achieved. The mesh-based approach and unstructured network utilising a fallback mechanism provides robustness against user churn.

The system performed well in all scenarios when simulated. To further improve the accuracy and credibility of the results, we suggest utilising a more detailed and complex network model.

P2P networks are complex and difficult to theorise, seen in our partially incorrect hypotheses. Therefore, we found that the simulations and visualisations were of great assistance in understanding the system’s behaviour and overall performance.

Bibliography

[1] VNI Cisco. "Cisco Visual Networking Index: Forecast and Methodology 2014–2019 White Paper". In: *Cisco, Tech. Rep* (2015).

[2] FENG Li. "The Future of Video". In: (2016).

[3] Andrew Fecheyr-Lippens. "A review of http live streaming". In: *Internet Citation* (2010), pp. 1–37.

[4] *Youtube*. `https://www.youtube.com/`. Accessed: 2017-10-13.

[5] Apple Inc. *HTTP Live Streaming Overview*. 2009. URL: `http://developer.apple.com/iphone/library/documentation/NetworkingInternet/Conceptual/StreamingMediaGuide` (visited on 06/29/2017).

[6] P. Hillmann et al. "Modeling the Location Selection of Mirror Servers in Content Delivery Networks". In: *2016 IEEE International Congress on Big Data (BigData Congress)*. June 2016, pp. 438–445. DOI: `10.1109/BigDataCongress.2016.68`.

[7] Roberto Roverso. "A System, Tools and Algorithms for Adaptive HTTP-live Streaming on Peer-to-peer Overlays". PhD thesis. KTH Royal Institute of Technology, 2013.

[8] *RFC 6886, 2013. NAT Port Mapping Protocol (NAT-PMP) [online] RFC*. `https://tools.ietf.org/html/rfc6886`. Accessed: 2017-10-28.

[9] *RFC 6887, 2013. Port Control Protocol (PCP) [Online] RFC*. `https://tools.ietf.org/html/rfc6887`. Accessed: 2017-10-28.

[10] Hidekazu Suzuki, Yuji Goto, and Akira Watanabe. "External dynamic mapping method for NAT traversal". In: *2007 International Symposium on Communications and Information Technologies*. Oct. 2007, pp. 723–728. DOI: `10.1109/ISCIT.2007.4392111`.

[11] Felipe Rocha Wagner, Marcio Garcia Martins, and Arthur Tórgo Gómez. "A Peer to Peer Architecture Applied to Multiplayer Games". In: *ICN 2015* (2015), p. 66.

[12] De Boever. "Peer-to-Peer Networks as a Distribution and Publishing Model". In: *ELPUB*. 2007.

[13] Xinyan Zhang et al. "CoolStreaming/DONet: A data-driven overlay network for peer-to-peer live media streaming". In: *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*. Vol. 3. IEEE. 2005, pp. 2102–2111.

[14] Xiaojun Hei et al. "A measurement study of a large-scale P2P IPTV system". In: *IEEE transactions on multimedia* 9.8 (2007), pp. 1672–1687.

[15] L. Massoulie. "Peer-to-peer live streaming: Optimality results and open problems". In: *2008 42nd Annual Conference on Information Sciences and Systems*. Mar. 2008, pp. 313–315. DOI: `10.1109/CISS.2008.4558542`.

[16] A. W. AlTuhafi, S. Ramadass, and Y. W. Chong. "Concepts and types of peer-to-peer network topology for live video streaming". In: *2013 IEEE International Conference on RFID-Technologies and Applications (RFID-TA)*. Sept. 2013, pp. 1–4. DOI: `10.1109/RFID-TA.2013.6694540`.

[17] Thomas Locher et al. "Push-to-pull peer-to-peer live streaming". In: *Distributed Computing* (2007), pp. 388–402.

[18] Xiaojun Hei, Yong Liu, and Keith W Ross. "IPTV over P2P streaming networks: the mesh-pull approach". In: *IEEE Communications Magazine* 46.2 (2008).

[19] Thomas Karagiannis et al. "Is p2p dying or just hiding?[p2p traffic measurement]". In: *Global Telecommunications Conference, 2004. GLOBECOM'04. IEEE*. Vol. 3. IEEE. 2004, pp. 1532–1538.

[20] Vinay Aggarwal, Anja Feldmann, and Christian Scheideler. "Can ISPs and P2P users cooperate for improved performance?" In: *ACM SIGCOMM Computer Communication Review* 37.3 (2007), pp. 29–40.

[21] Kolja Eger et al. "Efficient Simulation of Large-scale P2P Networks: Packet-level vs. Flow-level Simulations". In: *Proceedings of the Second Workshop on Use of P2P, GRID and Agents for the Development of Content Networks*. UPGRADE '07. Monterey, California, USA: ACM, 2007, pp. 9–16. ISBN: 978-1-59593-718-6. DOI: `10.1145/1272980.1272986`. URL: `http://doi.acm.org/10.1145/1272980.1272986`.

[22] Chi-Wen Lo et al. "A packet loss estimation model and its application to reliable mesh-based P2P video streaming". In: *Multimedia and Expo (ICME), 2011 IEEE International Conference on*. IEEE. 2011, pp. 1–6.

[23] Philipp Berndt, Dominic Battré, and Odej Kao. "A hybrid approach to modeling end-to-end delay in P2P networks". In: *Proceedings of the 2010 ACM workshop on Advanced video streaming techniques for peer-to-peer networks and social networking*. ACM. 2010, pp. 37–42.

[24] TS Eugene Ng and Hui Zhang. "Global network positioning: a new approach to network distance prediction." In: *Computer Communication Review* 32.1 (2002), p. 61.

[25] Ingmar Baumgart, Bernhard Heep, and Stephan Krause. "OverSim: A flexible overlay network simulation framework". In: *IEEE Global Internet Symposium, 2007*. IEEE. 2007, pp. 79–84.

[26] Sebastian Kaune et al. "Modelling the Internet Delay Space Based on Geographical Locations". In: *Proceedings of the 2009 17th Euromicro International Conference on Parallel, Distributed and Network-based Processing*. PDP '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 301–310. ISBN: 978-0-7695-3544-9. DOI: 10.1109/PDP.2009.44. URL: http://dx.doi.org/10.1109/PDP.2009.44.

[27] Ilias Chatzidrossos. "Live Streaming Performance of Peer-to-Peer Systems". PhD thesis. KTH Royal Institute of Technology, 2012.

[28] D. Ren, Y. T. Li, and S. H. Chan. "On Reducing Mesh Delay for Peer-to-Peer Live Streaming". In: *IEEE INFOCOM 2008 - The 27th Conference on Computer Communications*. Apr. 2008. DOI: 10.1109/INFOCOM.2008.160.

[29] F. Pianese et al. "PULSE: An Adaptive, Incentive-Based, Unstructured P2P Live Streaming System". In: *IEEE Transactions on Multimedia* 9.8 (Dec. 2007), pp. 1645–1660. ISSN: 1520-9210. DOI: 10.1109/TMM.2007.907466.

[30] Z. Liu et al. "LayerP2P: Using Layered Video Chunks in P2P Live Streaming". In: *IEEE Transactions on Multimedia* 11.7 (Nov. 2009), pp. 1340–1352. ISSN: 1520-9210. DOI: 10.1109/TMM.2009.2030656.

[31] *Golang documentation*. https://golang.org/doc/. Accessed: 2017-10-16.

[32] *JSON*. https://www.python.org/. Accessed: 2017-10-31.

[33] *Python*. http://www.json.org/. Accessed: 2017-10-31.

[34] *D3.js*. https://d3js.org/. Accessed: 2017-10-30.

[35] *Golang gorutines*. `https://golangbot.com/goroutines/`. Accessed: 2017-10-16.

Appendix A

Content distribution visualisations

The content flow for a certain chunk from the CDN through the P2P network in the optimal scenario is visualised for 100 peers in Figure A.1, and for 1000 peers in Figure A.2.

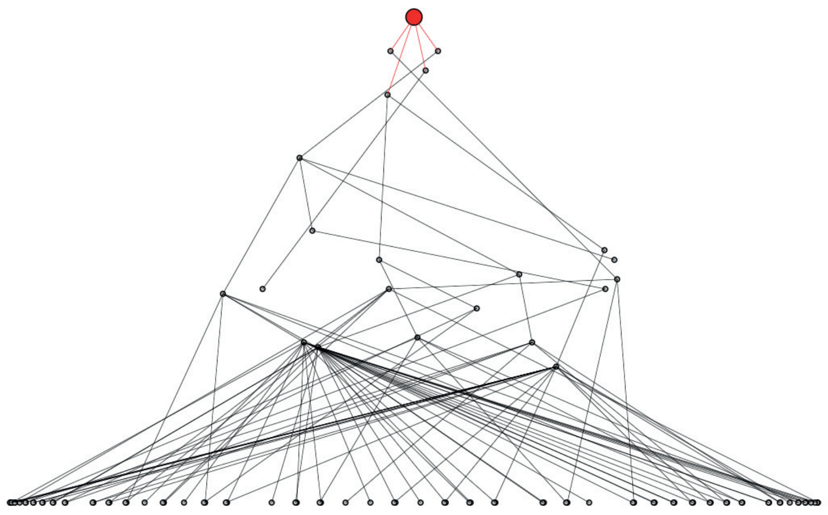

Figure A.1: Optimal scenario: content flow for certain chunk, from CDN (red node and lines) through the P2P network for 100 peers, at the 40th segment.

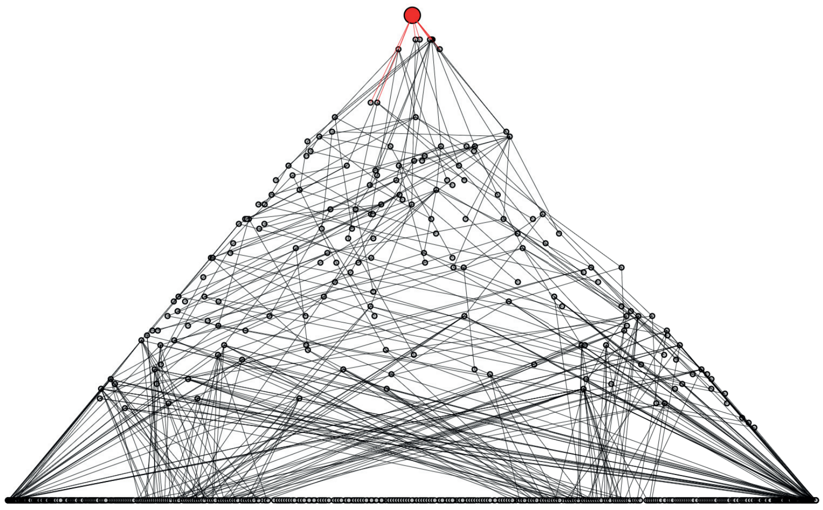

Figure A.2: Optimal scenario: content flow for certain chunk from CDN (red node and lines) through the P2P network for 1000 peers, at the 40th segment.

The content flow for a certain chunk from the CDN through the P2P network in the flash crowd scenario is visualised for 100 peers in Figure A.3.

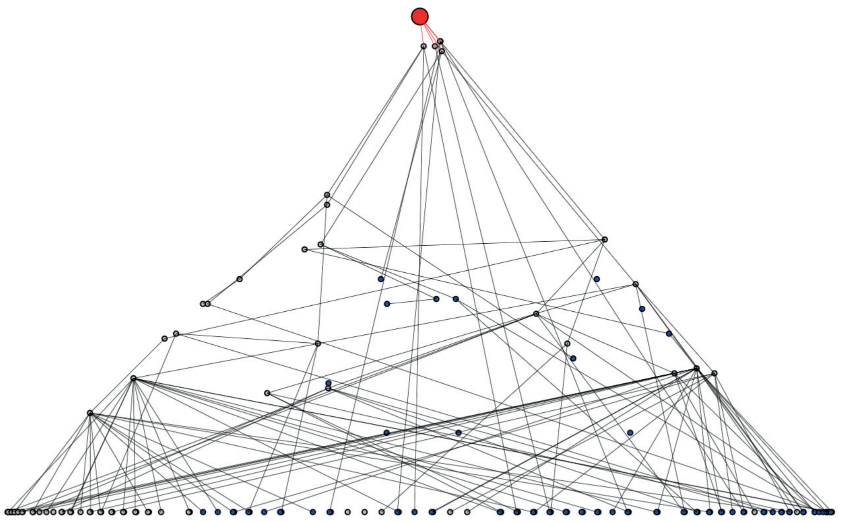

Figure A.3: Flash crowd scenario: content flow from CDN (red node and lines) through the P2P network for 100 peers, 1 minute after the flash crowd.

Appendix B

Network structure

A snapshot of the network structure from the flash disconnect scenario when simulating a 100 peers with a content sample interval of 1 minutes. The arrows in the visualisation show which *stratum* values the peers oscillate between.

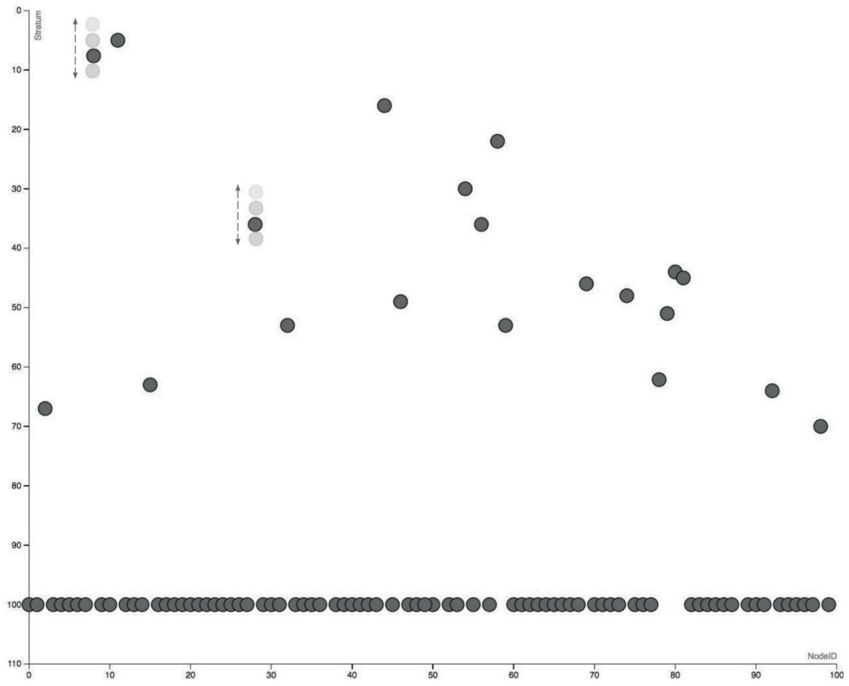

Figure B.1: Flash disconnect scenario: snapshot of *stratum* oscillation in the network structure when simulating 100 peers.

Printed by Tryckeriet i E-huset, Lund 2017

Series of Master's theses
Department of Electrical and Information Technology
LU/LTH-EIT 2017-611

http://www.eit.lth.se