Measurement Study of Peer-to-Peer Multimedia Streaming Systems

José Júlio Rodrigues Lucas

Thesis to obtain the Master of Science Degree in

Telecommunications and Informatics Engineering

Supervisors: Prof. Rui António dos Santos Cruz
Prof. Mário Serafim dos Santos Nunes

Examination Committee

Chairperson: Prof. Paulo Jorge Pires Ferreira
Supervisor: Prof. Rui António dos Santos Cruz
Member of the Committee: Prof. Artur Miguel do Amaral Arsénio

June 2015
Acknowledgments

First I would like to express my gratitude to my supervisors, Dr. Rui Santos Cruz and Dr. Mário Serafim Nunes, who proposed me this work, and for their expertise, support and availability. A special thank to Dr. Rui Santos Cruz, who, with his kind words and optimism along these months, made me believe that I would accomplish the goals of this project.

I would also like to thank João Pedro Taveira of INESC-ID, for the help and availability in the beginning of this work.

I would like to thank my colleges and managers at MEO and Neos of Bold International group, who supported me and proved to be always available.

A very special thanks goes out to Ricardo Ressurreição, who with his great technical knowledge and his friendship, gave me the key advises for developing this work.

To all my friends, they know who they are, thank you for the patience and support.

To my girlfriend's parents, Janete e Agostinho, who always believed in me. Thank you for the encouragement and dedication.

I would like to thank my parents for their friendship, encouragement and caring over all these years, for always being there for me and without whom this project would not have been possible. Particularly to my mother, who always fought for me to realize this dream, and is no longer among us to be able to see it become a reality.

Last but not least, to my girlfriend Iolanda, for being supportive and patience whenever I needed any advice or opinion. Her desire and willingness to share with me all the doubts that have arisen, always gave me the strength to continue. More than anybody, she was by my side when I needed the most. She's the one!

To each and every one of you – Thank you.
Abstract

Large-scale and heterogeneous peer-to-peer (P2P)-based multimedia streaming systems, for the distribution of Live and time-shifted (or On-demand) contents, have emerged in last years. However, most, if not all, of current solutions are closed-source or proprietary in terms of both architecture and streaming protocols used, and hardly reach high-quality streaming levels. The work developed in this thesis advances with a proposal of a monitoring solution for P2P video streaming systems that use the protocols being developed by the P2P Streaming Protocol (PPSP) working group of IETF. The monitoring system provides traffic analysis, with end-users able to monitor quality and system information on a web-based interface. Monitoring agents, implemented in peers, trackers and media serving nodes, are able to collect adequate information about the received quality of adaptive and scalable 2D/3D streamed contents in the P2P network. This information is useful for helping service providers to assess the streaming system performance as well as the quality of the service.

Keywords

P2P Multimedia Streaming Systems; PPSP; Quality Assessment; Monitoring System.
Resumo

Nos últimos anos surgiram Sistemas de Distribuição (Streaming) Multimédia heterogéneos e de larga escala baseados em tecnologia par-a-par (P2P), para a distribuição, em tempo-real ou em tempo-diferido de conteúdos. No entanto, a maioria, se não todas as soluções actuais, são fechadas ou proprietárias, no que diz respeito à sua arquitetura e protocolos utilizados e, dificilmente podem atingir níveis de distribuição com qualidade elevada. Esta Tese apresenta uma proposta de solução para monitorização de sistemas de distribuição de vídeo em P2P, utilizando os protocolos que têm sido desenvolvidos pelo grupo de trabalho P2P Streaming Protocol (PPSP) do IETF. O sistema de monitorização fornece análise de tráfego, a qualidade de experiência do utilizador final e informações de sistema, disponíveis em ambiente web. Os agentes de monitorização, implementados nos nós (peers) da rede P2P, nos trackers e nos nós servidores dos conteúdos, são capazes de recolher informações adequadas sobre a qualidade recebida de conteúdos de tipo adaptável e escalável, 2D/3D, transmitidos na rede P2P. Esta informação é bastante útil para avaliar o desempenho do sistema de distribuição, bem como a qualidade do serviço, importante para os operadores e fornecedores de serviços.

Palavras Chave

Sistemas de streaming P2P; PPSP; Avaliação de Qualidade; Sistema de Monitorização.
## Contents

1 Introduction
   1.1 Motivation and Objectives .................................................. 3
   1.2 Contributions ................................................................. 4
   1.3 Thesis layout ................................................................. 5

2 State-of-the-art
   2.1 P2P Multimedia Streaming ................................................... 9
   2.2 Overlay Topology ............................................................. 9
      2.2.1 Structured Systems ...................................................... 10
      2.2.2 Unstructured Systems ................................................ 10
   2.3 Tree-based P2P Streaming Systems ....................................... 11
      2.3.1 CoolStreaming ......................................................... 12
      2.3.2 Conviva ................................................................. 12
   2.4 Mesh-based P2P Streaming Systems ....................................... 13
      2.4.1 Octoshape ............................................................... 13
      2.4.2 PPLive ................................................................. 13
      2.4.3 Zattoo ................................................................. 14
      2.4.4 Tribler ................................................................. 14
      2.4.5 QQLive ................................................................. 15
      2.4.6 PPStream ............................................................... 15
      2.4.7 SopCast ............................................................... 16
      2.4.8 UUSee ................................................................. 16
   2.5 Hybrid P2P Streaming Systems ............................................. 16
      2.5.1 New Coolstreaming ................................................... 17
      2.5.2 Splitstream ........................................................... 17
      2.5.3 Anysee ................................................................. 17
   2.6 The P2P Streaming Protocols of IETF ................................... 18
   2.7 Streaming Measurement Metrics .......................................... 20
# List of Figures

1.1 The Peer-to-Peer Streaming Protocol (PPSP) Ecosystem .............................. 4

2.1 Tree-based overlay network ................................................................. 10
2.2 Mesh-based overlay network ................................................................. 11
2.3 PPSP System Architecture ....................................................................... 18
2.4 QoS and QoE overview for video streaming ............................................ 21

3.1 Log Classification .................................................................................... 28
3.2 Monitoring System Architecture ............................................................. 29
3.3 Log Sequence .......................................................................................... 30
3.4 Database UML ......................................................................................... 32
3.5 E-R Model ............................................................................................... 32
3.6 PPSP arch ............................................................................................... 34
3.7 User interface of the Monitoring system .................................................. 36

4.1 General Test Scenario .............................................................................. 40
4.2 Peers actions ............................................................................................ 42
4.3 Peer Mode ............................................................................................... 43
4.4 System Metrics ......................................................................................... 44
4.5 QoS Metrics: Upload Traffic ................................................................. 45
4.6 QoS Metrics: Download Traffic .............................................................. 45
4.7 Start-Up time for Test 2. ........................................................................ 47
4.8 Start-Up time for Test 3. ........................................................................ 48
4.9 Start-Up time for Test 4. ........................................................................ 49
4.10 Start-Up time for Test 5. ....................................................................... 50
4.11 Re-buffer Events for Test 3. ................................................................. 51
4.12 Re-buffer Events for Test 4. ................................................................. 53
4.13 Re-buffer Events for Test 5. ................................................................. 54
List of Tables

4.1 Network Link Profiles .................................................. 39
4.2 Video Characteristics .................................................. 39
4.3 SVC Layers Size ......................................................... 39
4.4 Metrics Descriptions .................................................... 40
4.5 Test Experiments .......................................................... 41
List of Algorithms

3.1 Monitoring Agent pseudo-code ......................................................... 30
3.2 Extraction and Transformation algorithm .......................................... 33
Listings

3.1 Example of a CSV log file. ......................................................... 35
## Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>CDN</td>
<td>Content Distribution Network</td>
</tr>
<tr>
<td>CPU</td>
<td>Central Processing Unit</td>
</tr>
<tr>
<td>CSV</td>
<td>Comma-Separated Values</td>
</tr>
<tr>
<td>ESM</td>
<td>End System Multicast</td>
</tr>
<tr>
<td>ETL</td>
<td>Extraction, Transformation and Load</td>
</tr>
<tr>
<td>HD</td>
<td>High Definition</td>
</tr>
<tr>
<td>HTML</td>
<td>Hypertext Markup Language</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hypertext Transfer Protocol</td>
</tr>
<tr>
<td>IETF</td>
<td>Internet Engineering Task Force</td>
</tr>
<tr>
<td>IP</td>
<td>Internet Protocol</td>
</tr>
<tr>
<td>IPTV</td>
<td>IP Television</td>
</tr>
<tr>
<td>MIB</td>
<td>Management Information Bases</td>
</tr>
<tr>
<td>MOS</td>
<td>Mean Opinion Score</td>
</tr>
<tr>
<td>MPD</td>
<td>Media Presentation Description</td>
</tr>
<tr>
<td>NAT</td>
<td>Network Address Translation</td>
</tr>
<tr>
<td>P2P</td>
<td>Peer-to-Peer</td>
</tr>
<tr>
<td>PHP</td>
<td>PHP: Hypertext Preprocessor</td>
</tr>
<tr>
<td>PPSP</td>
<td>Peer-to-Peer Streaming Protocol</td>
</tr>
<tr>
<td>QoE</td>
<td>Quality Of Experience</td>
</tr>
<tr>
<td>Acronym</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>---------------------------------</td>
</tr>
<tr>
<td>QoS</td>
<td>Quality Of Service</td>
</tr>
<tr>
<td>RMON</td>
<td>Remote Monitoring</td>
</tr>
<tr>
<td>RTCP</td>
<td>RTP Control Protocol</td>
</tr>
<tr>
<td>RTP</td>
<td>Real-time Transport Protocol</td>
</tr>
<tr>
<td>SNMP</td>
<td>Simple Network Monitoring Protocol</td>
</tr>
<tr>
<td>SVC</td>
<td>Scalable Video Coding</td>
</tr>
<tr>
<td>TCP</td>
<td>Transport Control Protocol</td>
</tr>
<tr>
<td>UDP</td>
<td>User Datagram Protocol</td>
</tr>
<tr>
<td>UI</td>
<td>User Interface</td>
</tr>
<tr>
<td>URL</td>
<td>Uniform Resource Locator</td>
</tr>
<tr>
<td>VoD</td>
<td>Video On Demand</td>
</tr>
</tbody>
</table>
Introduction

Contents

1.1 Motivation and Objectives ........................................ 3
1.2 Contributions ...................................................... 4
1.3 Thesis layout ....................................................... 5
1.1 Motivation and Objectives

In the last years the Internet has been experiencing a tremendous growth. IP video traffic will be 73% of all Internet Protocol (IP) traffic, excluding the amount of video exchanged through Peer-to-Peer (P2P) [1]. Service providers, independent producers, and the end-users themselves can now offer their video contents to the Internet community which makes media streaming very attractive and a potential market.

Media streaming is the process of delivering multimedia contents, such as video and audio, to a group of networked users and can be done using different techniques. The most common are the traditional client-server paradigm or Content Distribution Network (CDN) for web-streaming with adaptive or scalable methods, and P2P-based streaming. The content is typically streamed regarding timing and quality constraints, therefore, an efficient distribution process is required to satisfy the corresponding constraints and application requirements. Video streaming contents can be classified in two groups: on-demand and live. In Video On Demand (VoD), contents are pre-recorded and typically stored in centralized servers, and their size is limited and well known, making VoD similar to a (regular) file transfer process. The content can be delivered in a progressive way (streamed while being watched) or completely downloaded before being watched. The user can navigate the content, with playing and pause functions, as well as rewind or fast-forward. In live video streaming, the content to stream is generated during the execution of the event (e.g., music concert, sport event), making the content available at the “same time” for all the users, and the content cannot be fast-forwarded.

Streaming Media services became recently highly globalized, with users from all over the world not confined anymore to their “home providers” to get access to contents. Large-scale and heterogeneous CDN-based or P2P-based streaming systems, for the distribution of Live and On-demand contents, have emerged to cope with this demand. These systems try to accommodate the temporal and spatial dynamics of the demands, like location and network heterogeneity and the increasing quality of the distributed media, so far up to High Definition (HD) videos but still with several quality constraints. However, most, if not all, of current solutions are closed-source or proprietary in terms of both architecture and streaming protocols used and hardly reach high-quality streaming levels. The Peer-to-Peer Streaming Protocol (PPSP) working group of the Internet Engineering Task Force (IETF) is developing standard P2P streaming protocols aimed to cope with the aforementioned issues. While there is yet no complete implementation using these open protocols, it will be important to monitor and evaluate the scalability and effectiveness of the emerging solutions, starting with the existing prototypes developed in the scope of the European projects SARACEN [2, 3] and P2PNext [4].

Therefore, a monitoring system capable of providing QoS and QoE metrics for this type of solutions, is becoming essential to access their performance and quality, providing adequate data for comparison with existing closed-source systems.
This thesis describes the development of a monitoring system capable of retrieving the most common and adequate metrics to characterize P2P streaming systems that use PPSP protocols, as conceptualized in Figure 1.1.

1.2 Contributions

The contributions of this work are the following:

• A complete State of the Art document on the background of P2P multimedia streaming, overlay topology of existing P2P streaming systems, streaming measurement metrics and the development of standard streaming protocols by IETF;

• The development of a monitoring system to characterize P2P streaming solutions based on the IETF PPSP protocols that allows a comparison with closed-source commercial systems.
1.3 Thesis layout

This thesis is organized as follows: Chapter 1 gives a general introduction to the problem, with the motivations and objectives of the work. Chapter 2 provides background information as well as related works. In Chapter 3 the concept of the monitoring system is introduced, and the architecture of the prototype solution is described together with its implementation process, followed by its evaluation in Chapter 4. Chapter 5 summarizes the contributions, presents the architectural shortcomings and suggests areas for future work.
2

State-of-the-art

Contents

2.1 P2P Multimedia Streaming .................................................. 9
2.2 Overlay Topology ............................................................... 9
2.3 Tree-based P2P Streaming Systems ....................................... 11
2.4 Mesh-based P2P Streaming Systems ..................................... 13
2.5 Hybrid P2P Streaming Systems ............................................ 16
2.6 The P2P Streaming Protocols of IETF ................................. 18
2.7 Streaming Measurement Metrics ......................................... 20
2.8 Related Work ................................................................. 21
This chapter presents the concepts and the State-of-the-Art on P2P Multimedia Streaming Systems and P2P Streaming Measurements. The related work on monitoring systems for this type of streaming paradigm is presented and discussed.

2.1 P2P Multimedia Streaming

A P2P network consists of a group of nodes in the Internet that communicate and cooperate to achieve a common interest, e.g., sharing a file or streaming a media (video, audio). Those P2P nodes are logically connected among them through virtual “links”, implementing a network abstraction over the physical topology, i.e., an overlay network. Hence, each peer maintains a list of other peers that form its neighborhood, and the “overlay manager” is in charge of populating and updating such a list of neighbors. Therefore, peers establish communications with their neighbors, exchanging video contents and other information. Several “logical” entities are the constituents of a P2P network. For example, in file sharing systems some dedicated servers are employed to provide the indexing services used for searching files. Therefore, some control services may be provided through dedicated servers (easier to manage and update), while the exchange of the shared data is performed between peers. Such systems that include special-role centralized entities, for instance indexing servers, are named “hybrid” P2P systems, while “pure” P2P systems consists of only regular distributed peers.

P2P live streaming systems typically possess one or more peer nodes that are in charge of the distributing of an “endless” source media content, following a swarming paradigm, i.e., the source video stream needs to be divided into data segments of a limited size, usually named “chunks” or “pieces”, in order to be shared in the swarm of peers for “simultaneous” viewing by the end-users. Hence, such nodes are called source nodes because they are the actual injection points of the prepared media stream into the P2P network. When a new chunk is generated, it will be received by a few “closest to the source” peers that will then redistribute that fresh chunk to other neighbor peers.

In P2P on-demand streaming, “simultaneous” viewing of the same instant in the content timeline by the end-users is not a requirement. The process of content streaming is similar, in terms of content preparation, but for “stored” contents, i.e., the peers in the swarm request and share any part of the content at any moment in time.

2.2 Overlay Topology

In a P2P network, peers are connected among them in a “logical” manner. Those “virtual connections” are formed “over” the underlying physical topology, creating therefore an “overlay” network. P2P systems are broadly classified in structured and unstructured systems, depending on the delivery tree
that is formed [5]. Tree-based structured systems have the same delivery tree for all the packets (that corresponds to the actual tree overlay). Unstructured overlays dynamically build the delivery tree for each single packet, while the packet is traveling through the overlay.

### 2.2.1 Structured Systems

In structured systems, the overlay management protocol entails a fundamental role, aimed at building a multicast tree structure overlay (Figure 2.1). The content source node is at the root of the tree and the other peers establish parent-child relationships among them. Peers that forward data are called parent nodes, and peers that receive it are called children nodes, i.e., each peer has only one parent node, while “internal” nodes become $k$ degree parents, connecting to a maximum of $k$ children.

![Figure 2.1: Tree-based overlay network [6].](image1)

In these systems, when a peer receives a chunk of data from its parent, it forwards the chunk to its children, making the content to “flow” through the structure, from the root to the leaves of the tree. Although the tree structure is straightforward to implement and efficient, it suffers significantly from network dynamics, as the system must quickly detect the departure of peers, or else all the peers in the subtree that was rooted at the departed nod would experience data starvation. The overlay management protocol is required to take care of these types of issues, which introduces some signaling overhead. “In structured systems the multicast tree overlay is the strong and the weak point at the same time” [6].

### 2.2.2 Unstructured Systems

In unstructured systems, peers are connected in a mesh topology, establishing neighborhood relationships (Figure 2.2) among them. Such connections can be either symmetric or asymmetric, resulting in directed or undirected graphs. Mesh topologies are more resilient to peer departures than tree-based
topologies, because the departure of a peer in this type of topology results in the removal of an edge in the graph that can be replaced by adding another peer (i.e., another edge) to the neighborhood [6].

In these systems, each chunk of data follows a different distribution path, from the source node to each other peer in the network. The routing intelligence resides in the scheduler of the system which uses neighborhood, network information and current state, to perform the content distribution. In these systems peers exchange information among them, such as the fraction of chunks received or missed, the size of the neighborhood, etc. The scheduler uses that information to select the target peers to contact and the chunks to exchange. A “peer selection” algorithm in these systems employs special policies in order to find suitable peers in the neighborhood to contact while a “chunk selection” algorithm chooses the set of chunks to exchange.

2.3 Tree-based P2P Streaming Systems

In tree-based systems, peers are organized to form a tree-shape overlay network where the streaming source plays the role of root node, and multimedia content delivery is typically push-based. Due to their structured nature, tree-based P2P streaming applications guarantee both topology maintenance at very low cost and good performance in terms of scalability and delay. On the other side, they are not very resilient to peer churn, that may be very high in a P2P environment [6]. The main commercial systems that use this type of overlay are described below.
2.3.1 CoolStreaming

CoolStreaming is the commercial implementation of Data-Driven Overlay Network (DONet) [7]. DONet organizes nodes in a dynamic tree, depending on data availability.

The main components of the system are the membership manager (that preserves information about the overlay network and the exchange of buffer maps among peers), the partnership manager (that manages the collaboration between peers), and the scheduler (that schedules the data transmissions towards the suitable partner peers).

The latest implementation of CoolStreaming is no longer a simple tree-based system (further details in Section 2.5).

2.3.2 Conviva

Conviva is a real-time media control system for multimedia broadcasting over the Internet. The End System Multicast (ESM) [8] is the underlying networking technology used to organize and maintain the overlay topology of Conviva.

The ESM maintains a single tree in the overlay and the basic functional components include a bootstrap protocol, a parent selection algorithm, and a light-weight probing protocol used for topology construction and maintenance. A separate control structure decoupled from the tree uses a gossip-like algorithm to allow each member to know a small random subset of group members. Members also responsible to maintain paths from the source.

When a new node joins the system, it gets a subset of the group membership from the root node, and then tries to find parents using a parent selection algorithm. Light-weight probing heuristics are also used by the new node to evaluate other remote nodes from the subset of members it knows, in order to choose candidate parents. Additionally, the new node also uses a parent selection algorithm to deal with performance degradation related with churn in the network.

ESM has a version of the protocol for smaller scale conferencing applications (with multiple sources), and another version for larger scale broadcasting applications using a Single source.

The second ESM prototype is also able to cope with receiver heterogeneity and presence of Network Address Translation (NAT)/firewalls. Actually, the tree is structured in such a way that public hosts use hosts behind NAT/firewalls as parents. The audio stream is kept separated from video stream and multiple bit rate video streams are encoded at source and broadcast in parallel though the overlay tree. The system can dynamically select the most suitable video stream according to receiver bandwidth and network congestion level, because the audio is always prioritized over video streams, and lower quality video is always prioritized over high quality video.
2.4 Mesh-based P2P Streaming Systems

In mesh-based systems, peers are organized in a randomly connected overlay network, and multimedia content delivery is pull-based. This is the reason why these systems are also referred to as “data-driven”. Due to their unstructured nature, mesh-based P2P streaming applications are very resilient with respect to peer churn and guarantee higher network resource utilization than the one associated with tree-based applications. On the other side, the cost to maintain the overlay topology may limit performance in terms of delay, and in terms of pull-based data delivery it calls for large size buffers where to store chunks [6]. The main commercial systems that use this type of overlay are described below.

2.4.1 Octoshape

Octoshape is a P2P plug-in that has been developed by the homonym Danish company and has become popular for being used by CNN\(^1\) to broadcast live streaming content. In Octoshape there is no tracker protocol but only the peer protocol. The information on peers that already joined the channel is transmitted in form of metadata when streaming the live content. Each peer maintains a sort of Address Book with the information necessary to contact other peers that want the same content. In order to mitigate the impact of peer loss, the address book is also used at each peer to derive the so called Standby List, which Octoshape peers use to probe other peers in order to be sure that they can take over if one of the current senders leaves or is congested [9].

2.4.2 PPLive

PPLive was developed in Huazhong University of Science and Technology in 2004, and it is one of the earliest and most popular P2P streaming software in China. The PPLive website reached 50 million visitors for the opening ceremony of Beijing 2008 Olympics, and the dedicated Olympics channel attracted 221 million views in two weeks [10].

The PPLive system includes the following main components:

1. Video Streaming Server, that plays the role of video content source and copes with content coding issues;
2. Peer, also called node or client, is the PPLive entity that downloads video content from other peers and uploads video content to other peers;
3. Channel Server, that provides the list of available channels (live TV or VoD content) to a PPLive, as soon as the peer joins the system;

\(^1\)http://www.cnn.com
4. Tracker Server, that provides a PPLive peer with the list of online peers that are watching the same channel as the one the joining peer is interested in.

As soon as a PPLive peer joins the systems and selects the channel to watch, it retrieves from the tracker server a list of peers that are watching the same channel.

Both peer discovery and chunk distribution processes are controlled by the peer protocol. After retrieving the list of active peers watching a specific channel from the tracker server, a PPLive peer sends out probes to establish active peer connections, and some of those peers may return also their own list of active peers to help the new peer discover more peers in the initial phase. Chunk distribution process is mainly based on buffer map exchange to advertise the availability of cached chunks.

Due to the proprietary nature of PPLive protocols, most of known details have been derived from measurement studies [11] [10].

2.4.3 Zattoo

Zattoo was founded in 2005 and offers 237 channels across Europe in different languages, distributing those channels between countries based upon their IP [12].

Zattoo system includes the following main components:

1. Broadcast server: is in charge of capturing, encoding, encrypting and sending the TV channel to the Zattoo network. A number $N$ of logical sub-streams is derived from the original stream, and packets of the same order in the sub-streams are grouped together into the so-called segments;

2. Authentication server: is the first point of contact for a peer that joins the system. It authenticates Zattoo users and assigns them with a limited lifetime ticket. Then, a user contacts the Tracker server and specifies the TV channel of interest;

3. Tracker Server: provides a Zattoo peer a set of neighboring peers covering the full set of sub-streams in the TV channel.

A measurement study of the Zattoo streaming application was performed in [12].

2.4.4 Tribler

Tribler is a research project funded by multiple European research grants. It follows the BitTorrent model but with several improvements, namely, to support video streaming in two different forms: on-demand and live streaming [13].

Differently from BitTorrent, where a tracker server centrally coordinates peers in uploads/downloads of chunks and peers directly interact with each other only when they actually upload/download chunks
to/from each other, there is no tracker server in Tribler, and as a consequence, there is no need of a tracker protocol.

In Tribler several protocols extending BitTorrent were designed: BuddyCast (that is a gossip protocol used to exchange preference lists containing the info-hashes of downloaded torrents), MegaCache (where up to 2500 preference lists of other peers are stored in an embedded SQLite database). Tribler also introduces permanent identifiers for peers called PermIDs which act as cryptographically secure communication primitives.

2.4.5 QQLive

QQLive is a large-scale video broadcast software developed by Tencent [14]. QQLive adopts CDN and P2P architectures for video distribution and is different from other popular P2P streaming applications. QQLive provides video source by “source servers” and CDN and the video content can be pushed to every region by CDN throughout China. In each region, QQLive adopts P2P technology for video content distribution.

The main protocols in QQLive are the tracker protocol and the peer protocol. These two protocols are proprietary and the exchanged messages are encrypted. The tracker protocol uses User Datagram Protocol (UDP) and the port for the tracker server is fixed. For the video streaming, if the client gets the streaming from CDN, the client uses the Hypertext Transfer Protocol (HTTP) with port 80 and no encryption; if the client gets the streaming from other peers, the client use UDP to transfer the encrypted media streaming and not Real-time Transport Protocol (RTP)/RTP Control Protocol (RTCP). If there are messages or video content missing, the client will take re-transmission and the re-transmission interval is decided by the network condition. The QQLive does not care about the strategy of transmission and chunk selection which is simpler and not similar to BitTorrent because of the CDN support.

A measurement study of the QQLive streaming application was performed in [14].

2.4.6 PPStream

PPStream is a very popular P2P streaming software in China and in many other countries of East Asia [15]. The system architecture of PPStream is very similar to the one of PPLive. There are four types of nodes on the PPStream network: channel list servers, trackers, peer list servers, and media chunk sharers. When a PPStream peer joins the system, it retrieves the list of channels from the channel list server. After selecting the channel to watch, a PPStream peer retrieves from the peer list server the identifiers of peers that are watching the selected channel, and it establishes connections that are used first of all to exchange buffer-maps.

A measurement study of the PPStream streaming application was performed in [15].
2.4.7 SopCast

SopCast maintains a number of channels, each one for a separate live video and transmitting the content over its own independent overlay network. The main components of this streaming system are, a live streaming server, a bootstrap server and a set of clients (peers) [16].

Clients establish partnerships and exchange data among themselves to receive the live streaming video. The media streaming server is a special client which encodes the media and transmits it to other clients directly connected to it. The bootstrap server maintains a centralized record of all clients. When a new client connects to a particular channel, it contacts the bootstrap server, which, in turn, returns a list of clients already receiving the live content. The joining client may establish partnerships with a subset of these clients.

Partnerships may be established and undone dynamically in response to client behavior and according to the quality of the service provided by current partners.

A characterization of the SopCast Overlay Network was performed in [16].

2.4.8 UUSee

UUSee is one of the leading P2P multimedia streaming solution providers in China, with legal contractual rights to most of the channels of CCTV (the official Chinese television broadcaster), and also had the online broadcasting rights to the 2008 Summer Olympics. With a large collection of streaming servers around the world, it simultaneously broadcasts hundreds (over 800) live streaming channels to very wide audiences (millions of peers), with most contents encoded with high quality streaming in mind (around 500 Kbps) [17].

Similar to most state-of-the-art meshed-based P2P streaming protocols, the UUSee streaming protocol design is based on the principle of allowing peers to serve each other by exchanging blocks of data, which are received and cached in their local playback buffers.

Each peer establishes Transport Control Protocol (TCP) or UDP connections with other peers, and buffer availability bitmaps (i.e., buffer maps) are exchanged periodically. During this process, it measures the throughput of the connection, and then selects a number of most suitable peers, from which it actually requests media blocks.

A measurement study of the UUSee streaming application was performed in [18].

2.5 Hybrid P2P Streaming Systems

Hybrid P2P Streaming Systems includes all the P2P applications that cannot be classified as simply mesh-based or tree-based and present characteristics of both mesh-based and tree-based categories.
2.5.1 New Coolstreaming

The New Coolstreaming adopts a hybrid mesh-tree overlay structure and a hybrid pull-push content delivery mechanism [19].

Similarly to the old Coolstreaming, a newly joined node contacts a special bootstrap node and retrieves a partial list of active nodes in the system. The interaction with bootstrap node is the only one related to the tracker protocol. The rest of New Coolstreaming interactions are related to peer protocol.

The newly joined node then establishes a partnership with few active nodes by periodically exchanging information on content availability. Streaming content in New Coolstreaming is divided into equal-sized blocks or chunks, which are unambiguously associated with sequence numbers that represent the playback order.Chunks are then grouped to form multiple sub-streams.

Like in most of P2P streaming applications information on content availability is exchanged in form of buffer-maps.

2.5.2 Splitstream

Splitstream was designed to overcome the unbalanced forwarding load in traditional tree-based multicast systems [20]. It distributes the forwarding load over all the peers respecting the different capacity limits of each individual peer. For that purpose SplitStream splits the multicast stream into multiple stripes, and uses separate multicast trees to distribute each of the stripes.

This mechanism provides a generic infrastructure aimed for high bandwidth content distribution where the Slipstream-based application controls how the content is encoded and divided into stripes.

SplitStream builds the multicast trees for the stripes respecting the bandwidth constraints (inbound and outbound) of the peers. Contents are encoded such that each stripe requires approximately the same bandwidth, and the content can be reconstructed from any subset of the stripes.

2.5.3 Anysee

Anysee is a P2P live streaming system that adopts an inter-overlay optimization scheme, enabling resources to join multiple overlays simultaneously. A public implementation, AnySee v.1.1, was released on June 2004. It has been used to broadcast live-streaming media, including TV programs, movies and the Grid and Cooperative Computing (GCC’04) international conference in Wuhan, to tens of thousands of end users in China Education and Research Network (CERNET) [21].

The AnySee system is comprised of a rendezvous point (RP), a media source, a monitor, and end systems. Each end system contains an IP to Network Coordinates Database (INCD), which is pre-built and integrated into the end system software. The modular system of AnySee comprises several functions (e.g., Buffer Management and HTTP Server). The HTTP server module creates a virtual HTTP
service at a local machine. After retrieving media data packets from the buffer, the HTTP service module sends them to media players, under the HTTP protocol.

### 2.6 The P2P Streaming Protocols of IETF

A set of open core protocols, i.e., Tracker Protocol and Peer Protocol is being developed by IETF to standardize signaling operations in P2P streaming systems. As the existing proprietary protocols for P2P streaming perform similar functions, the consequence is repetitious development efforts for any new system, with the lock-in effects leading to substantial integration difficulties with other players such as CDNs. As an example, the enhancement of existing caches and CDN systems in order to support P2P streaming, requires the integration with a multitude of proprietary protocols, increasing the complexity of the interactions with different P2P streaming applications.

The design of PPSP includes a signaling protocol between trackers and peers (the PPSP Tracker protocol) and a signaling protocol among the peers (the PPSP Peer protocol) that can also be used to the exchange of data, as illustrated in Figure 2.3, enabling peers to receive streaming contents within the time constraints of the media.

The P2P distribution network architecture requires peers able to communicate with a tracker in order to participate in a particular swarm. This centralized tracker service is used for peer bootstrapping and for content registration and location. Content Media Presentation Description (MPD) files for each content, include reference to swarm identifier and the corresponding tracker, allowing the association of
content location information to the swarm of peers streaming the content.

The functional entities related to PPSP are the Client Media Player, the service Portal, the Service Tracker and Peers. The Service Tracker is a logical entity that maintains the lists of PPSP active peers storing and exchanging chunks for a specific content. The tracker answers queries from peers, collects information on the activity of peers, and stores the status of peers to help in the selection of appropriate candidate peers for a requesting peer. The service Portal is a logical entity typically used for client enrollment and content information publishing, searching and retrieval. The Client Media Player provides a direct interface to the end user at the client device, and includes the functions to select, request, decode and render contents. In PPSP the Client Media Player interfaces with the peer using request and response standard formats for HTTP Request and Response messages.

The PPSP request messages can be of the following types:

- **CONNECT**: used when a peer registers in the tracker (or if already registered) to notify it about the participation in named swarm(s);

- **FIND**: used by peers to request to the tracker, whenever needed, a list of peers active in the named swarm;

- **STAT_REPORT**: that allows an active peer to send status (and optionally statistic data) to the tracker to signal continuing activity. This request message is sent periodically to the tracker while the peer is active in the system.

The process used for streaming distribution relies on a chunk transfer scheme whereby the original content is re-encoded using adaptive or scalable techniques and then chopped into small video segments corresponding to a short play-out duration (in the order of a few seconds). With this method the system can support the following streaming mechanisms [22]:

- **Adaptive** – alternate versions of the content with different qualities and bit rates;

- **Scalable description levels** – multiple additive descriptions of the content (i.e., addition of descriptions refine the quality of the video);

- **Scalable layered levels** – nested dependent layers corresponding to several hierarchical levels of quality, i.e., higher enhancement layers refine the quality of the video of lower layers;

- **Scalable multi views** – views correspond to 2D and to stereoscopic 3D videos, with several hierarchical levels of quality.
2.7 Streaming Measurement Metrics

In proprietary systems like those described in previous sections, there is no available information to study streaming characteristics and quality. Typically, performance measurements can only be done by sniffing the network and trying to infer its behavior. It is also not clear what metrics are used by proprietary systems to evaluate their streaming mechanisms. Therefore the most common and well known metrics for streaming measurements should be available in standard protocols like those proposed by IETF.

The delivery of video streaming based on QoS techniques assures packet differentiation and indicates the impact of multimedia traffic on the network performance, but do not reflect the user perception. Therefore, a combined control of QoS with QoE support can assure the distribution of video content according to video content characteristics and the user experience, while optimizing the usage of network resources (Figure 2.4).

2.7.1 QoS Metrics

Network parameters are typical QoS parameters affecting the performance of networked applications. Specification of such QoS parameters usually depend on the context of the involved applications, but the following parameters are considered the basic form of QoS as other forms can be mapped to them [23]:

- **Throughput**: refers to the amount of data that can be transferred from a source to a destination per time unit (bit per second);
- **Delay**: is the time interval between data departure and arrival;
- **Packet jitter**: refers to packet delay variation;
- **Packet Loss**: corresponds to data packets not reaching their destination.

Due to the transfer mechanism considered in PPSP-based implementations (a chunk transfer scheme) and since the end-user experience is affected by the aforementioned parameters, but cannot be directly correlated to it, only **Throughput** will be considered for the work in this thesis.

**QoE** metrics will describe end-user experience because massive packet loss or delay in the network will cause streaming freezes or jumps (typically, chunks not arriving in-time are not considered for decoding and not played out).

2.7.2 QoE Metrics

When it comes to **QoE** assessment methods a distinction is usually made between subjective and objective methods. In subjective experiments a number of people are usually asked to watch a set
of video clips for later on to rate what they have seen and experienced. The rating is often done in accordance with the Mean Opinion Score (MOS).

Objective quality measurements are not dependent on test users’ opinions, but are rather based on direct measurements of some process or outcome of user behavior. Objective measurements are typically technology centric where data is automatically collected by monitoring tools.

To investigate the streaming quality of peer-assisted live streaming systems the following main metric were proposed \[24\] \[25\]:

- **Start-up Time**: defined as the time interval from when one channel is selected until actual playback starts on the screen;
- **Re-buffering frequency**: refers to the number of freezing events during a time interval;
- **Mean duration of a re-buffering event**;
- **Bit rate**: refers to the quality level available and received by the end-user;

An overview of QoS and QoE for video streaming is illustrated in Figure 2.4.

2.8 Related Work

There are several research works which measure and monitor P2P live streaming systems. Among these measurements, there are mainly three types of data collection methodologies:

- **Passive Monitor**
- **Active Crawler**
- **Log Collection**

The characteristics of these three methods and related research are described below.
2.8.1 Passive Monitor

The passive approach uses devices to watch the traffic as it passes by. These devices can be special purpose devices such as a “Sniffer”, or they can be built into other devices such as routers, switches or end node hosts. Examples of such built in techniques include Remote Monitoring (RMON), Simple Network Monitoring Protocol (SNMP) and Netflow capable devices. The passive monitoring devices are polled periodically and information is collected to assess network performance and status (in the case of SNMP devices the data is extract from Management Information Bases (MIB)).

The passive approach does not increases the traffic on the network for the measurements. It also measures real traffic. However, the polling required to collect the data and the traps and alarms all generate network traffic, which can be substantial. Further the amount of data gathered can be substantial especially if flow analysis is being performed or if trying to capture information on all packets.

The passive approach is extremely valuable in network trouble-shooting, however they are limited in their ability to emulate error scenarios or isolate the exact fault location.

Several research works were developed through this method. During the 2006 FIFAWorld Cup, an extensive measurement was performed, on network traffic generated by the most common P2P IP Television (IPTV) applications, namely PPLive, PPSStream, SOPCast, an TVAnts [11]. They extracted several statistics, which help in having a better understanding of the behavior of P2P IPTV systems.

A passive measurement of two commercial P2P video streaming systems [26], has concluded that P2P live streaming has a more significant impact on network bandwidth utilization and control than P2P file transfer.

A framework was proposed, to analyze the traffic that P2P applications generate [27]. The selected metrics allowed to analyze some of the most popular P2P application nowadays, highlighting their main similarities and differences.

2.8.2 Active Crawler

An Active Crawler is able to take full advantage of the partial membership list maintained by each peer, and recursively query online nodes in a P2P system [24] [28].

Typically, a crawler registers itself as a peer in the system, requests a set of partners from the tracking server, and then adds them into the crawling list. For each node in the crawling list, the crawler requests for its status information and partner list, and then merges the returned partner list into the crawling list. Existing literature has shown that a good crawler can find 95% of peers for a channel within 5 seconds [24].

Though the active crawler is able to monitor a much larger set of peers than the passive monitor, it cannot obtain the detailed status of a peer, as only a few bytes of information are allowed to be
exchanged between each other due to overhead concerns.

The two techniques were also combined to provide reasonably accurate estimates of ongoing video playback quality throughout the network [28].

2.8.3 Log Collection

The Log Collection method requires that a dedicated log server is placed in the system, and each peer periodically reports its activities, such as user behavior events and internal status, to the log server.

In [19] the information from a peer is compacted into several parameters of the Uniform Resource Locator (URL) string. The log server stores the reports received from peers into a log file. In the log file, each log entry corresponds to a HTTP request URL string, referred as a “log string”, that contains various data blocks, which are formed in “name=value” pairs and are separated by “&”. Reports from peers are further divided into two classes, i.e., activity reports, which are sent out immediately when the corresponding activities take place, indicate the peer activities such as joining and departure, and status reports, which are sent out every 5 minutes, indicate the internal state of peers.

To inspect the run-time behavior of UUSee P2P streaming, detailed measurement and reporting capabilities were implemented within its P2P client software [18]. Each peer collects a set of its vital statistics, and reports them to dedicated trace servers every 5 minutes via UDP. The statistics include the IP address(es), the “channel” the end user is watching, the current buffer map, the number of consecutive blocks in its current playback buffer (i.e., the buffering level), its instantaneous aggregate sending and receiving throughput to and from all its partners, as well as a list of all its neighbor partners, with their IP addresses, TCP/UDP ports, and sending/receiving throughput to/from each partner. As a streaming server in UUSee utilizes a P2P protocol similar the one used by regular peers, it reports its related statistics periodically as well.

The authors in [29–31] propose an agent-based real-time monitoring system for large scale P2P video streaming platforms, which provides traffic, user receiving quality and system information through web-based presentation. The monitoring agents are installed in both servers and end-users to collect information and send the information to the real-time monitoring system. The system has a Log server that collects statistics and notification logs from the monitoring agents, parsing the information carried in the logs and stored them into database. In the meanwhile, the data in the database will be aggregated for web-based output.
3 Monitoring System Architecture

Contents

3.1 Design Requirements .................................................. 27
3.2 Monitoring System Design ............................................. 29
3.3 Development Process and Implementation ......................... 34
P2P live streaming service is an up and coming Internet application which is reshaping the network traffic profiles. Live streaming systems making use of P2P paradigm break the limitation of the server computational power and network bandwidth bottleneck of traditional client-server based architectures which cost much higher in terms of deploying servers and service backbone bandwidth. However, the dynamic nature of P2P networks, where peers can appear and disappear whenever they want, brings reliability and performance issues into the system.

The PPSP working group of IETF is developing open-source protocols that allows adaptive and scalable video streaming over P2P architectures. While there is no complete implementation of these open protocols, a monitoring system capable of providing QoS and QoE metrics for this type of solutions, is becoming essential to access their performance and quality, providing adequate data for comparison with existing closed-source systems.

In order to provide information about P2P streaming applications using the PPSP, a log collection methodology has been chosen for the solution described in this Chapter.

We describe the stages followed in the development and implementation of the Monitoring System prototype. A brief overview of the programming languages, tools and libraries that were used is also presented, as well as the development environments and the solutions used for building the various modules.

3.1 Design Requirements

The main goal for this work is to create a system, capable to provide, not only the traffic, system information or user status notifications, but also several metrics regarding the content quality experienced by the end-user. The following set of functional requirements were considered for the solution:

Traffic Statistics: In this monitoring system, traffic statistics are important in measuring QoS of multimedia streaming. For Serving Nodes and Peer Nodes, the traffic statistic contains the incoming and outgoing traffic from other Serving Nodes and Peer Nodes.

System Information Statistics: System information statistics includes the current system status information of a device, a Serving Node or Peer Node, such as Central Processing Unit (CPU) utilization and memory consumption.

Content Quality Measurement Statistics: The content quality measurements statistics includes the number of re-buffer events and their duration, start-up time and receiving quality for the allowed streaming mechanisms. The information provided about streaming mechanisms, within the log interval, should identify the maximum and average bit rate of an adaptive stream and the maximum layer of a stream encoded
with Scalable Video Coding (SVC). Regarding 3D streaming, the log report should identify the number of received views and their quality levels. This information is closer to the user’s service experience and reflects user’s player status and therefore plays an important role in the QoE measurement.

The following log classification scheme was adopted, in order to adapt the monitoring system to the P2P distribution system developed by the PPSP working group of IETF.

![Log Classification Diagram](image)

**Figure 3.1: Log Classification.**

One vital non-functional requirement was also considered in order to achieve the aforementioned functional requirements:

**Scalability:** The monitoring system should not interfere with the network and streaming conditions in the presence of large number of peers.
3.2 Monitoring System Design

The Monitoring System architecture considers Monitoring Agents, a Log Server, a Database Server and a Web Server as illustrated in Figure 3.2. To dynamically monitor the entire streaming system, detailed log capabilities are implemented within the client applications and transferred to the Log Server. After a parser procedure, the measurement data is loaded into a data warehouse for near real-time analysis.

![Monitoring System Architecture](image)

**Figure 3.2:** Monitoring System Architecture.

The client system consists in the implementation of the PPSP protocols reference code, both as a Client System (network node with a Peer Agent, a Media Player and the Monitoring Agent) and a Tracker System (network node running the Tracker Process and the Monitoring Agent).

3.2.1 Monitoring Agents

The Monitoring Agents are responsible for collecting and parsing the logs from the P2P nodes. The generated data is stored containing information of one minute of log and the report is sent to the Monitoring Server periodically, with a configurable time interval that aggregates all the generated files within that period. The communication between these two entities relies on HTTP POST request messages and response as illustrated in Figure 3.3.
The Monitoring Agent in the Client System is responsible for parsing the logs generated by the Peer Agent and Player Emulator.

**Algorithm 3.1: Monitoring Agent pseudo-code**

```plaintext
begin
    logInterval ← inputLogInterval
    startTime ← GetDate()
    nextMinute ← GetNextMinute()
    minuteCount ← 0
    while startTime ≠ nextMinute do
        ParseAgentLog()
        ParsePlayerLog()
        GetCpuAgent()
        GetMemoryAgent()
        GetCpuPlayer()
        GetMemoryPlayer()
        minuteCount ← minuteCount + 1
        if minuteCount = logInterval then
            AggregateLogs()
            SendLog()
            minuteCount ← 0
            startTime ← GetDate()
            nextMinute ← GetNextMinute()
```

CPU consumption and memory utilization of the Monitoring Agent and the two components of the Client System that are also computed and parsed.

The routines implemented by the Monitoring Agent read and parse the generated logs in periods of one minute, and are described in Algorithm 3.1.
3.2.2 Log Server

The log server is responsible for receiving all the statistics reports and to handle the files posted to the server by the Monitoring Agents, validating and storing them. This storage area is continuously waiting for new files transferred from all the Monitoring Agents in the P2P streaming system, and calls the the Extraction, Transformation and Load (ETL) procedures from the Database Server when new files arrive into the server.

3.2.3 Database Server

The database server should facilitate the presentation of data in the monitoring web interface, and the most common way to achieve this purpose is to use a Relational Database with the appropriate constraints. Although the presentation of monitoring data is the main objective, reporting capabilities usually provided by data warehouses are also possible options in the design of this server. Data warehouses based, for example, in “star” schemas with “fact” and “dimension” tables, have redundant information to obtain small time response to complex queries.

In the design of this monitoring system prototype, these two options were considered and the one that fits better to the main purpose of presenting near real-time data is the relational database, but taking advantage of ETL process, usually adopted for data warehouses.

The data comes from flat log files generated by Peers and is stored in the database after the ETL process. This process includes three steps:

- **Extract stage**: the data is parsed into the desired format for transforming;
- **Transform stage**: applies a series of rules to the extracted data from the source such as data aggregation;
- **Loading stage**: stores data into end target.

These steps have the objective of transforming received information into stable, valid, consistent and periodically updated data, capable of being used for reporting, data visualization and further analysis of streaming conditions.

For the database, the data generated by the streaming system was previously analyzed in order to build a Logic Entity-Relationship (E-R) model. Entity-relationship model is an abstract and conceptual representation of the data in the P2P streaming system.

Each rectangle shown in the diagram of Figure 3.5, corresponds to an entity. A relationship, which is a diamond in the figure, describes the entities related to the other entities. The content requested by a client, “End-user”, represent a “Service” which can be Live or On-Demand. The service is available for consuming at the “Media Server Peer”.
Figure 3.4: Database UML model.

Figure 3.5: E-R Model of the database.
The analysis on the logs provided by the reference implementation of PPSP protocols have shown that not all the metrics defined in section 3.1 are available (e.g., Connection Type) in that implementation. However, there are other metrics available, that allow the developers to have a deeper understanding on the behavior of the protocols. These metrics are also stored in the database.

### 3.2.4 Extraction and Transformation Module

This module is responsible for extracting and transform the information stored in the Log Server. The original data in the Log Server is not deleted after the extraction and transformation stages. In case of data loss or corruption in the database, a valid backup of all incoming data is available and ready to be re-processed. The Algorithm 3.2 summarizes the processing of these two stages.

**Algorithm 3.2: Extraction and Transformation algorithm**

```plaintext
begin
  logDirectory ← inputLogDirectory
  for Each logFile in logDirectory do
    getConnectionInfo()
    getNeighbours()
    getPeerMode()
    getOverlay()
    getActions()
    getUpTraffic()
    getDownTraffic()
    getBandwith()
    getChunks()
    getCPU()
    getMemory()
    getPlayerReport()
    getStartdelay()
  saveBackupFile()
end
```

### 3.2.5 Load Module

The Load Module reads the temporary files created in the last stage of the Extraction and Transformation Module and loads the data to the corresponding tables.

### 3.2.6 Web Server

The web server is responsible for presenting measurement data to an administrator or system operator. Therefore, the server has a Web User Interface (UI) engine, capable of retrieving data from the database, and generate charts and tables in Hypertext Markup Language (HTML) format for presentation to the user.
3.3 Development Process and Implementation

The development process of the Monitoring System prototype was divided into the following stages:

- Technology Research and Related Works
- Requirements Gathering and Analysis
- Design of the Architecture
- Implementation Process
- Testing and Functional Validation

The P2P streaming solution uses the reference code for both peer and tracker, developed in the scope of the European projects SARACEN [2] [3] and P2PNext [4].

The main component developed in those projects was the library libppsp, illustrated in Figure 3.6. That library is composed by several core modules and has the possibility to be extended by plugins.

The **Logger** is the key core component, regarding the implementation of the Monitoring System, because it provides the logging features, essential for development and debugging of libppsp.

![Figure 3.6: Libppsp components.](image-url)
The prototype of the Monitoring System was developed keeping in mind the scalability of the solution in the presence of large number of peers in the P2P streaming network. The Monitoring Agents placed in the Client System must automate the log parsing and transfer the information over the network without interfering with the streaming conditions. Therefore, the programming language chosen for this task was Perl, due to its powerful file manipulation and string parsing capabilities, combined with shell scripts. The result of the parsing procedures is a Comma-Separated Values (CSV) file containing the metrics in Figure 3.1. When the configured log interval is reached, all the generated CSV are aggregated and transferred to the Monitoring Serving within an HTTP request POST message. An extract of content of a CSV log file is depicted in Listing 3.1.

Listing 3.1: Example of a CSV log file.

| UPTRAFFIC | 2015-04-25T19:55:51.000 | PEER 01;1234 |
| INFO | PEER 01; 198.18.13.20; 8070 |
| CPU | PEER 01; 2015-04-25T19:55:57.000; 3.2; 0.2; 2.9 |
| PPSPTP | 2015-04-25T19:54:01.281; PEER 03; C; JOIN; SUCCESSFUL |
| DOWNLOAD | 2015-04-25T19:54:28.310; PEER 03; /Chunk 2D-L2-16; SEEDER 02; 394920; 6902; 0.074 |

In the database server, the ETL process is automated by several Perl scripts that store data in a MySQL database. PHP: Hypertext Preprocessor (PHP) scripts are used to handle uploaded files to the Log Server and query the database.

Based on the E-R model, the tables to store data in the database were created, where the peer entity, defined by a Peer ID, IP address and Port number, has a central role. In this schema, the peer entity represents both Media Server and Peer Nodes because both have several measurement metrics in common. A combination of several Perl scripts process the data transferred from all nodes in the P2P streaming system, and for each table in the database, a temporary file aggregates the transformed data, ready for the loading stage. In order to handle the files posted to the server by the Monitoring Agents, a PHP script is used to validate and store the files.

The Web interface of the Monitoring System uses a combination of Javascript and PHP scripts to get information from database and jQuery Flot library of JavaScript for plotting dynamic charts and tables. This server has been implemented with a default configuration of Apache2. The programming languages used were PHP and JavaScript. The UI of the monitoring system is illustrated in Figure 3.7.

The Monitoring System was developed on a 64-bit Linux OS (Ubuntu 12.04 LTS) hosted on a machine with 8Gb of memory and an Intel i7 processor with four Cores of 1.8Ghz.
Figure 3.7: User interface of the Monitoring system
4 Evaluation

Contents

4.1 Evaluation Objectives and Experimental Scenarios ......................... 39
4.2 Analysis and Evaluation of Results ........................................ 41
This chapter describes and presents the evaluation of the Monitoring System prototype. A description of the evaluation objectives, metrics and experimental scenarios are also detailed.

4.1 Evaluation Objectives and Experimental Scenarios

The P2P streaming network was implemented in laboratory using the Common Open Research Emulator (CORE)\(^1\), an emulator that allows the deployment of network topologies running in real-time. Each host on the network is a virtual machine that runs a "simplified" version of a Linux operating system, capable of running the P2P network entities, such as Trackers and Peers (including the Peer Agent and the Media Player Emulator) incorporating the Monitoring Agent. CORE [32] consists of a GUI for drawing topologies of lightweight virtual machines, and Python modules for scripting network emulation. The emulated network is connected to a live network where the Monitoring Server is located.

The components of the Monitoring Server are external to the CORE emulator. The CORE emulated network provides connection to external networks, allowing all elements in the emulated network, to communicate with an external Monitoring Server.

The type of network links in the P2P network are described in Table 4.1.

Table 4.1: Network Link Profiles

<table>
<thead>
<tr>
<th>Class</th>
<th>Network Profile</th>
<th>Down-link</th>
<th>Up-link</th>
<th>Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Ethernet</td>
<td>100 Mbps</td>
<td>100 Mbps</td>
<td>5 ms</td>
</tr>
<tr>
<td>1</td>
<td>high-end cable</td>
<td>100 Mbps</td>
<td>8 Mbps</td>
<td>5 ms</td>
</tr>
<tr>
<td>2</td>
<td>high-end 3G/4G</td>
<td>24 Mbps</td>
<td>5.8 Mbps</td>
<td>20 ms</td>
</tr>
<tr>
<td>3</td>
<td>high-end DSL2+</td>
<td>12 Mbps</td>
<td>1 Mbps</td>
<td>5 ms</td>
</tr>
<tr>
<td>4</td>
<td>mid-range DSL2</td>
<td>6 Mbps</td>
<td>768 Kbps</td>
<td>10 ms</td>
</tr>
<tr>
<td>5</td>
<td>mid-range DSLLite</td>
<td>1 Mbps</td>
<td>512 Kbps</td>
<td>15 ms</td>
</tr>
<tr>
<td>6</td>
<td>low-end 3G</td>
<td>1 Mbps</td>
<td>256 Kbps</td>
<td>40 ms</td>
</tr>
<tr>
<td>7</td>
<td>low-end DSL</td>
<td>512 Kbps</td>
<td>128 Kbps</td>
<td>20 ms</td>
</tr>
</tbody>
</table>

For the test scenarios the same video content was used, and its characteristics are described in Tables 4.2 and 4.3.

Table 4.2: Video Characteristics

<table>
<thead>
<tr>
<th>Type of Streaming</th>
<th>Number of Layers</th>
<th>Bandwidth Min. Layer</th>
<th>Bandwidth Max. Layer</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVC</td>
<td>4</td>
<td>0.25 Mbps</td>
<td>2 Mbps</td>
<td>80 seconds</td>
</tr>
</tbody>
</table>

Table 4.3: SVC Layers Size

<table>
<thead>
<tr>
<th>Layer</th>
<th>File Min. Size</th>
<th>File Avg. Size</th>
<th>File Max. Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>L0</td>
<td>28140 bytes</td>
<td>73027 bytes</td>
<td>119251 bytes</td>
</tr>
<tr>
<td>L1</td>
<td>45935 bytes</td>
<td>131314 bytes</td>
<td>213735 bytes</td>
</tr>
<tr>
<td>L2</td>
<td>121506 bytes</td>
<td>351068 bytes</td>
<td>560592 bytes</td>
</tr>
<tr>
<td>L3</td>
<td>234691 bytes</td>
<td>673065 bytes</td>
<td>1155548 bytes</td>
</tr>
</tbody>
</table>

\(^1\)http://cs.ltd.nrl.navy.mil/work/core/
The P2P test scenario is illustrated in Fig. 4.1, and the objective is to evaluate the ability to collect the values from the metrics defined in section 3.1 and listed in Table 4.4, measuring the impact on network and streaming conditions. Several experiments with different number of peers and network links capacity and latency, were setup in order to evaluate the solution.

Along with peer connection information and actions of the aforementioned P2P system, QoS and QoE metrics were also evaluated in the experiments that are representatives of a real-world streaming network, as described in Table 4.5.

Table 4.4: Metrics Descriptions

<table>
<thead>
<tr>
<th>Metric Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Notifications</td>
<td></td>
</tr>
<tr>
<td>Actions</td>
<td>Join and leave a swarm. Send statistics report to Tracker</td>
</tr>
<tr>
<td>Peer Mode</td>
<td>Availability of content in Peer. When peer has 100% it becomes a Seeder</td>
</tr>
<tr>
<td>System Metrics</td>
<td></td>
</tr>
<tr>
<td>CPU Load</td>
<td>CPU load by minute</td>
</tr>
<tr>
<td>Memory Consumption</td>
<td>Memory consumption by minute</td>
</tr>
<tr>
<td>QoS Metrics</td>
<td></td>
</tr>
<tr>
<td>Throughput</td>
<td>Traffic from/to Media Server and other Peers</td>
</tr>
<tr>
<td>QoE Metrics</td>
<td></td>
</tr>
<tr>
<td>Start-up Time</td>
<td>Download time of consumed layers for the first chunk plus read delay</td>
</tr>
<tr>
<td>Re-buffer Events</td>
<td>Periods were next chunk is not downloaded in time for presentation</td>
</tr>
<tr>
<td>Downloaded Layers</td>
<td>Maximum and average downloaded layer in time</td>
</tr>
</tbody>
</table>

In all the experiments, peers start joining the streaming “swarm” 70 seconds after trackers and media serving peers have boot up, at a cadence of one each second. All test experiments differ only in the

\[\text{Re-used from the experimental P2P laboratory setup of the SARACEN project [3]}\]
Table 4.5: Test Experiments

<table>
<thead>
<tr>
<th>Test Number</th>
<th># of Trackers</th>
<th># of Peers</th>
<th># of Seeders</th>
<th>Peers Network Class</th>
<th>Seeders Network Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>8</td>
<td>1</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>8</td>
<td>1</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>32</td>
<td>2</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>32</td>
<td>2</td>
<td>7</td>
<td>1</td>
</tr>
</tbody>
</table>

number of peers present in the “swarm”, and in their uplink and downlink capacity.

The first experiment considers only one Peer, one Peer Seeder (content source) and one Tracker, and is used as reference for the evaluation of the Monitoring System capabilities. Test experiments 2 and 3, represent a small size streaming network with only eight peers, one seeder and different link configurations. In these experiments Peers leave the “swarm” after receiving the full the content, and the idea is to measure the performance of PPSP and Monitoring System in scenarios with heterogeneous peers (having, different uplink and downlink capacities) with some churn. The objective in test experiments 4 and 5 are similar but for a larger streaming network.

The tests were designed to measure the scalability of the system, regarding the streaming quality and network disturbance. Since the original logs are available for post-processing analysis, the Content Quality metrics values collected will be compared with or without the Monitoring Agents in the system, in order to evaluate the introduced overhead.

The host system where the Monitoring System and the P2P network (emulated) were setup is a 64-bit Linux OS (Ubuntu 12.04 LTS) with 8 GB of RAM and an Intel i7 processor with four Cores running at 1,8 Ghz.

4.2 Analysis and Evaluation of Results

In this section, the results from the experiments are presented and grouped by the metrics defined in Table 4.4. A comparative study of QoE metrics with or without the presence of Monitoring Agents in the system is also presented.

4.2.1 Peers Actions

This type of notification allows to monitor peer actions such as, join a swarm, send statistics to tracker an leave a swarm. It is possible to observe content availability in each peer. Figure 4.2 illustrates peer actions in test experiment 1 (a single Peer consuming the content from a single Peer seeder).

It can be observed in Figure 4.2 that the peer joins the “swarm” and starts receiving content, sending statistics reports in periods of 30 seconds. When the streaming of the content ends, this interval returns to the default period of 10 seconds (keep-alive signal), showing that the peer stayed in the swarm after
receiving all chunks from the seeder.

In peer mode notifications (Figure 4.3), the peer sends to the monitoring system, the percentage of available chunks from the streaming content. This type of notification is only possible in VoD contents where the duration of the streaming is well known.

4.2.2 System Metrics

**Memory Utilization:** The memory utilization of PPSP peer agent, player and monitoring agent are negligible (a maximum of around 8 MB), as illustrated in Figure 4.4(a) for test experiment 1. In terms of memory utilization, the monitoring agent, and also the player and peer agent, exhibit a fairly good performance.

**CPU consumption:** The CPU load of the PPSP peer agent, player and monitoring agent is illustrated in Figure 4.4(b) for test experiment 1. The average CPU load of the Monitoring Agent is only 4%, and lower than the Peer Agent which is the process with a central role in the streaming process. In terms of CPU load the Monitoring Agents also exhibit a good performance.
4.2.3 QoS Metrics

For traffic statistics, the Monitoring Agent collects upload and download values for each peer and media server in the “swarm”. They represent a cumulative sum of download and upload traffic of all peers for each time step of the streaming process. Figure 4.5 illustrate the upload traffic, for test experiment 2. Figure 4.6 illustrate download traffic, for test experiment 5.

It can be observed from the data that, after the initial period where peers get the content from the media server, they start to share the content. Less than 1 minute after the start of the streaming, peers continue to exchange content and the downloaded traffic from the media serving peer (seeder) decreases significantly, meaning that, at least one peer has downloaded all the content and is sharing it with the others.

Regarding the overhead traffic generated by the Monitoring reports, the average size of reports is 190 KB for 1 minute of log. Taking into account that peers are prepared to wait a random period, between 1 and 10 seconds, to send the report, the traffic is effectively spread in time.

4.2.4 QoE Metrics

Start-Up Delay: The time elapsed, between the client requesting the content and the first frame displayed by the player is computed as the “start-up delay” of the streaming. This means that the cumulative downloading time of all consumed layers at first chunk, plus the read delay of the
Figure 4.4: System Metrics
Figure 4.5: QoS Metrics: Upload Traffic

Figure 4.6: QoS Metrics: Download Traffic
player, represent the start-up delay for each peer in a given “swarm”.

In test experiment 2, all the the peers have good network conditions and the “start-up delay” is less than 1 second with and without the Monitoring Agent in the system, as illustrated in Figure 4.7. This is a small scale streaming system, and both the Monitoring Agent and PPSP protocols show good performance.

In test experiment 3, when peers have low bandwidth links, the traffic and computation overhead starts to have impact in the “start-up delay”, as illustrated in Figure 4.8. However the increase in the delay is not significant since it remains in values near 1 second.

In test experiment 4, all the peers have good network conditions and the start-up delay remains low for the majority of the peers, with and without the Monitoring Agent in the system, as illustrated in Figure 4.9. Just a few peers show a large “start-up delay”, and this is caused by the existence of only one Media Serving peer in the network. All peers compete to get the initial content almost at the same time and this leads to resource starvation.

This is a large scale streaming system, and both the Monitoring Agent and PPSP protocols show an overall good performance.

In test experiment 5, when peers have low bandwidth links the traffic and computation overhead starts to have impact in start-up delay, as illustrated in Figure 4.10. However, just one peer had a large increase in the delay, showing that the Monitoring Agent has a small impact in the initial streaming conditions.

Re-buffer Events: The SVC video content is streamed in chunks of 2 seconds duration (video playout). This means that, in order to avoid a freezing image, the next full media chunk (comprised of 1 or more quality layer chunks) must be received in less than 2 seconds. Therefore, the metric is computed by finding the number of periods in chunk arrivals that are superior to this interval, and also its average duration. For test experiment 2, where peers have considerable bandwidth available, the system shows no re-buffer events with and without the Monitoring Agents in the system. This means a good experience for the end-user and a good performance of the PPSP protocols and no interference caused by the Monitoring System.

In test experiment 3, when peers have low bandwidth links, the traffic and computation overhead start to have impact and re-buffer events occur for some peers. In this experiment only one re-buffer event was registered occasionally during the streaming of the content, and the average duration becomes larger with the Monitoring Agent in the system, as illustrated in Figure 4.11. As the download and upload capacities of peers are quite limited, this condition may partially explain these re-buffer events, when the required bandwidth is fully occupied by the media stream and the monitoring traffics.
Figure 4.7: Start-Up time for Test 2.
Figure 4.8: Start-Up time for Test 3.
Figure 4.9: Start-Up time for Test 4.
Figure 4.10: Start-Up time for Test 5.
Figure 4.11: Re-buffer Events for Test 3.
In test experiment 4, several peers experience re-buffer events with and without Monitoring Agents in the system, as illustrated in Figure 4.12. Since the scenario has only one Media Serving peer, the upload capacity of peers is not enough to serve the content requests.

Test experiment 5 shows that almost every peer experienced re-buffer events due to low bandwidth links. Peers that are in a different Autonomous System, from the Seeders and the majority of Peers, experienced more events and with superior average duration. The presence of Monitoring Agents have no overall impact, since only one peer manifested more and with larger average duration of re-buffer events, as illustrated in Figure 4.13.

**Downloaded Layers:** The number of quality layers received by peers can be correlated to a good quality of experience by the end-user. For test experiment 2, as illustrated in Figure 4.14, peers receive the maximum layers of quality for the overall streaming duration. The presence of the Monitoring Agent has no significant impact.

In test experiment 3, peers continue to receive the maximum quality layers possible for the bandwidth constraint, during a considerable period of time, with and without the Monitoring Agent, as illustrated in Figure 4.15. However, the decrease in received quality layers is noticeable, caused by low bandwidth links of the peers.

In test experiments 4 and 5, the quality layers consumed by peers decreases and the Monitoring Agent starts to have impact in the streaming conditions. However the profile remains similar, showing that the available bandwidth of peers links, is the factor that influences the most the streaming experience.

### 4.2.5 Summary

The designed test experiments were representative of real-world network conditions, and lead to a comprehensive study of PPSP protocols behavior under different scenarios. The monitoring system proved to be capable of gathering all the defined metrics with low impact in the overall streaming and network conditions.
Figure 4.12: Re-buffer Events for Test 4.
Figure 4.13: Re-buffer Events for Test 5.
Figure 4.14: Downloaded Layers for Test 2.
Figure 4.15: Downloaded Layers for Test 3.
Figure 4.16: Downloaded Layers for Test 4.
Figure 4.17: Downloaded Layers for Test 5.
Conclusion and Future Work

Contents

5.1 Conclusions ................................................................. 61
5.2 System Limitations and Future Work ............................... 61
5.1 Conclusions

Nowadays, media streaming services are becoming globalized and represent over 70% of the Internet traffic. Several commercial P2P streaming systems have emerged but most, if not all, of current solutions are closed-source or proprietary in terms of both architecture and streaming protocols used and hardly reach high-quality streaming levels.

The PPSP working group of the IETF is developing standard P2P streaming protocols aimed to cope with the aforementioned issues, and while there is no complete implementation of solutions using these open protocols. It becomes essential to develop a monitoring system capable of providing QoS and QoE metrics for this type of solutions, to access their performance and quality, providing adequate data for comparison with existing closed-source systems.

This thesis presented a monitoring system for P2P streaming systems based on PPSP protocols of IETF and the system architecture respected functional and non-functional requirements. The log classification adopted is capable of describing this streaming solutions regarding, host System, QoS and QoE metrics, providing the necessary statistics for comparison with other commercial systems.

The evaluation of the implemented prototype demonstrated a system capable of retrieving the proposed metrics, with Monitoring Agents implemented in Peers, Trackers and Media Serving nodes. The addition of this prototype to the streaming solution proved to have a small impact in streaming and network conditions.

The monitored information is stored in back-end database and will be valuable to diagnose the streaming system afterwards.

5.2 System Limitations and Future Work

Although the evaluation of the proposed solution shows that it is possible to implement a system with the enumerated requirements in large scale streaming networks, there are still unsolved aspects that should be addressed in future works. In this section we will thoroughly list some of these future functionalities.

• **Security and Authentication** – The Monitoring Agents should have a secure registration mechanism in the Monitoring server;

• **Scalability** – The Monitoring Agents could be embedded into the PPSP Agents (core libppsp) taking advantage of the plugins system available;

• **Reporting** – The database could be optimized and a data warehouse should be implemented in order to provide quick and helpful information in presence of a large scale streaming system.
Bibliography


