Multi-Criteria Optimization of Content Delivery within the Future Media Internet

Joachim Bruneau-Queyreix

THÈSE

PRÉSENTÉE À

L'UNIVERSITÉ DE BORDEAUX

École Doctorale de Mathématiques et d'Informatique

par **Joachim BRUNEAU-QUEYREIX**

Pour obtenir le grade de

DOCTEUR

Spécialité : Informatique

Optimisation Multi-Critères pour la Diffusion Vidéo au sein de l'Internet Media du Futur

---

Multi-Criteria Optimization of Content Delivery within the Future Media Internet

---

Soutenue le 21 November 2017

Devant le jury:

Laurent Réveillère	Pr, Université de Bordeaux	Président
Hermann Hellwagner .	Pr, Université de Klagenfürt	Rapporteur
François Taiani	Pr, Université de Rennes	Rapporteur
Jordi Mongay Battala	MCF, Université de Varsovie	Examinateur
Christian Timmerer	MCF, Université de de Klagenfürt, CIO Bitmovin	Examinateur
Yiping Chen	Dr, Chef de Projet, Orange R&D Paris	Examinateur
Daniel Négru	MCF, Bordeaux INP	Directeur
Baptiste De Meerler ..	Chef de Projet, Viotech Communications	Invité

Titre Optimisation Multi-Critères pour la Diffusion Vidéo au sein de l'Internet Media du Futur

Résumé Les solutions de streaming vidéo adaptatives basées sur l'utilisation du protocol HTTP ont été largement plébiscitées dans les mondes de l'industrie et de la recherche, notamment pour les possibilités d'améliorations de qualité d'experience qu'elles offrent ainsi que pour leurs facilités de déploiement liées au protocol HTTP. Pour autant, bien que ces solutions permettent d'augmenter la qualité d'experience utilisateurs en diminuant la qualité de la vidéo transmise sur les réseaux pour minimiser les interruptions vidéo liées au temps de chargement, la qualité intrinsèque de la vidéo est limitée par les capacités physiques du chemin entre le serveur utilisé et le client. Dans l'objectif d'augmenter la qualité d'experience utilisateurs et de diminuer les couts de déploiements des services de streaming, les travaux de cette thèse de doctorat proposent de faire évoluer de façon pragmatique les solutions de streaming adaptatives actuelles vers l'utilisation en simultané de plusieurs sources (serveurs ou pairs).

La première contribution de cette thèse présente MS-Stream, une technique évolutive de streaming adaptatif basé sur HTTP et utilisant plusieurs serveurs simultanément. MS-Stream offre la possibilité d'exploiter la bande passante disponible dans les infrastructures distribuées et les réseaux hétérogènes. La deuxieme contribution de ce document est MATHIAS, un groupe d'algorithmes d'adaptation centrés client, implémentés dans MS-Stream, qui a pour vocation d'optimiser l'utilisation des ressources réseau hétérogènes mises à disposition du client pour obtenir une qualité vidéo cible. MATHIAS permet à chaque client de controller le nombre de serveur utilisé en simultané, de faire face à l'hétérogeneité des resources disponibles, de réagir aux fluctuations soudaines et non-anticipées des capacités des serveurs tout en donnant à l'utilisateur une experience de streaming ininterrompu. Pour finir, nous allons plus loin dans les capacités de scalabilité et de qualité d'experience de MS-Stream et MATHIAS en tirant profit des ressources physiques des consommateurs. Nous proposons une solution hybride pair-à-pair/multi-server de streaming adaptative: PMS. Au sein de PMS, les logiques d'adaptation de la qualité vidéo et de la scalabilité sont distribuées pour permettre à chaque client de tendre vers une utilisation optimale de l'infrastructure de streaming.

Mots-clés Streaming Video, Multi-source streaming, QoE, P2P, Adaptation, Systemes distribués

Title Multi-Criteria Optimization of Content Delivery within the Future Media Internet

Abstract Single-source HTTP Adaptive Streaming solutions (HAS) have become the de-facto solutions to deliver video over the Internet mostly due to their capabilities to increase end-user's Quality of Experience (QoE) as well as their ease of deployment due to the usage of the HTTP protocol. Although HAS solutions can increase QoE by trading off the delivered video quality to minimize the number of video freezing events, they are limited by the bandwidth available on the considered communication channel between the client and the server.

This thesis exposes our contributions in building lightweight pragmatic and evolving solutions advocating for the simultaneous usage of multiple sources with heterogeneous capacities so as to achieve high QoE content delivery at low cost. The first contribution of this work presents a streaming solution extending HAS capabilities to a pragmatic multi-server technique: *MS-Stream*. MS-Stream provides the means to exploit expanded bandwidth and link diversity in distributed heterogeneous network infrastructures. In our second contribution, we propose *MATHIAS*, a client-side two-phase consumption and adaptation algorithm implemented into MS-Stream. MATHIAS aims at increasing the end-user's perceived streaming quality while utilizing the most of the heterogeneous capacities offered at the service and network environments. Finally, we further extend the QoE and scalability capabilities of MS-Stream and MATHIAS by leveraging on clients' connectivity capacities and we expose our third contribution: a hybrid P2P/Multi-server live-Streaming system (*PMS*) incorporating distributed quality and scalability adaptation mechanisms.

Keywords Streaming Video, Multi-source streaming, QoE, P2P, Adaptation, Distributed systems

Laboratoire d'accueil LaBRI (Laboratoire Bordelais de Recherche en Informatique) UMR 5800 - 251 Cours de la Libération, 33405 Talence

Remerciements

Bien que la couverture de ce manuscript ne comporte que mon nom en tant qu'auteur des travaux décrits, il s'agit là, en réalité, du résultat de l'investissement de plusieurs personnes. C'est pourquoi je tiens très sincèrement à remercier tous ceux qui ont contribué à l'aboutissement de ces recherches que ce soit par un soutien moral, technique, scientifique, financier, amical ou familial.

Contents

List of Figures

List of Tables

Acronyms

AIAD	Additive Increase and Additive Increase
AIMD	Additive Increase and Multiplicative Decrease
AP	Access Point
API	Application Programming Interface
AVC	Advanced Video Coding
AVC	Advanced Video Coding
CDN	Content Delivery Network
CS2P	Cross Session Stateful Predictor
DASH	Dynamic Adaptive Streaming over HTTP
DASH-IF	DASH Industry Forum
DHT	Distributed Hash Table
FTP	File Transfer Protocol
GoP	Group of Pictures
GP	Greedy Policy
HAS	HTTP Adaptive Streaming
HTTP	Hyper Text Transfer Protocol
ILP	Integer Linear Programming
IP	Internet Protocol
ISP	Internet Service Provider
MATHIAS	Multiple-source and Adaptive sTreamIng AlgorithmS
MPD	Media Presentation Description
MPEG	Moving Picture Experts Group
MS-Stream	Multiple-Source Adaptive Streaming over HTTP
NAT	Network Address Translators
NP	Network Profiles
OTT	Over-The-Top
P2P	Peer-to-Peer

PANDA	Probe ANd Adapt
PMS	P2P/Multi-Server streaming system
PSNR	Peak Signal to Noise Ratio
QoE	Quality of Experience
QoS	Quality of Service
RTCP	Real-time Transport Control Protocol
RTP	Real-time Transport Protocol
RTSP	Real-time Transport Streaming Protocol
RTT	Round-Trip Time
SAND	Server and Network-assisted DASH
SCTP	Stream Control Transmission Protocol
SDN	Software Defined Networking
SLA	Service Level Agreement
SSL	Secure Sockets Layer
SVC	Scalable Video Coding
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
URL	Uniform Resource Locator
VBR	Variable Bitrate
VoD	Video on Demand
WebRTC	Web Real-Time Communication
WiFi	Wireless Fidelity
WL	Water Levelling
XML	Extensible Markup Language

Chapter 1

Introduction

"Begin at the beginning," the King said gravely, "and go on till you come to the end: then stop"

— Lewis Carroll, Alice in Wonderland

1.1 Motivation

With the advent of technology characterized by the increasing offers of video streaming services over the Internet (IPTV, video conferencing, video-on-demand -VoD-, live, and mobile streaming), Quality of Experience (QoE) has become a crucial factor directly defining the success or failure of emerging services. According to several studies related to the usage of the Internet [Cisco, 2016; Sandvine, 2015b,a], video traffic is witnessing an explosive growth. Estimations predict that Internet video traffic including TV, VoD, live and P2P video streaming will have a near two-fold increase by 2020, accounting for more than 82% of all Internet traffic. This poses powerful challenges on the quality and scalability of the offered services. Hence, video traffic is driving next-generation service provider network designs.

Given the global trend of increasing video traffic, issues related to the underlying network architectures, as well as data transfer within network boundaries, Internet video streaming has gathered considerable attention from industry and academia. In general, video streaming services are provided using two different streaming methods that rely on the IP protocol: IPTV streaming, and Over-The-Top (OTT) video streaming. The former utilizes a managed delivery network where Quality of Service (QoS) is guaranteed end-to-end from the streaming provider to the consumer based on a Service Level Agreement (SLA) while the latter uses best-effort content delivery,

which is usually performed via the Internet and quality is not guaranteed[Timmerer and Begen, 2014].

The last few years witnessed tremendous deployments of OTT video streaming systems. These deployments are based on a variety of architectures, including: Cloud platforms and Content Distribution Networks (CDN) such as CloudStream [2017], Netflix [2017] [Adhikari *et al.*, 2012c], Youtube [2017] [Hoßfeld *et al.*, 2013], Twitch [2017] or even more recently Periscope [2017]; Peer-to-peer (P2P) overlays such as, PPLive [Huang, 2008], Zattoo; and hybrid solutions combining P2P and CDN content delivery such as Xunlei Kankan [2017] [Zhang *et al.*, 2015a], LiveSky [Yin *et al.*, 2009, 2010] and Vivendi with its multimedia streaming offerings including Canal+ and Dailymotion [Streamroot, 2017].

In CDN and cloud-based architectures, distributed replica servers are positioned as close as possible to the consuming clients. When accessing a content, consuming clients are automatically redirected to one of the best available servers based on proximity so as to temper network congestion and achieve higher throughput. In P2P solutions, peers are considering obtaining content from several neighboring peers simultaneously so as to achieve high throughput while contributing to the rest of the community by engaging in P2P data transfer. Consequently, the P2P approach provides cost-effective video streaming solutions with enhanced scalability capabilities whereas CDN and cloud-based solutions enable to support video streaming with enhanced performance at the cost of the deployed reliable infrastructure. The hybrid P2P-CDN streaming solutions constitute viable alternatives to the latter architectures as they offer a trade-off between the utilization of the deployed infrastructure and the streaming performance.

To a large extent these deployments realize what traditional IPTV streaming and multicasting architectures have struggled to achieve: large-scale distribution of video-on-demand and live video streaming services to a massive number of end-users distributed over the Internet.

In OTT video streaming, end-users consuming video typically rely upon different devices, ranging from smartphones to tablets and PCs, and also undergo different network connectivity conditions on the basis of their location: at home, in office, on the road. Therefore, in addition to the absence of relationship or agreement between the streaming service providers and the involved network operators, both the delivery network and the end-user devices are unmanaged and have heterogeneous capacities. For these reasons, improving the consumers' QoE is more challenging compared to that in IPTV. The use of different adaptation techniques has become a common

approach to tackle the latter issue. As pointed out by Mok *et al.* [2012], a higher QoE cannot always result from simply playing out videos at higher average bitrates.

OTT streaming services are characterized by the diversity of applications and transport protocols (e.g., RTSP, RTP, UDP, TCP, STCP, FTP, and HTTP). The HTTP protocol has rapidly and widely become the de-facto protocol for web content delivery over the Internet. For this reaseon, and since HTTP packets can easily travel across firewalls, Network Address Translators (NAT) and proxies without restrictions, video streaming over HTTP/TCP has been adopted as a cost-efficient streaming solution.

To counteract fluctuating network conditions and also to cope with the heterogeneity of resources, a new delivery streaming framework termed HTTP Adaptive Streaming (HAS) has been introduced. The HAS technique is now adopted in most of the CDN and Cloud-based streaming architectures for its potential to improve end-users' QoE and to better utilize content and network resources. HAS techniques use flexible bitrate adaptation to deliver the highest video quality possible and to minimize video freezing events, hence increasing the end-users' QoE. In a typical pull-based HAS use-case, a video player located at the client side is in charge of sequentially downloading video segments from a single server hosting several qualities -termed representations- of the same content. The client dynamically adjusts the requested content bitrate according to the current network context including available bandwidth, local conditions such as the duration of buffered video content, and device capabilities including screen resolution and device type. Therefore, opposite to the traditional RTP/UDP-based streaming techniques, the quality adaptation mechanism is located at the client side. Several proprietary solutions have first flourished, ranging from Smooth Streaming by Microsoft [Zambelli, A., 2009; Microsoft, 2017] to HTTP Live Streaming by Apple [Pantos, 2015], HTTP Dynamic Streaming by Adobe [Adobe, 2010] and Akamai's solutions [Akamai]. They have in turn triggered the release of the Dynamic Adaptive Streaming over HTTP (DASH) standard [ISO/IEC MPEG, 2014; Sodagar, 2011], which is currently the most widespread and internationally recognized approach for video streaming. In DASH, the "adaptive" term refers to the capability of the technique to modify the video quality as well as the video transmission itself. The ultimate goal of pull-based adaptive streaming techniques, such as HAS, is to rely on the client-centric approach in order to maximize QoE and to optimize network utilization, which are contradictory objectives and not easily achieved simultaneously.

1.2 Research challenges

Distributed content delivery infrastructures are currently extensively used for the reliable delivery of video content over the Internet at large-scale. These solutions enable to allocate storage and bandwidth to multiple distributed servers, and clients are retrieving content from a single geographically-close server. Most of the time, HTTP Adaptive Streaming (HAS) techniques are employed so as to reach the highest possible end-users' QoE. Consequently, they tend to avoid the video freezing events that are mainly caused by the lack of throughput at the client or at the content delivery network side. However, due to expensive network resources in the CDN, the achievable network throughput for video content delivery is still limited. When bandwidth capacity bottlenecks occur at the content delivery network side, horizontally scaling infrastructures can provide greater network throughput and with path diversity, which result in higher QoS and QoE for the end-users. Nevertheless, in view of the drastic video traffic growth forecasts, the deployment and maintenance costs covered by content delivery service providers will rise and eventually make their services highly pricey for content providers and consumers seeking high QoE.

This thesis addresses the challenge of improving the QoE potential of HAS solutions employed with content delivery networks so as to mitigate the issue of bandwidth capacity bottleneck at the content delivery network side while to optimizing the infrastructure utilization. Consequently, multiple-server HAS is the key feature for enabling clients to effectively make use of distributed network resources. As pointed by the study of Adhikari *et al.* [2012c] on understanding and improving multi-CDN delivery, the QoE and network utilization would greatly benefit from the advent of practical HAS solutions that can actually utilize multiple servers simultaneously.

Hybrid Peer-to-Peer(P2P)/CDN streaming solutions rely on the utilization of CDN infrastructures and on the contribution of each consuming peer to exchange data with their neighbors. Therefore, hybrid P2P/CDN solutions permit to lower scalability costs by relying on P2P traffic to offload the CDN utilization while keeping reliable services along with the deployed distributed server infrastructure. However, mixing HAS technologies along with P2P has not yet been explored for QoE improvement [Anjum *et al.*, 2017]. Therefore, the second challenge adressed in this thesis relates to further evolve HAS capabilities to P2P/multi-source HTTP adaptive streaming so as to benefit from three advantages: reduced scalability costs, reliable streaming services, and improved QoE potential.

The central concepts to achieve these goals are content delivery adaptation and content quality adaptation in distributed systems as well as QoE awareness. Content delivery adaptation describes the spectrum of possible methods to adapt how content is delivered from the source(s) to the end-users whereas QoE awareness implies that quality and delivery adaptations are performed in a way that maximizes the perceived quality. Content quality adaptation in distributed systems explicitely requires local and global optimization methods to adapt the quality (codecs, profiles, bitrate, resolution) of the content itself.

1.3 Research goals

The research goals adressed in this thesis directly derive from the described challenges. The main objective consists in the design, conception, realization, and evaluation of a multiple-source HTTP adaptive streaming solution leveraging on the distributed resources at the content delivery network side and at client side to achieve simultaneously high QoE and reduced scalability costs. Three subgoals are thusly derived from this objective:

Research Goal 1: *Investigating multiple-source HTTP adaptive video streaming.*
This first goal is to investigate if multiple-source adaptive streaming can effectively improve the achieved QoE of streaming services in a pragmatic way. We aim at proposing a framework for multiple-source HTTP adaptive streaming that enables client-centric decisions to adapt the content quality and the content delivery means. Evaluation in a controlled environment is required to understand how this framework can impact the current solutions for multimedia delivery over the Internet in terms of QoE, network bandwidth consumption and processing power.

Research Goal 2: *Investigating the QoE-aware adaptation mechanisms in multiple-source streaming.*
Deriving from the proposed framework, the second goal of this thesis is to investigate the mechanisms involved in increasing the end-users QoE. To that end, we proposed a set of adaptation algorithms to be implemented at the client side. Real-world experiments assessing QoE permit to qualify and quantify the significance of the proposed algorithms.

Research Goal 3: *Investigating hybrid P2P/multi-source adaptive stream-*

ing, its content quality adaptation and network resource utilization.
The last goal is to investigate hybrid P2P/multi-server adaptive streaming by relying on the proposed multiple-source HTTP adaptive streaming framework along with the suggested QoE-aware adaptation algorithms. The ultimate goal is to explore the field of content quality adaptation and the self-scaling properties of a hybrid P2P/multi-server streaming solution based on the two above-cited research goals. Real-world experiments with multiple clients located in different cities in France allowed a large-scale validation of the proposed distributed content quality adaptation and scale adaptation regarding the achievable QoE gains and scalability cost reduction.

The research work of this thesis has been presented in several international conferences and journals. As a recognized technical outcome of this thesis work, we have developed an online demonstrator (available at http://msstream.net [MS-Stream, 2017]), which was awarded several prizes in international conference demonstration sessions.

1.4 Roadmap of the Thesis

The road map for the rest of thesis is outlined below:

Chapter 2 provides the background on the video streaming technologies used during the past three decades, including HTTP adaptive streaming. The exposed related work extensively reviews content quality adaptation and content delivery adaptation proposals found in the literature and in the industry. Hybrid P2P/CDN streaming techniques are also explored, with a special focus on the methods to improve QoE or QoS.

Chapter 3 proposes *MS-Stream*, a solution to enable the support of multiple-source streaming in HAS techniques. We present the core components of our proposal to independently use multiple sources with heterogeneous capabilities. We briefly overview the spectrum of possible adaptations opened by our approach before conducting QoE and system evaluations in a controlled environment. The contribution on MS-Stream was demonstrated in two European projects [DELTA, 2015; DISEDAN, 2016].

Chapter 4 exposes *MATHIAS*, our solution for improving the QoE of end-

users in the context of MS-Stream. MATHIAS is a set of QoE-aware algorithms aiming at utilizing the most of the resources made available to the MS-Stream client to reach a target video bitrate while achieving as few video stalls as possible. We deployed a testbed over the Internet at the premises of several european project partners (universities and companies from DELTA [2015] and DISEDAN [2016] projects) to evaluate the QoE gains of MATHIAS.

Chapter 5 presents *PMS*, a P2P/Multi-Server video streaming system proposal relying on MS-Stream and MATHIAS to provide the highest possible QoE and to optimize the resources made available at the server infrastructure. A national testbed with 300 peers was used to assess PMS's performance.

Chapter 6 concludes and overviews the perspectives of the work conducted in this thesis.

Appendix A lists the outcomes of this thesis in terms of publications in International peer-reviewed journals and conferences, and also includes a list of European project reports.

Chapter 2

Background and related work

"Study the past if you would define the future"

— Confucius

2.1 Background on video streaming evolution

In this section, we shortly review the widely used video streaming technologies during the past three decades. Figure 2.1 illustrates the emergence of the most used streaming protocols on a time scale.

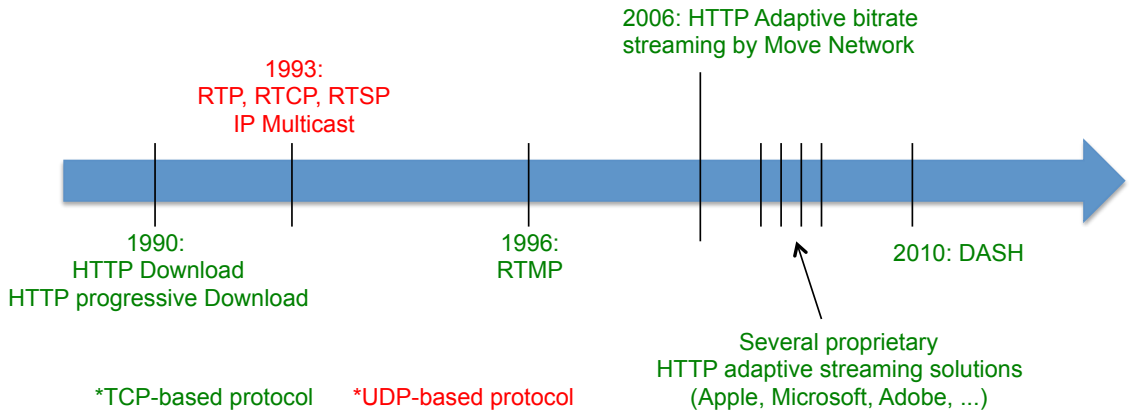

Figure 2.1 – A bit of history of video streaming

First of all, a digital video is the consecutive sequence of video pictures usually displayed at a rate between 24 and 30 frames per second thus providing the illusion of actual motion.

Due to the large volume of data that videos represent, compression algorithms have been employed to lower the size of these types of files. Most video compression algorithms aim at reducing redundancy in video data by combining spatial image compression and temporal motion compensation. In most approaches, three types of encoded video frames exist (depicted in Figure 2.2): Intra/Key (I) frames that contain all the necessary information to decode and display the original frame and can consequently be very large; predicted (P) and bidirectional (B) frames, much smaller than I frames because intra and inter-frame compression is used for size reduction [Varma, 2015]. A Group of Pictures (GoP) is the consecutive serie of an I frame followed by several B and P frames until the next I frame. The variability in encoded bits per second leads to Variable Bit Rate (VBR) video, as defined by the ITU (International Telecommunication Union) H.264 and MPEG-4/AVC video coding standards [Schwarz *et al.*, 2007].

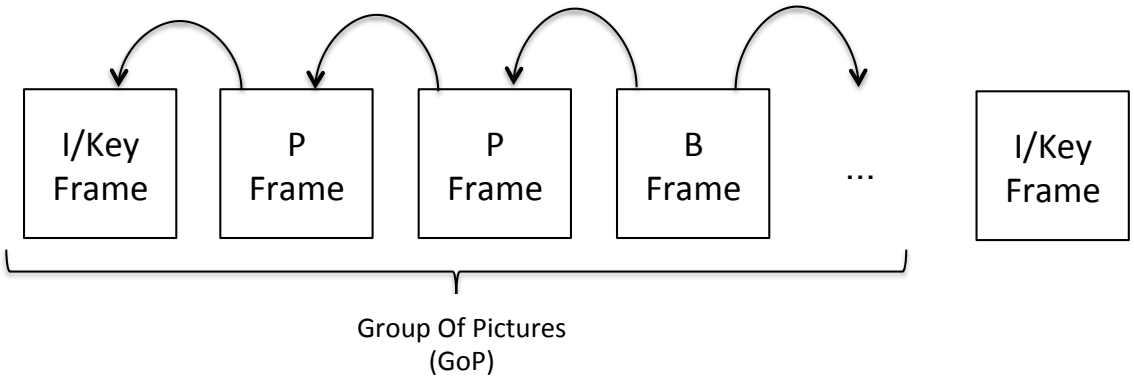

Figure 2.2 – Sequence of video frames

2.1.1 Traditional streaming

The Internet was not originally designed for the sustained delivery of modern bandwidth-intensive applications such as high quality multimedia streaming. The fundamental difference between regular Internet traffic and video traffic is the real-time constraints of video traffic.

Most of the early work on packet video transmission focused on providing real-time transmission with techniques that support resource reservations and Quality of Service (QoS), such as Resource ReSerVation Protocol (RSVP) [Braden *et al.*, 1997] and Integrated Services (IntServ) [Bernet *et al.*, 2000]. Other protocols such as Real-time Transport Protocol (RTP) [Schulzrinne *et al.*, 2003], Real Time Streaming Pro-

tocol (RTSP) [Schulzrinne *et al.*, 1998], Session Description Protocol (SDP) [Handley and Jacobson, 1998], Real Time Control Protocol(RTCP) [Friedman *et al.*, 2003] were developed over the years in order to support real-time streaming over UDP and to control the end systems that support video streams. However, these techniques have issues in traversing NATs and firewalls. They also require dedicated services and network infrastructures, thus increasing deployment and operating costs.

TCP is a reliable protocol which guarantees the delivery of data. However, this reliability comes at the expense of variable delay as senders wait for acknowledgments before continuing sending packets and retransmiting lost packets. Since video is often delay intolerant and does not need high reliability to be acceptable, TCP was initially assumed unsuitable for multimedia delivery [Varma, 2015].

2.1.2 Unicast and Multicast Streaming

In the early 1990s, only a small number of users were able to enjoy video streaming on the Internet. A single streaming server was in charge of delivering all video requests, and unicast connections were established between the clients and the server. As the population of end-users consuming video streaming services drastically increased, the limited scalability of the unicast approach was rapidly reached. Subsequently, multicast protocols have been proposed to be more scalable under large number of consuming clients. IP multicast [Deering and Cheriton, 1990; Sahasrabuddhe and Mukherjee, 2000; Diot *et al.*, 2000] is an extension of the IP protocol, with the objective of providing efficient multipoint packet delivery. Given that the network topology is best known in the network layer, multicast routing associated to this layer is also the most efficient.

A common use of IP multicast is for Internet Protocol television (IPTV) applications. Although IPTV uses the IP protocol, it is not limited to television delivered over the Internet. IPTV is widely deployed in subscriber-based networks with high-speed access channels at end-user premises via set-top boxes or other customer-premises equipment. IPTV is also used for the delivery of content in corporate and private networks.

One major challenge for video multicast is the heterogeneity of end-user devices. With multicast, it can be difficult to find a suitable video bitrate fitting different hardware capabilities and network resources of multiple clients simultaneously. Although multicast seems an attractive solution for the delivery of video content, it was not largely adopted by the streaming actors [Quinn and Almeroth, 2001]. In-

deed, forwarding multicast traffic imposes a great deal of protocol complexity on network service providers. Moreover, core network infrastructures are particularly vulnerable to denial-of-service attacks along with IP multicast.

2.1.3 Traditional Adaptive Streaming

The Internet is the interconnection of multiple networks with best-effort traffic, therefore there is no guaranteed bandwidth for the real-time delivery of video packets. If the network bandwidth is not sufficient to support the video bitrate, then the video decoder at the client side consumes the video at a greater speed than the delivery rate of the data. Thus, the streaming client eventually runs out of video data to decode, which in turn results in screen freezes (video stalls or rebuffering events). In order to avoid such events without having to introduce costly and complex bandwidth reservation mechanisms, adaptive streaming solutions have been used to try to match the video bitrate to the available network bandwidth [Varma, 2015]:

- Using a playout buffer embedded at the client side to pre-fetch data and store it locally in order to absorb the short-term variations of network throughput.
- On-the-fly video transcoding at server-side or in the network in order to adjust the bitrate (or resolution, frame rate, compression ratio) of the requested video to match the network capacities. This solution has a very high processing footprint and requires complex hardware support.
- Layered video coding [Schwarz *et al.*, 2007] to allow the encoding of a video into multiple dependent layers: a unique base layer (representing the least quality level) and several enhancement layers that improve the viewing quality. Hence, the encoded video can be adapted on-the-fly by adding or removing layers to the delivered content. However, such solutions require specialized servers and encoding scheme.
- Stream switching adaptation is a widely employed technique and is also the simplest to implement. The original video content is encoded offline in multiple different bitrates, resulting in multiple versions of the same content. A client-side adaptive algorithm is then used to select the most appropriate video bitrate according to the varying network conditions during transmission. These solutions do not require specialized servers, use the least processing power and provide high scalability due to the client-centric adaptation logic. However,

more storage and finer granularity of encoded bitrates are required to enable the client to optimize the quality adaptation process.

Considering simplicity of implementation and deployment, the playout buffers and client-centric stream switching solutions were widely adopted in the industry.

Multicast streaming solutions also exploited adaptive bitrate techniques [Cable Television Laboratories, 2016] that can be classified into three main categories: single stream approaches, replicated stream approaches, and layered stream approaches. In the single stream approach, a single video stream is transmitted to the multicast group and feedback is received from all clients participating in the group. In the replicated stream approach, the same video is replicated in multiple streams (each with different bitrates) and the client can join a stream that fits its capability. In the layered stream approach, the server sends the video stream in multiple layers and each client can then subscribe to a subset of layers that fits its hardware capabilities and available network throughput.

2.1.4 HTTP progressive download and adaptive streaming

In the beginning of the 2000s, TCP was identified as an interesting candidate for delay-tolerant video transmission. An application layer playout buffer was introduced to absorb the rate fluctuations of TCP. The first implementations of video streaming over HTTP/TCP are called HTTP progressive download. In this scheme, the client simply downloads the entire video file with constant video quality as fast as TCP allows and starts the video playback as soon as enough video data are delivered. One major drawback of this technique is that all end-users receive the same video quality regardless of the heterogeneous network connections and capabilities of the end-users' devices. This can rapidly cause unwanted interruptions in the video playout if the clients' network connections do not reach the video bitrate.

The mid-2000s witnessed the rise of many proprietary HTTP Adaptive Streaming (HAS) solutions. Typically, HAS solutions rely on client-centric stream switching with an embedded buffer at the client side, while using the HTTP/TCP protocol for content delivery. Hence, the client is able to request different video qualities to match the requested bitrate to the varying network conditions. The most notable differences between HAS and traditional streaming protocols lie in the fact that HAS is built on top of TCP instead of UDP, and that HAS clients request and receive video data in terms of video segments (containing few seconds of video playback) instead of continuous streams of video packets. Although some of the above-presented video

streaming technologies are still in use, the video streaming industry has now adopted HAS as the main solution for video streaming over the Internet.

2.2 HTTP Adaptive Streaming and DASH framework

In the literature, many surveys reviewed the framework of most HTTP Adaptive Streaming solutions and more specifically, the Dynamic Adaptive Streaming over HTTP standard -i.e., DASH, also know as MPEG-DASH- [ISO/IEC MPEG, 2014; Kua *et al.*, 2017; Sodagar, 2011; Diallo *et al.*, 2013; Sani *et al.*, 2017; Garcia *et al.*, 2014].

2.2.1 Architecture overview

In a DASH-based solution, a video content is encoded into multiple versions -termed representations- at different video bitrates. Each encoded video is then chunked into small video units called segments, each containing a few seconds of video playback. Segments from one bitrate are aligned in the video timeline to the segments from other bitrates so that the client can smoothly switch bitrates, if necessary, at the segment boundary. The DASH standard does not impose the way the content is delivered to the client. Many architectures exist for DASH-based solutions (client-centric, server-centric, etc...). Figure 2.3 represents a simple and common client-centric DASH-based client/server architecture.

Content information such as video profiles, bitrates, resolutions, codecs, metadata, mimeType, server IP addresses, and segment URLs are described in the associated XML Media Presentation Description (MPD) files handed out prior to the streaming session. The MPD describes a piece of video content within a specific duration as a *period*. In a period, there are multiple types of content available for adaptation such as video, audio and subtitles. They are referred to as *adaptation set*. In an adaptation set, there are multiple versions of the content, each known as a *representation*, each containing multiple segments (video segments for the case of a video adaptation set). Figure 2.4 illustrates the structure of the MPD file. URLs pointing to the video segments in a MPD can either be explicitly described or be constructed via a template (client deriving a valid URL for each segment at a given quality). The format of MPEG-DASH video segment is derived from the MPEG

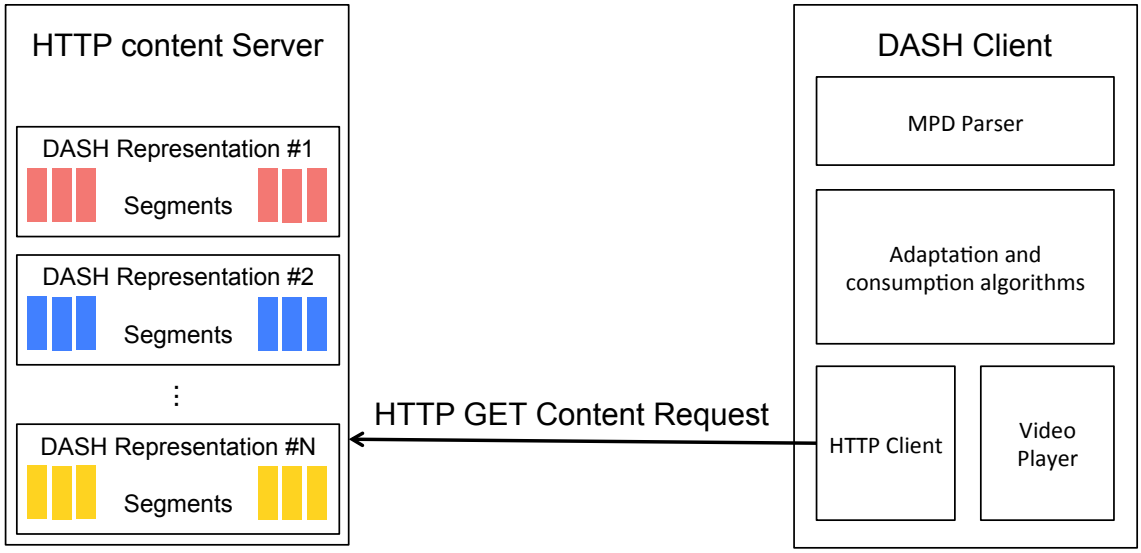

Figure 2.3 – A common client-centric DASH-based client/server architecture

ISO Base Media File Format (ISOBMFF) [ISO/IEC, 2017b, 2012] container and MPEG-2 Transport Stream (MPEG-TS) [ISO/IEC, 2017a].

In each representation, there is a single initialisation segment containing meta data, and many video segments. Concatenating the initialisation segment with regular video segments results in a continuous video stream. Video segments are served to clients by using the HTTP protocol.

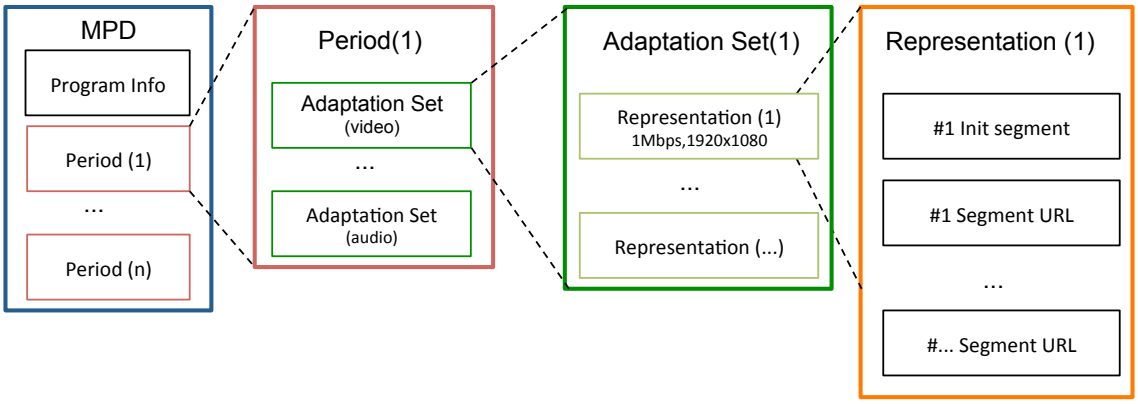

Figure 2.4 – High level overview of the structure of a DASH Media Presentation Description (MPD)

Unlike traditional streaming strategies, the DASH standard does not enforce any specific implementations, adaptation mechanisms or segment scheduling policies. In its most basic form (and also the most widely implemented form), when a DASH streaming session starts, the client obtains and parses the MPD file associated with

the requested content, and starts requesting video segments (typically in sequential order) as fast as possible to fill the playout buffer. Then, the player enters in a steady state where it periodically downloads new segments according to the implemented adaptation logic. In the steady state, the player is in the ON state when it is downloading a segment, and in the OFF state otherwise, resulting in an alternating ON-OFF traffic pattern [Akhshabi *et al.*, 2012a].The client typically keeps a few segments in the buffer to maintain video playback.

In order to select a suitable video bitrate for the next segment to be downloaded, the video player uses various feedback signals observed for each segment.In a typical scenario, the achieved network throughput is used as a criterion for bitrate selection decisions. For example, if the available network throughput is elevated, the DASH client selects a higher video bitrate to provide better QoE to the end-user. On the other hand, if the throughput drops, the client dynamically switches to a lower video bitrate in order to avoid buffer starvation and video freezing event that would cause major degradation of the end-user's QoE. An example of such a HAS streaming session is depicted in Figure 2.5. A "good" adaptation and consumption algorithm is expected to smoothly adapt the video bitrate to provide better QoE [Tian and Liu, 2012, 2016].

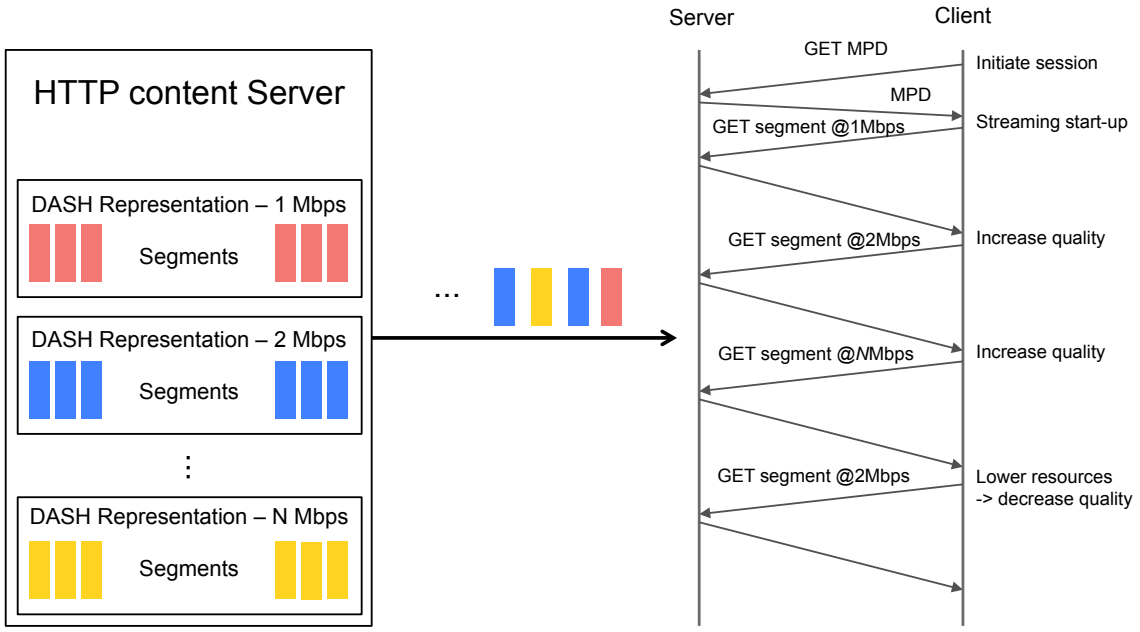

Figure 2.5 – Example of client-centric HTTP Adaptive Streaming session

By using HTTP on top of TCP, DASH (and HAS) has several advantages:

- Clients use the standard HTTP protocol used for web traffic, which provides ubiquitous access of streaming video services on the Internet (through proxies, NAT and firewalls) [Popa *et al.*, 2010].
- DASH servers can be commodity web servers, which significantly reduces the operating costs and allows the deployment of caches to improve performances and reduce the network load.
- A client requests each video segment independently and maintains the playback session state, hence servers are stateless. Maintaining session state at the client means clients can retrieve video segments from multiple servers, hence providing the means to balance the load of requests among servers [Liu *et al.*, 2012b,c].
- TCP reliability and inter-flow friendliness improve the likelihood that streaming traffic consumes a fair fraction of the network bandwidth when competing with other non-video traffic.

These advantages enable streaming service providers to leverage existing and significantly cheaper HTTP infrastructures. Proprietary commercial systems such as Microsoft's Smooth Streaming [Microsoft, 2017], Adobe's HTTP Dynamic Streaming (HDS) [Adobe, 2010] or Apple's HTTP Live Streaming (HLS) [Pantos, 2015] leverage existing CDNs and proxy caches. Throughout this thesis manuscript, the term "*bitrate*" refers to the bitrate of the video. The terms "*bandwidth*" and "*throughput*" both relate to the capacity of the network or of a device (server or client) to transmit/receive data.

2.2.2 Standardization

Move Networks was the first industrial actor on HAS and patented their technology [Major and Hurst, 2014] on adaptive streaming at the United States Patent and Trademark Office (USPTO) in 2010. The patent covers the structure of video content and the intelligent requests sent by clients to adapt video bitrate over IP networks. DASH is based on the work in 3GPP Release 9 [3GPP, 2012] and Open IPTV Forum Release 2 [OIPF, 2014]. MPEG-DASH was first a Draft International Standard in January 2011, and became an ISO/IEC (International Standardization Organization/International Electrotechnical Commission) standard for adaptive streaming later in this year. The MPEG-DASH standard was published in

April 2012 [ISO/IEC MPEG, 2014]. The standard defines guidelines for media presentation, segmentation, and a collection of standard XML formats for the manifest file (MPD). However, specific client implementation, consumption and adaptation algorithms are not part of the standard [Stockhammer, 2011]. This field is left for researchers and industrial actors to explore and define their own solutions for content quality and delivery adaptation. The standard is also codec agnostic to favor and support future improvements in the field of media coding. DASH has been adopted by other standardized multimedia streaming systems such as in the IPTV (Open IPTV [OIPF, 2014]) standard and Digital Video Broadcasting (DVB) [DVB, 2016] for video delivery over the Internet.

The DASH Industry Forum (DASH-IF) [DASH-IF, 2017a] is a group of focusing on streaming companies and researchers leading adoption and research initiatives in current adaptive streaming systems. DASH-IF provides specific implementation guidelines and regular documentation of interoperability [DASH-IF, 2017b]. The community has also developed an open-source dash.js [DASH-IF, 2017c] reference player, which employs the Media Source Extensions of web/HTML5-based browsers. DASH-IF also provides a comprehensive list of publicly available test datasets [DASH-IF, 2017d], network profiles [DASH-IF, 2014] and client software for test, content preparation and validation [DASH-IF, 2016].

2.2.3 HAS adaptation classification

The classification of existing research contributions on content adaptation show two main categories [Diallo *et al.*, 2013]: content adaptation and content delivery adaptation.

The authors Diallo *et al.* [2013] explain that content adaptation is the process of selecting, generating, or modifying content to suit the end-user's preferences, consumption style, computing and communication environments as well as usage context. This directly relates to the version of the content that shall be transmitted in terms of codecs, bitrates, frame rates, video resolutions, etc. Content adaptation can be implemented in three different approaches: client-side approach; server-side approach; in-network (proxy-intermediate) approach. In the client-based approach, the content adaptation logic is located inside the client that attempts to maximize the QoE delivered to the end-user. The client selects the content that fits its hardware requirement (screen resolutions, supported codecs), observable network conditions (achievable TCP throuhgput), and system conditions (buffer occupancy,

battery levels). In the server-based approach, the server takes the decisions regarding the version of the content delivered to the client. Hence the content is adapted by the server before it is sent, thus reducing the transmission time, the bandwidth consumption, as well as the processing time at the client side. This approach scales with more difficulties compared to its client-side alternative as the adaptation logic is not distributed anymore and requires the server to hold and manage information related to the clients' sessions. Examples include Windows Media Services, Adobe Flash Media Server, and QuickTime Streaming Server, where multiple variants of the same content are hosted in the server, and the content best matching to the client's context is selected. As to the in-network approach, content adaptation is performed by an intermediate network element. For example, a proxy between the server and the client gathers the client's hardware capacities and the network characteristics in order to select the best possible content representation based on the retrieved information.

Differently from content adaptation, content delivery adaptation focuses on the service and network aspects of the content transmission only, instead of adapting the content quality. This directly relates to the choice of content delivery techniques as well as the flexibility offered to the clients and servers (unicast or multicast? From which server(s)? Possibility to handover the delivery to other servers? Through which access network? Possibility to use multiple network interfaces? Sequential or parallel segment downloads?).

Most studies focus on content adaptation methods that adapt the content based on information related to network congestion, terminal capacity, measured QoE. Fewer contributions address the adaptation of delivery means for HAS solutions. More importantly, there is very little research work on combining all these factors for both content adaptation and delivery in HTTP adaptive streaming. In our work, we consider a multidimensional approach that adapts the content quality and its delivery means so as to enhance the overall QoE.

2.2.4 QoE of HTTP Adaptive Video Streaming

QoE refers to the subjective perceived quality by end-users. In the case of non quality-adaptive streaming, the essential criteria for QoE can be grouped into two categories: the initial start-up delay and the video stalling due to rebuffering [Hoßfeld *et al.*, 2012, 2011]. When considering quality-adaptive streaming protocols that perform trade-offs between video quality and video rebuffering events,

the introduction of intrinsic video parameter modifications during the video playback influences the end-users' perceived quality. Therefore, the QoE of HAS also includes additional criteria: the average displayed video bitrate, the number of quality switches performed by the protocol and the amplitudes of the latter switches. Several surveys and studies can be found in the literature related to the QoE of HAS [Seufert *et al.*, 2015a; Oyman and Singh, 2012; Seufert *et al.*, 2015b; Hossfeld *et al.*, 2014; Vriendt *et al.*, 2014; Essaili *et al.*, 2013; Yitong *et al.*, 2013b].

The authors in [Hossfeld *et al.*, 2013] conducted QoE evaluations on video glitches and exposed that the perceived end-users' dissatisfaction during the playback of video is highly correlated with the number of video rebufferings. The authors in [Hoßfeld *et al.*, 2012] investigated video rebuffering and initial start-up delay to conclude that rebuffering plays a more important role in the end-user dissatisfaction than the initial delay. Finally, the authors in [Seufert *et al.*, 2015b] draw the conclusions that the number of video playback disruptions should be minimized at all costs, even at the expense of other criteria such as initial delays, mean displayed bitrate or number of quality switches. Studies on QoE for adaptive streaming services [Yitong *et al.*, 2013a; Mok *et al.*, 2012] also pointed out that confining the amplitude of rate variations and minimizing the gap between two consecutive video quality switches reduces the negative effects on the perceived video quality. Finally, the work [Ni *et al.*, 2011] showed that the number of quality switches has a minor impact on the overall observed QoE level. In this thesis, we mostly focus on the initial start-up delay, the number of video stalling events as well as on the mean displayed bitrate in order to characterize the level of QoE obtained.

2.3 Content quality adaptation in HAS

In this section we review some of the most significant content adaptation contributions proposed in the literature. Table 2.1 summarizes the adaptation techniques organized by function types and by the resources driving them.

To better manage the complexity of client-centric adaptation in HAS solutions, the authors of [Jiang *et al.*, 2012] propose a general framework exposing the three major functional components of HTTP adaptive streaming: (1) resources estimation, (2) quality adaptation, (3) segment request scheduling. Figure 2.6 illustrates the interactions between the latter components. The segment scheduling function takes as inputs the history of segment download completion times as well as streaming session related information such as the current buffer level, and is responsible for

deciding how and when the next video segment is to be requested. Then, the adaptation module decides on the video bitrate of the next segment to be downlaoded based on inputs given by the resource estimation module.

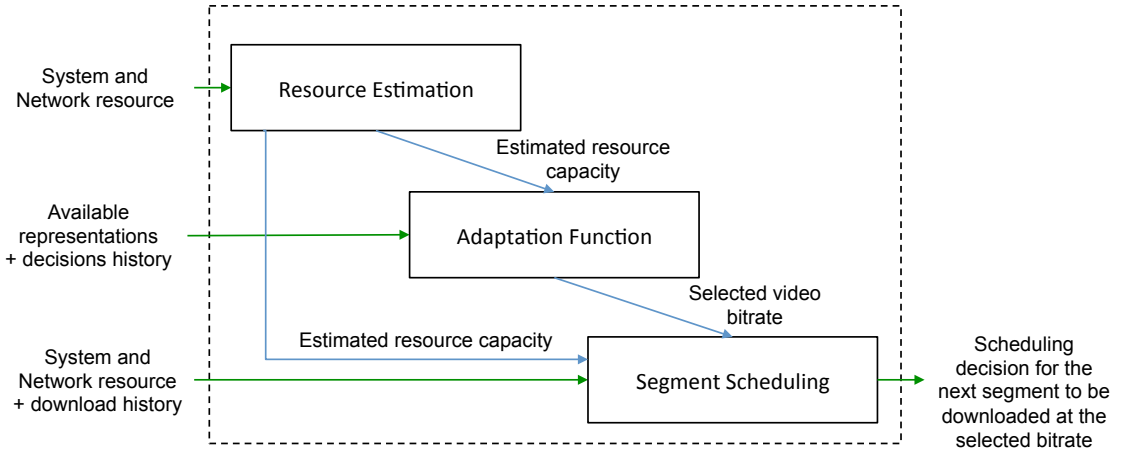

Figure 2.6 – General adaptation framework/scheme of client-centric HAS solutions

Although for the case of client-centric adaptation the latter three modules are embedded in the client, they are not necessarily colocated. Any of these modules can be on a separate system (at the server side, or in the network). In this section, we focus on the client-centric HAS solutions that embed these three components. This section is sub-divided into (1) throughput estimation and (2) content quality adaptation functions. The related work on segment scheduling is surveyed in section 2.4.

2.3.1 Throughput estimation

Resource availability directly affects the capabilities of HAS clients to provide smooth streaming and high QoE to the end-users. Therefore, it is crucial to understand how resources are measured and estimated, and how they affect the content adaptation mechanisms. Typically, a resource estimation function monitors the resource of interest used for the adaptation functions of the HAS client. Because content providers target a large number of clients, and as most of the streaming session's parameters can be better observed from the client's point of view (e.g., last-mile bandwidth, buffer occupancy, etc.), resource estimation is usually implemented at the client-side, hence providing a more scalable solution than its server-side or in-network alternatives. However, solely relying on client-side observations can result in opportunistic behaviors [Huang *et al.*, 2013]. To address this challenge,

Table 2.1 – Quality adaptation function classification

Adaptation function type	Resource dependency	Work
Heuristic	Throughput	[Liu *et al.*, 2011b] [Jiang *et al.*, 2014] [Mok *et al.*, 2012] [Miller *et al.*, 2012] [Akhshabi *et al.*, 2012b] [Sun *et al.*, 2016] [Gouache *et al.*, 2011] [Tian and Liu, 2013] [Thang *et al.*, 2012a] [Thang *et al.*, 2012b] [Ramamurthi and Oyman, 2013] [Ramamurthi *et al.*, 2015] [Houdaille and Gouache, 2012] [Jiang *et al.*, 2016]
	Buffer	[Jiang *et al.*, 2016] [Miller *et al.*, 2012] [Miller *et al.*, 2012] [Huang *et al.*, 2013] [Xu *et al.*, 2014] [Huang *et al.*, 2012] [Huang *et al.*, 2014]
Control Theory	Throughput	[Abdelzaher *et al.*, 2008] [Tian and Liu, 2012] [Zhou *et al.*, 2013b] [Zhou *et al.*, 2013a] [Cofano *et al.*, 2014]
	Buffer	[De Cicco *et al.*, 2011]
	Buffer/Throughput	[Yin *et al.*, 2015] [Yin *et al.*, 2014] [Miller *et al.*, 2015]
Optimization	Throughput	[Qiu *et al.*, 2010]
	Buffer	[Thang *et al.*, 2012a] [Li *et al.*, 2014] [Spiteri *et al.*, 2016] [Cicco *et al.*, 2013]
Artificial Intelligence	Domain expert knowledge	[Xiong *et al.*, 2012] [Vergados *et al.*, 2014] [Sobhani *et al.*, 2015]
	Machine Learning	[Thang *et al.*, 2014] [Chien *et al.*, 2015] [Menkovski and Liotta, 2013] [Claeys *et al.*, 2014b] [Claeys *et al.*, 2014a] [Chien *et al.*, 2015] [Basso *et al.*, 2014] [Hooft *et al.*, 2015]
SVC	Heuristic	[Fuente *et al.*, 2011] [Tappayuthpijarn *et al.*, 2011] [Xiang *et al.*, 2012] [Grafl *et al.*, 2013] [Abboud *et al.*, 2011] [Famaey *et al.*, 2013] [Muller *et al.*, 2012b] [Muller *et al.*, 2012a] [Sieber *et al.*, 2013] [Andelin *et al.*, 2012]

server-centric and in-network solutions have been proposed. A central control plane suggested by Liu *et al.* [2012c] aggregates measurements from many clients. This ensures that the adaptation scheme globally optimizes the performance across all clients. The authors of [Cofano *et al.*, 2014] also propose a network control plane, aiming at maximizing network-wide QoE and bandwidth utilization. The approach relies on bandwidth reservation on a per-video-flow basis.

The choice of resources to be considered as an input for adaptation and scheduling function is also context dependent [Miller *et al.*, 2012]. As exposed by Akhshabi *et al.* [2011]; Thang *et al.* [2012a], the first generation of quality adaptation scheme mostly relies on throughput estimation and always selects the highest video bitrate that fit the measured throughput [Akhshabi *et al.*, 2011; Thang *et al.*, 2012a]. It was assumed that this strategy can avoid rebuffering while at the same time providing the highest possible video quality. Later, it became obvious that throughput estimation alone is not a sufficient parameter for efficient adaptation scheme [Jiang *et al.*, 2014; Huang *et al.*, 2012]. For a player not to run out of playable content, the delivered throughput should be at least equal to the rate at which the content is being decoded. In HAS, video segments are pre-fetched and are stored into the client's local buffer. This ensures that the HAS client will continue displaying video from the buffered video for at least the duration of the buffered content. Hence, there is an inverse relationship between buffer occupancy and the probability of video stalls (i.e., the bigger the size of a buffer, the longer it takes to run out of content). Therefore, the buffer occupancy was introduced as another parameter for the design of efficient client-side quality adaptation algorithms.

2.3.1.1 Throughput estimation techniques

Content adaptation schemes based on throughput availability try to estimate average unutilized capacity of a network path over a specific time period [Jain and Dovrolis, 2004]. However, because the available bandwidth in the networks is time-varying [Prosad *et al.*, 2003] and because HAS employs TCP as its transport protocol, the variability of the observed throughput (usually estimated above the application layer) is highly intensified by the TCP characteristics (i.e., slow start and congestion control, etc.), resulting into potentially inaccurate estimates [Huang *et al.*, 2012].

Additionally, due to the discrete nature of HAS solutions, bandwidth estimation is performed on a per-segment download basis. The idea is to use the throughput of a recently downloaded segment as an approximate estimate of the current network

conditions. This is typically performed by dividing the amount of data obtained for a given segment by the segment download duration. However, the instant throughput derived from the download of a single segment is very likely to be affected by short-term fluctuations as a result of the time-varying nature of the available bandwidth, or the dynamics of TCP. Regardless of the cause of fluctuations, the per-segment estimated throughput can result in significant variability in the video quality. Due to this and to the difficulty in accurately estimating throughput above the HTTP layer, various techniques are used to improve the quality of the measurement.

Before HAS techniques were proposed, Prosad *et al.* [2003] argued that appropriate bandwidth estimation techniques need to average the instantaneous estimates over a time. Several research papers on HAS have used different types of averaging techniques to estimate the available throughput. Akhshabi *et al.* [2011] experimentally evaluated some industral HAS solutions. For the case of the Microsoft Smooth Streaming player [Microsoft, 2017], they observed that regardless of the client's decisions in increasing or decreasing video quality, the decisions were not instantaneously following throughput variations. In [Jiang *et al.*, 2014], a harmonic mean is used to smooth the estimated instantaneous throughput to provide robustness to large outliers. The authors Qiu *et al.* [2010] employed an exponentially weighted moving average. In doing so, they are not only able to incorporate historical estimates into the current throughput estimate but they also exponentially reduce the significance of the historical data as time passes. In [Gouache *et al.*, 2011], the authors compute the moving average and the standard deviation of the estimated throughput for several segments using a low-pass filter. The throughput is then calculated as the difference between the computed moving average and a factor of the standard deviation. The latter factor controls the conservativeness of the adaptation logic. The authors showed that the algorithm can obtain estimates close the actual throughput capacity and is robust to network errors. To improve the stability of quality adaptation algorithm that relies on this type of history-based throughput prediction, a safety factor is sometimes applied to the estimate, as discussed by Tian and Liu [2013]; Akhshabi *et al.* [2012b]. On the downside of estimation smoothing techniques, the resulting responsiveness of the associated content adaptation algorithm is reduced. In the context of video quality adaptation, this may result in late response to large throughput decrease and subsequently lead to video freezing events, especially when the buffer occupancy is critically low. To counteract this lack of responsiveness, an adaptive coefficient (weight) for the weighted moving average is proposed in [Thang *et al.*, 2012a,b]. The approach increases the responsiveness of

the quality adaptation algorithms without causing unnecessary video rebuffering.

2.3.1.2 Reliability of the throughput estimate

An open challenge in the adaptation schemes of HAS is the extent to which the throughput estimates truly reflect the available bandwidth. Any throughput measurement done at the application layer can only consider the throughput calculated by the underlying TCP protocol. However, the authors of Jain and Dovrolis [2004] argue that equating the available bandwidth with the TCP throughput is error-prone since TCP throughput depends on many factors (including socket buffer sizes at the sender and receiver, the nature of the competing traffic, RTT, packet loss rate, the nature of TCP congestion control etc.). Similarly, an argument against matching the TCP throughput observed at the application layer with the available bandwidth is presented in [Li *et al.*, 2014]. The paper showed that when clients compete on the same bottleneck, the presence of competing applications and the ON-OFF nature of the HAS downloads make it difficult for a client to correctly perceive its share of the available bandwidth. This results in an under-utilization of the available bandwidth leading to video quality flickering, which is known to negatively impact the end-users QoE [Akhshabi *et al.*, 2012a; Seufert *et al.*, 2015a]. To tackle this problem, the paper proposes a "Probe ANd Adapt" (PANDA) technique. The algorithm somehow copies the congestion control of TCP at the application layer. The TCP throughput is then used as input when it represents an accurate indicator of the fair-share of bandwidth, which is argued to happen when the network is congested. Otherwise, the algorithm probes the network by incrementing the sending rate and stops when a congestion is detected.

Application layer based schemes are not the only techniques used in predicting the available throughput. Many attempts were made on cross-layer throughput estimation. Based on the machine learning methods of Mirza *et al.* [2007], the authors [Tian and Liu, 2012] predict the achievable throughput. The latter method uses the support vector regress algorithm [Cortes and Vapnik, 1995] to train a throughput prediction model with network layer information such as packet loss, delay, and RTT. In [Ramamurthi *et al.*, 2015; Ramamurthi and Oyman, 2013], the throughput measured at the physical layer is used to complement the application layer estimate. This results in an improvement in the perceived video quality and a reduction in the rebuffering frequency.

2.3.2 Adaptation functions

The adaptation function -also called adaptation logic or adaptation strategy- is the element within the HAS framework that decides the representation of a segment to be requested in terms of video bitrates, resolutions, framerates, codecs, etc. Although HAS permits to adapt the content according to a large panel of criteria, research contributions principally focus on adjusting the video bitrate (which can then be turned into other parameters) as it is the fundamental parameter that should best match the available network resources if video freezing events are to be prevented. Most adaptation logics usually take as input information regarding the available resources and the set of all the possible content representations in order to return the quality of the next segment to be downloaded.

In its most basic form, the adaptation logic simply chooses a segment with the highest video bitrate that the estimated available resource can support. Although easy to implement, this basic algorithm is very sensitive to the bandwidth variations, which can make its outcome oscillatory, abrupt [Akhshabi *et al.*, 2012a; Rao *et al.*, 2011], and unfair in allocating bandwidth to competing clients [Huang *et al.*, 2012]. Traditionally, most HAS players rely on an adaptation logic that emulates the concept of an AIMD control scheme due to its rapid convergence to an efficient usage of the resources [Chiu and Jain, 1989]. Indeed, it permits to reduce video quality fluctuations and abrupt changes in video quality that are not well perceived by end-users [Mok *et al.*, 2012]. Furthermore, it was demonstrated in [Cranley *et al.*, 2006; Mu *et al.*, 2015; Georgopoulos *et al.*, 2013] that when the video quality is high, an aggressive increase in the requested video bitrate does not necessarily translate into an improvement in the end-users QoE. Therefore, with AIMD-like approaches, a client increases the video bitrate of segment requests in a stepwise manner. When the network resources weaken, the bitrate is aggressively reduced to avoid fast buffer depletion and to allow a rapid refill of the buffer. However, it has been shown in [Huang *et al.*, 2012] that the simpler the adaptation logic the better, regardless of the approach used in building an adaptation module. In the literature, the spectrum of quality adaptation approaches can be divided into five main categories(Table 2.1): (1) heuristic based adaptation, (2) control theory based adaptation, (3) optimization based adaptation, (4) artificial intelligence and (5) layered-coding based adaptation.

2.3.2.1 Heuristic based adaptation

Most of the early adaptation proposals are based on heuristics. For instance, the authors of [Liu *et al.*, 2011b] implemented an AIMD-like adaptation logic that employs a stepwise increase and aggressive decrease logic. This technique prevents sudden video quality degradation. They also point out that with an aggressive switch-down in quality the video buffer refills faster, which lessens the probability of video buffer unde-run, thus preventing video interruption. However, the algorithm was found to chose suboptimal video representation and to provide unstable video quality.

As discussed in the previous section on the QoE of HAS, drastic quality decreases of the video bitrate has a negative impact on the end-users QoE. In order to improve the QoE, Mok *et al.* [2012] propose a QoE-aware adaptation algorithm called QDASH (QoE-aware DASH). The client reduces the video bitrate in a stepwise manner when the achievable throughput drops. Although this may result in suboptimal choices, the QoE is improved by enhancing the stability of the delivered quality. The conducted experiments in [Akhshabi *et al.*, 2012b] showed that the Microsoft Smooth Streaming HAS player is using a similar approach. Although the switch-up transitions are faster than the downward transitions, the quality switching is not immediately performed to the video quality that matches the network throughput.

Another work on heuristic based adaptation is FESTIVE [Jiang *et al.*, 2014]. The client slowly switches to the top video quality with lower incremental step in the video bitrate when the actual bitrate of the downloaded segment increases. In doing so, the authors propose to mitigate the unnecessary oscillation between different video representations. The authors introduce a score that measures the trade-off between efficiency/fairness and stability, allowing improvement in the session's stability.

By analyzing the throughput characteristics of a large dataset consisting of 20 million sessions, Sun *et al.* [2016] developed CS2P (Cross Session Stateful Predictor) to improve the bitrate selection and adaptation by using data-driven throughput prediction algorithms. Similarly to Jiang *et al.* [2016], the authors concluded that sessions sharing similar key features (e.g. ISP, geographical region) have similar network throughput values and dynamic patterns. They observed a natural stateful behavior in the throughput variations within a streaming session. Subsequently, the authors proposed a throughput prediction system that uses data-driven approach to learn clusters of similar sessions, predict initial throughput, and model the throughput evolution with a hidden-markov-model. After integration within a DASH player

and real-world experiments, CS2P outperformed existing prediction approaches by up to 50% in terms of throughput prediction error.

Huang *et al.* [2014, 2012, 2013] are among the first to contribute to content adaptation assisted with buffer-based adaptive strategies. Indeed, only the buffer state is used to determine the video bitrate of the next segment to be downloaded. Nevertheless, when the buffer level is too low and prevents decision making, throughput estimation is performed by probing the network. The proposed algorithm was experimentally evaluated with real end-users on the Netflix streaming platform. Results showed that this approach reduce the amount of rebufferring events by 10 to 20% compared to Netflix's algorithm, while delivering a similar average video bitrate.

Miller *et al.* [2012] proposed an algorithm that uses three threshold levels for the playout buffer, such that $0 < B_{min} < B_{low} < B_{high}$. The target interval B_{target} is between B_{low} and B_{high}, and the optimum buffer level $B_{optimum}$ is at the middle of the target interval. The algorithm attempts to keep the buffer level close to $B_{optimum}$. It allows the designer to explicitly control the trade-off between the variations in buffer occupancy and the fluctuations in video bitrate by controlling the B_{low} and B_{high} thresholds. Based on experiments conducted in a WiFi environment with and without throughput limitation at the server side, the authors showed that the algorithm presents a stable and fair behavior when multiple clients compete on a common network path. Other players that employ heuristic based adaptation logics are proposed in AdapTech Streaming [Akhshabi *et al.*, 2012b], and in the Akamai HD Video Streaming services [Cicco and Mascolo, 2010].

2.3.2.2 Control theory based adaptation

There have been many attempts to design adaptive bitrate strategies based on predictive and descriptive models. Control theory is used to model dynamical systems that are stable, accurate and settle quickly into a steady state [Abdelzaher *et al.*, 2008]. The controller manipulates the inputs of a system to produce the desired outputs. Typically, a controller computes the distance between a measured variable and an output value as a process error. The goal is to reduce this error by adjusting the input parameters.

The authors of [De Cicco *et al.*, 2011] propose an adaptation logic based on feedback control. The video rate adaptation controller takes a target buffer as an input and returns the video rate of the segment to be downloaded. The goal of the controller is to ensure that the buffer is always maintained at the target level. This is achieved by computing the error between the target buffer and the measured buffer

level. The error is then passed to a proportional integral controller that outputs a video bitrate matching the estimated available throughput. Experiments confirm that the controller selects the highest video bitrate that the available bandwidth can sustain. In [Tian and Liu, 2012], a control theoretic client-side rate adaptation performs a trade-off between the stability of the video quality and bandwidth utilization. In [Yin *et al.*, 2015, 2014], a model predictive control based algorithm is proposed to optimally combine throughput and buffer occupancy feedbacks. The authors formulate the video bitrate selection problem as a stochastic control problem to predict the expected throughput for the future segments and to maximize the end-users' QoE. Other papers propose adaptation functions that are implemented using control theory [Zhou *et al.*, 2013b; Miller *et al.*, 2015; Zhou *et al.*, 2013a; Cofano *et al.*, 2014].

2.3.2.3 Optimization based adaptation

Qiu *et al.* [2010] tried a different approach by exploiting an optimization technique for bitrate adaptation called Intelligent Bitrate Switching. The authors modelled the adaptation logic as an optimization problem, which maximizes benefits -the quality level of each segment- while minimizing penalties. A maximum penalty is assigned to video stalls. The authors proposed an adaptable model where users can adjust the penalty score based on its viewing experience. An optimal solution is expected to select a segment with the highest video rate among all the segments that satisfy the given constraint of a minimum number of video interruptions. Although metrics like PSNR (Peak signal to Noise Ratio) are not sufficient to capture the quality perceived by the end-users, the authors claim that the proposed algorithm can also use the PSNR to select the best solution. Another adaptation logic based on optimization techniques is presented in [Thang *et al.*, 2012a].

According to Bouten *et al.* [2014, 2013], the support for coordinated management and global optimization is essential to improve QoE. The authors propose to control the allocated network resources among competing clients. They employ an integer linear programming (ILP) model to either maximize the QoE of all end-users or minimize the penalties incurred when resource allocation is not optimal. The authors of [Joseph and Veciana, 2014] propose NOVA to solve the resource allocation and quality adaptation problem for multiple clients employing optimization techniques. The algorithm attempts to maximize the average video quality and minimize the quality variations of HAS streaming session under network constraints.

Li *et al.* [2014] propose the "Probe ANd Adapt" (PANDA) mechanism for bitrate

adaptation which emulates the TCP congestion control at the application layer. PANDA probes the network by setting a target average data arrival rate, which in turn, is used to determine the next video segment bitrate and the interval between two segment downloads. The emulated congestion control employs an AIMD probing mechanism. However, unlike TCP that detects congestion based on packet losses or increases in RTT, PANDA detects congestion with the reduction of throughput. This feature ensures that PANDA clients converge to a fair utilization of bandwidth. The PANDA mechanism schedules the next request by considering the average target rate and buffer level. PANDA incorporates a buffer filling based adaptation algorithm that solves the quality selection optimization problem. Li *et al.* [2014] claim to rely on the PSNR to define quality perceived by end-users. In testbed experiments, PANDA reduces video instability by 75% when compared with other conventional algorithms.

The authors Spiteri *et al.* [2016] formulated video quality adaptation as a utility maximization problem and proposed an online control algorithm *BOLA*, using the Lyapunov optimization functions to minimize rebuffering and maximize video quality. BOLA does not require any throughput estimation, and assumes that the buffer level is sufficient to provide all the information about past bandwidth variations. The authors evaluated BOLA on 12 test vectors defining network characteristics (bandwdith delay, packet loss), referred as network profiles, provided by DASH-IF [DASH-IF, 2014] with 85 publicly available 3G mobile bandwidth traces. They compared the obtained results with an optimal offline algorithm that guarantees the maximum achievable time-average utility for any given network trace (having the prior knowledge of future bandwidth variations) and found that BOLA achieves between 84% and 95% of the optimal utility. The authors also compared BOLA with ELASTIC [Cicco *et al.*, 2013] and PANDA [Li *et al.*, 2014], and concluded that BOLA provides higher utility.

2.3.2.4 Artificial intelligence

In [Xiong *et al.*, 2012], the authors argue that since the end-users QoE is not easily described and does not exclusively rely on video bitrate, rebufferings and quality changes, control theory and other mathematical models that are based on precise definitions of input and output are not necessarily the best tools for video bitrate adaptation strategy. Subsequently, they propose a fuzzy controller based on fuzzy logic, called Network-Bandwidth-Aware Streaming Version Switcher. The fuzzy controller is composed of three components, a fuzzifier, a fuzzy interface engine, and a

defuzzifier. The fuzzifier takes as input the estimated throughput, and translates it into a format that the controller understands. The fuzzy interface engine takes the fuzzified input and produces an output based on rules generated from the domain knowledge and expert experience. The defuzzifier converts the produced output to a format that the system understands (i.e., the video rate of the segment to be downloaded). Experiments showed the proposed technique to be responsive to changes in network conditions although with unwanted instability even in the presence of stable throughput.

Unlike Xiong *et al.* [2012], Vergados *et al.* [2014] suggest a fuzzy logic based on the buffer state changes to adjust the video bitrate to the changing network conditions. The aim of the algorithm is to prevent buffer overflows and unnecessary fluctuations in the video quality. However, the algorithm suffers from a high amplitude of quality variations. The authors of [Sobhani *et al.*, 2015] tackle this issue by proposing an AIMD-like fuzzy controller that considers both the estimated throughput and buffer occupancy and returns the appropriate video bitrate for the next segment.

Fuzzy logic requires the usage of domain expert knowledge, which is hardly accessible and difficult to acquire. Even with such knowledge, defining a set of rules based on it [Thang *et al.*, 2014] is very challenging. Based on this fact, other research work [Chien *et al.*, 2015; Menkovski and Liotta, 2013; Claeys *et al.*, 2014b,a] looked at alternative artificial intelligence techniques free of domain expert knowledge requirement to perform video quality adaptation: machine learning. With machine learning techniques, a client learns to adjust its video quality to the evolving network context without the need of any human intervention. In [Chien *et al.*, 2015], MLASH (Machine Learning-based Adaptive Streaming over HTTP), an elastic framework that exploits a wide range of useful network-related features to train a rate classication model is presented. Rather than proposing a new algorithm, the authors proposed to rely on machine learning to improve the accuracy prediction of network conditions (bandwdith, round-trip tume) and streaming conditions (buffer occupancy, video bitrate) of existing adaptation algorithms. MLASH is is trained on a dataset provided in [Basso *et al.*, 2014]. The training can be done either online or offline. Trace-based simulations show improvements in the accuracy of the prediction made when MLASH is incorporated into throughput-based and buffer-based adaptation logic proposed in the literature.

Classification schemes generally require a training dataset. However, in highly dynamical system like HAS, it is difficult to obtain a training set that is both correct

and representative of all possible situations. Reinforcement Learning (RL) allows an agent to discover the right action to take within a specific context based on a feedback from its environment. To do so, an adaptation module interacts with its environment by sensing the factors that are expected in-advance to influence its decision. For example, the authors of [Hooft *et al.*, 2015] use the average and the mean absolute difference in bandwidth, while [Claeys *et al.*, 2014b,a] rely on information about both the buffer occupancy changes and the available bandwidth. Then the agent acts typically by changing the video bitrate to incrementally maximize its reward, such as improving a mean opinion score and reducing the rebuffering [Hooft *et al.*, 2015].

2.3.2.5 Layered coding content adaptation

In the literature, most HAS research work assume that every segment is self-contained and independently encoded. To some extent, this is a valid assumption since most video codecs, including the widely adopted H.264/AVC and VP8, propose the latter content format. However, for each representation, all segments have to be encoded and stored separately which can represent significant storage requirement for a video streaming provider employing HAS-based techniques. In [Huysegems *et al.*, 2012], the authors could show that the Microsoft Smooth Streaming services necessitates between 200% to 300% of storage overhead compared to having only the highest video representation available. The authors of [Fuente *et al.*, 2011] also confirm the suboptimal performance of self-contained segment based HAS in terms of caching efficiency and of additional bandwidth to transport the segments to the servers and caches in the network [Lin and Hwang, 2011].

The purpose of any quality adaptation logic is to enable clients to adjust the quality of the requested video to evolving conditions. In self-contained segment based HAS services, a segment must be completely delivered. A better solution argued by [Fuente *et al.*, 2011] is to use layered coding. Scalable Video Coding (SVC) [Schwarz *et al.*, 2007] is an extension of the H.264/AVC standard for layered video coding. Layered coding allows the encoding of a video into a number of layers, composed of a unique base layer (representing the least quality level) and several enhancement layers, with each enhancement layer improving the viewing quality. With SVC, a video is encoded once into multiple layers, and is decoded based on frame rate, resolution, or fidelity requirements. In this way, a client can select the appropriate number of layers in order to adapt the content to the varying network conditions as well as the varying terminal capabilities [Schwarz *et al.*, 2007].

SVC allows for three different kinds of scalability: temporal, spatial and quality. With temporal scalability, the base layer represents the source content with a reduced frame rate, while in the case of spatial scalability the resolution is reduced. The quality scalability presents a scenario where the base layer has the least fidelity (in terms of PSNR). Any addition of enhancement layers to a base layer increases one of the latter parameters in the displayed video. A detailed discussion on SVC can be found in [Schwarz *et al.*, 2007; Schwarz and Wien, 2008; Unanue *et al.*, 2011]. Even with layered coding, a video file needs to be chopped into segments to suit HAS. Multiple segment creation strategies were proposed in the literature. In [Tappayuthpijarn *et al.*, 2011], each segment is composed of several blocks, each block representing a layer. Xiang *et al.* [2012] propose a different approach, the encoded video is divided along the layers, and then split into segments. Therefore, each segment request refers to a specific layer. The authors of [Grafl *et al.*, 2013] use multiple independent groups of segments, each group of segments having the same class of base layer so that they represent a particular resolution. Layers within a segment are used for quality adaptation.

Because DASH is codec agnostic, SVC video segments can easily replace single-layer segments in traditional bitrate adaptation strategies. For instance, Abboud *et al.* [2011]; Famaey *et al.* [2013] use SVC with the open source version of the Microsoft Smooth Streaming adaptation algorithm. Quality adaptation is performed by selecting a base layer and the required enhancement layers matching the available resources. The authors of [Muller *et al.*, 2012b; Sieber *et al.*, 2013] successfully adapted SVC segments to the adaptation scheme proposed in the work of Muller *et al.* [2012a], originally designed for single layered content.

The techniques discussed download all together a base layer and the required enhancement layers. These techniques are referred to as *vertical adaptation* because they sequentially decide on the number of layers that should be retrieved for each video segment. Oppositely, horizontal adaptation techniques Sieber *et al.* [2013] propose to have the client retrieving a fixed number of consecutive video segments with the base layer only. Then, enhancement layers are retrieved sequentially in order to gradually increase or reduce the visual quality of the video and to provide consistency in the displayed quality.

The authors of [Andelin *et al.*, 2012] propose the combination of the horizontal and vertical adaptation techniques, i.e, diagonal adaptation. As a resut, the granularity of bitrate adaptation is improved by allowing the client adaptation module to cancel the ongoing downloads without much penalty. In the diagonal scheme, a

client can either choose to operate the *backfilling* mode by downloading enhancement layers of the current segment -hence increasing the current quality- or to operate the *pre-fecthing* mode by downloading the base layer of the next segment to aim at an uninterrupted streaming experience while ensuring better quality in the future.

In their work, Andelin *et al.* [2012] propose a heuristical model to determine how a client alternates between the backfilling and pre-fetching modes.

Many researchers have investigated the performance of SVC in HAS [Fuente *et al.*, 2011]. In addition to better caching efficiency, and since SVC allows clients to abort segment downloads without much overhead, the use of SVC improves the responsiveness in HAS schemes to the variations of network conditions [Huysegems *et al.*, 2012]. The authors of [Famaey *et al.*, 2013; Basso *et al.*, 2014], pointed out that due to the increased number of requests compared to single layered based HAS solutions, SVC based HAS proposals are more vulnerable to high RTTs. Indeed, when RTT augments, the achievable throughput decreases, and SVC-based HAS techniques are expected to perform badly in low throughput conditions. In addition, despite reducing the storage requirements of HAS, SVC-based solutions require at least 10% of encoding overhead [Schwarz *et al.*, 2007] in terms of data storage, resulting in higher bandwidth requirements. Kalva *et al.* [2012] compared the financial cost of the storage reduction to the cost of bandwidth requirement increase, and found that the latter outweighs the former. For all these reasons, SVC has been been adopted in the industry, despite its advantages.

2.4 Adaptation of both quality and delivery in HAS

In this section, we look at content delivery adaptation and quality adaptation proposals made in the literature. We review server-side and in-network solutions before exposing several works on client-side segment scheduling. Quality adaptation is referred to as the process of selecting, generating, or modifying content to suit the end-user's preferences, consumption style, computing and communication environments, and usage context. Delivery adaptation refers to adaptation made on the service and network aspects of the content transmission only, instead of adapting the content quality.

2.4.1 Server-side and in-network content delivery and qualiy adaptation

We now look at some existing server-side, transport layer and network-level solutions for optimizing HAS-based services. Although most contributions on HAS focus on client-side adaptation mechanisms due to ease of implementation and scalability, there has been some work on optimizing server-side bitrate selection and congestion control. Table 2.2 summarizes the different proposals on content delivery and quality adaptation found in the literature.

Table 2.2 – Content delivery and quality adaptation classification

Content delivery mean and quality adapation	Work
Server-side application layer	[Akhshabi *et al.*, 2013] [De Cicco *et al.*, 2011] [Mueller *et al.*, 2012] [Alcock and Nelson, 2011] [Satoda *et al.*, 2012]
In-network solutions	[Mok *et al.*, 2012] [Mansy *et al.*, 2013] [Pu *et al.*, 2012] [Havey *et al.*, 2012] [Siekkinen *et al.*, 2013]
SAND	[Begen *et al.*, 2016b] [Halepkidis *et al.*, 2015] [Kleinrouweler *et al.*, 2016] [Cofano *et al.*, 2016]
New protocols (HTTP 2.0, QUIC, SPDY)	[Wei and Swaminathan, 2013] [Huysegems *et al.*, 2015] [Cherif *et al.*, 2015] [Mueller *et al.*, 2013] [Timmerer, 2016b] [Timmerer, 2016a] [Cardwell, 2016]

2.4.1.1 Server-side application layer

Akhshabi *et al.* [2013] proposed a server-based traffic shaping method to reduce video bitrate variations and instability due to multiple clients competing on the same bottleneck. Such instability is caused by the ON-OFF activity pattern of clients, which can lead to bandwidth over-estimation and video bitrate variations between consecutive video segment downloads. A traffic shaping module is proposed to limit the throughput for each video stream sent to the clients to the actual video bitrate of the segment. In doing so, the download duration is roughly matching the segment duration, hence reducing the effect of the OFF periods, as long as the available bandwidth is higher than the limited throughput. The approach aims to stabilize the players by allowing them to request the highest video bitrate that will not cause too high oscillations in bandwith usage. Experimental evaluations on

different scenarios with multiple shaped/unshaped players competing with persistent TCP transfers show significant reduction of bandwidth usage fluctiations without a major loss of bandwidth utilization.

The authors of [De Cicco *et al.*, 2011] propose a control-theory server-side quality adaptation switching technique by defining a Quality Adaptation Controller (QAC). A feedback control loop is employed based on information reported by the clients regarding the successful delivery of a particular video bitrate. QAC utilises two controllers: a playout buffer level controller whose goal is to bring the buffer occupancy level to a target length; and a stream-switching logic that selects the appropriate video level to be streamed to the clients. They ran their prototype against Akamai's Adaptive HD video server and found that QAC is able to throttle the video quality and match the available bandwidth within a delay of 30 seconds at most. Additionally, the available bandwidth is fairly shared in the presence of cross-traffic. Another work by Mueller *et al.* [2012] proposes an adaptation algorithm implemented at the proxy level by using the concept of fairness among clusters of clients consuming the same videos.

In [Alcock and Nelson, 2011], the authors explained that YouTube uses an application flow control technique. Satoda *et al.* [2012] introduced a server-side adaptive video pacing algorithm that delivers video data just-in-time in order to prevent bandwidth waste caused by the downloads of unnecessary segments that end-users may no watch. The proposed technique adapts the delay between two consecutive segment deliveries so as to keep the playout buffer length as close as possible to a pre-determined value. The quality adaptation logic is led by a throughput estimation method employing a stochastic Brownian model. This method allows network operators to better manage their bandwidth for video traffic while maintaining satisfying QoE for the end-users. The authors conducted experiments in both High Speed Downlink Packet Access (HSDPA) and LTE environments and showed that the method can decrease the playout buffer size by up to 42%.

2.4.1.2 In-network solution

The authors Mok *et al.* [2012] that proposed the QoE-aware DASH system (QDASH) incorporate an in-network adaptation solution with the use of a bandwidth measurement proxy. Based on subjective experiments under Adobe OSMF, the authors concluded that end-users prefer a step-wise quality variations rather than sudden and abrupt changes. QDASH consists of two modules: QDASH-abw and QDASH-qoe. QDASH-abw is a measurement proxy placed in front of the server.

The proxy probes and detects the highest quality level that the current network conditions can support by measuring the available bandwidth based on RTT estimates. On the client side, QDASH-qoe adapts the video bitrate by relying on measurement updates delivered by QDASH-abw. Mansy *et al.* [2013] evaluated the performance of DASH streaming to mobile devices with different operating systems. They observed that unfairness can result when different device platforms are used. In [Pu *et al.*, 2012], the authors proposed a proxy for video adaptation between fixed and wireless networks to increase the fairness for wireless clients.

Another work on proxy that can be applied to HAS is MOCHA, proposed by [Rejaie and Kangasharju, 2001], a quality adaptive proxy being able to cache multiple layer-encoded content. The proxy adjusts the quality of cached layers based on their popularity and the available bandwidth between the proxy and clients. In doing so, MOCHA improves caching efficiency without compromising the delivered quality. The algorithm implements fine-grained replacement and fine-grained pre-fetching mechanisms to adaptively increase or decrease the quality of cached streams. Although MOCHA is oriented towards RTP based solutions (because HAS was not coined yet), the proposed concept can be applied to HAS because MOCHA relies on clients sequentially donwloading video chunks (similarly to HAS).

2.4.1.3 Server and Network assisted DASH

As previously discussed, client-centric adaptation approaches are not optimal because of the multiple HAS clients competing for a limited resource on a bottleneck link. To address this issue, research studies have been conducted to define interaction between video and network elements in different ways so as to leverage the in-network information. In this regard, MPEG introduces the new baseline architecture SAND (Server and Network Assisted DASH) [DASH-IF, 2016; ISO/IEC, 2017c] that defines the signaling mechanisms enabling network assisting adaptative streaming strategies, as illustrated in Figure 2.7. The cooperation and information exchange between network elements and HAS clients are enabled in order to manage traffic and to support QoS. SAND has also emerged in the IETF [Begen *et al.*, 2016b,a].

Figure 2.7 shows bi-directional messages between a HAS client and other DASH-Aware Network Elements (DANE), giving them the means to trigger a control mechanism such as flow prioritization, bandwidth reservation and video quality adaptation based on the network state. PER (Parameters for Enhancing Reception) and PED (Parameters for Enhancing Delivery) messages are exchanged between the

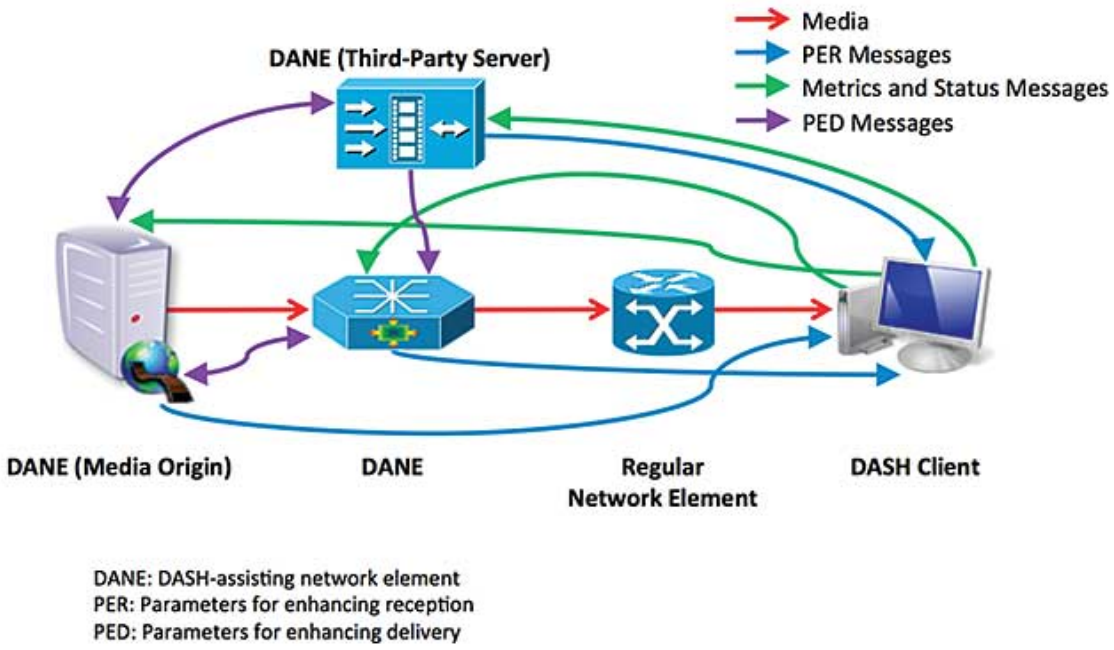

Figure 2.7 – Server and Network assisted DASH streaming architecture [Begen *et al.*, 2016a]

network elements.

Software Defined Networking (SDN) [Halepkidis *et al.*, 2015] is a viable technology to implement such mechanisms due to the presence of a centralized control element. In [Kleinrouweler *et al.*, 2016], the authors proposed a DASH-aware networking architecture based on SDN. Network controllers, with a broad overview on the network activity, provide two mechanisms for explicit adaptation assistance: (1) signaling the target bitrates that DASH players should select; (2) dynamically controlling traffic in the network to enable QoS. The authors evaluated their prototype in a WiFi setting and showed that the video bitrate can be doubled while the number of quality switches are significantly reduced. This approach enables ISPs and network administrators to configure and define how bandwidth should be shared between video and non-video traffic, and among competing video players.

Cofano *et al.* [2016] investigated several network-assisted streaming strategies (with SDN) by relying on active cooperation between the video streaming applications and the network. They built a video control plane which enforces video quality fairness among concurrent video flows generated by heterogenous client devices. They compared two approaches to reach the optimal solution in an SDN network: (1) bandwidth allocation, and (2) video bitrate adaptation assisted strategies. Bitrate

adaptation assistance provided the best results in terms of video quality fairness among clients, whereas bandwidth allocation improved the average video quality depending on the adaptive strategy.

2.4.1.4 DASH with HTTP 2.0 and QUIC

Google developed SPDY [Chromium, 2010] (later becoming HTTP 2.0 in [Belshe *et al.*, 2015]) and Quick UDP Internet Connections (QUIC [Chromium, 2017], in Internet Draft [Hamilton *et al.*, 2016]) to reduce web latencies.

In the context of HAS streaming, HTTP 1.1 introduces a latency of at least one segment duration, representing a problem in live video streaming scenario where latency constraints are usually tighter than for VoD. To tackle this issue, a low latency live video streaming technique over HTTP 2.0 is proposed in [Wei and Swaminathan, 2013; Huysegems *et al.*, 2015; Cherif *et al.*, 2015]. The authors propose a server push strategy based on HTTP 2.0 allowing a server to push content to a receiver directly without the need of an explicit request. The authors of [Wei and Swaminathan, 2013, 2014] presented three-based push strategies that exploit the server push feature of HTTP 2.0: no-push, pull-push and k-push. In pull-push, after the initial request, the server sequentially sends segments to clients without any pause, and stops only when explicitly requested by the client. In k-push, a client initiates a request for one segment, the server deliver the identified segment and then pushes the next k-1 segments. The no-push scenario is the usual one when the server is not allowed to push segments. The push strategies were found to reduce live latency, improve link utilisation [Wei and Swaminathan, 2013], and overall improve the live streaming performance.

The work in [Mueller *et al.*, 2013] presented the first comparison between HAS over HTTP 1.1 and over HTTP 2.0 (SPDY) -that offers HTTP 1.1 functionalities implicitly- with and without SSL encryption. The VLC DASH player [Müller and Timmerer, 2011] is employed to evaluate the performance of DASH in terms of protocol overhead and performance over 0 to 150ms RTT. They found out that the overhead for both HTTP versions are small, i.e 5-7% for 2-sec segments and video bitrates higher than 700kbps. Both HTTP 1.1 and SPDY perform consistently over RTTs between 0 and 150ms due to the persistent connection and pipelining features. Despite the overhead introduced by SPDY framing and being not as efficient as HTTP 1.1, SPDY and SPDY with SSL encryption are very robust against increasing RTT because they are maintaining only one TCP connection during the whole session. However, SSL encryption introduces additional computational overhead on

both server and client.

Carlucci *et al.* [2015] evaluated QUIC in terms of web content traffic (not streaming) and assessed the performance of QUIC compared to SPDY and HTTP 1.1. Preliminary experiment results and analysis on video streaming over QUIC are presented in the reports [Timmerer, 2016b,a]. In their experiments, the authors compared the performance of DASH streams when using HTTP 1.1, HTTP 2.0/SPDY over TCP and QUIC using a controlled testbed. QUIC comes with a slightly higher protocol overhead than TCP but is below 10% except for very low bitrates (less than 100kbps). The link utilization decreases with increasing RTT but remains higher than 87% of the available bandwidth and is steady for different bandwidths. A new transport protocol named BBR (Bottleneck Bandwidth and RTT) [Cardwell, 2016] aims at maximizing the network throughput with minimal queue by probing the network bandwidth and RTT sequentially. BBR is now deployed for all TCP services on the Google Wide Area Networks (WAN) backbone, and has replaced TCP for Google and YouTube services.

2.4.2 Client-side sequential versus parallel segment scheduling approaches

Segment scheduling is the process of deciding how and when the client requests segments from the server(s). A scheduler takes as input a set of parameters (such as the buffer size, the target video quality, the target buffer level, the time the previous download ends, etc.) and outputs the time to send the next request to the considered server(s). Segment scheduling can either be sequential or parallel [Liu *et al.*, 2012a]. In a sequential scheduler, video segments are requested one after the other, not necessarily immediately after receiving a response from a server. Oppositely, a parallel scheduler sends multiple segment requests at the same time. However, this does not imply that each request is for a separate segment. In some cases, multiple requests are sent for the same video segment, with each request targeting a subpart of the segment. Parallel scheduling is mainly used when a client intends to use multiple interfaces and/or wants to access content from multiple locations in order to increase streaming performance. Table 2.3 summarizes the different proposals on client-side segment scheduling found in the literature.

Table 2.3 – Classification of segment scheduling approaches

Scheduling approach	Scheduling type	Work
Sequential	Progressive	[Microsoft, 2017] [Akhshabi *et al.*, 2012b] [Rao *et al.*, 2011] [Johansen *et al.*, 2009] [Jiang *et al.*, 2014] [Huang *et al.*, 2013]
	Periodic	[Akhshabi *et al.*, 2012a] [Li *et al.*, 2012a] [Gautam *et al.*, 2013] [Rao *et al.*, 2011] [Kupka *et al.*, 2012] [Villa *et al.*, 2012] [Jiang *et al.*, 2016]
Parallel	MultiPath TCP	[Kurosaka and Bandai, 2015] [Corbillon *et al.*, 2016] [Singh *et al.*, 2012] [Kuhn *et al.*, 2014] [Raiciu *et al.*, 2012] [Wang *et al.*, 2009] [Fairhurst *et al.*, 2017] [Han *et al.*, 2016] [Chen *et al.*, 2013] [Jurca and Frossard, 2007] [Deng *et al.*, 2014] [Hesmans *et al.*, 2013] [Honda *et al.*, 2011]
	Multi-homing	[Evensen *et al.*, 2011] [Johansen *et al.*, 2009] [Evensen *et al.*, 2010] [Kaspar *et al.*, 2010b] [Kaspar *et al.*, 2010a] [Evensen *et al.*, 2012]
	Application layer	[Nguyen and Cheung, 2005] [Kuschnig *et al.*, 2010]
	Multi-server (block based)	[Tian and Liu, 2016]
	Multi-server (SVC based)	[Zhang *et al.*, 2015b] [Zhou *et al.*, 2014]
	Multi-server (1 segment -> 1 request)	[Liu *et al.*, 2012a] [Liu *et al.*, 2011b] [Pu *et al.*, 2011]
	Multi-server (1 segment->N requests)	MS-Stream [Gouache *et al.*, 2011]

2.4.2.1 Sequential scheduling

The most basic of all sequential scheduling techniques used in HAS services is called progressive dispatch. In progressive dispatch, a request is made as soon as the download of a video segment is completed. This approach permits to aggressively

fill the content buffer. During the streaming start-up period or when the buffer level is below a predefined threshold, the progressive dispatch represents the most appropriate scheduling mode as it prevents buffer starvation.

Commercial players such as Microsfot Smooth Streamning [Microsoft, 2017], Netflix [Akhshabi *et al.*, 2012b], and YouTube [Rao *et al.*, 2011], as well as some non-commercial players (e.g., DAVVI [Johansen *et al.*, 2009], and FESTIVE [Jiang *et al.*, 2014] are known to use progressive dispatch scheduling. [Huang *et al.*, 2013] argue that in order for a client to get its fair share of the available bandwidth, progressive dispatch should be used for the download of all the available content representations, with the exception of the top video quality level.

Similarly to the video bitrate adaptation strategy that adjusts the video quality to the available resources, it is highly beneficial to have a flexible scheduling logic implemented at the client side. The periodic dispatch approach is one example where each new segment request is scheduled in a specified time interval after receiving the previous segment. In addition, periodic dispatch can be used to avoid buffer overflow [Huang *et al.*, 2013], or to save energy when streaming in mobile environments [Li *et al.*, 2012a; Gautam *et al.*, 2013]. Most implementations derive the time interval between requests from the segment size [Akhshabi *et al.*, 2012b; Rao *et al.*, 2011].

In [Kupka *et al.*, 2012], the authors investigate the performance of periodic dispatch. Results show that the use of periodic dispatch can cause a deterioration in TCP throughput. To adress this problem, the authors then suggest the use of longer video segments. Akhshabi *et al.* [2012a] look at the behaviour of periodic dispatch scheduling when multiple players are trying to compete on the same bottleneck link. They observed that periodic dispatch scheduling leads to instability, unfairness, and underutilisation of resources.

In [Evensen *et al.*, 2012], the authors tackle the issue of throughput unfairness between competing clients, and instability in the displayed quality for the periodic scheduling logic. Rather than using a fixed delay between each segment requests, a threshold is set to determine when a client operates aggressively or conservatively on this time interval. The authors tested their approach through simulations and found improvements in the network throughput utilization fairness among clients. In their later work, Villa *et al.* [2012] investigated a scenario where each of the competing sessions is assigned a fixed but unique inter segment request time. They report similar performance as in the case of randomized inter segment request time. However, applying this in a practical system will require having the video content

available in multiple segment sizes, which can dramatically increase storage overhead. In [Jiang *et al.*, 2014], the value of the target buffer level is randomised. In doing so, the scheduling is still periodic but individual periods are independent of the client's streaming session start time and the issues reported by Akhshabi *et al.* [2012a] are mitigated.

Avoiding rebuffering and providing the best possible video quality are the two main objectives of any bitrate adaptation algorithm [Garcia *et al.*, 2014]. However, live streaming sessions have an additional liveliness requirement that will skip late segments if the supposed stream playback time is too far behind its deadline. Additionally, the buffer of live streaming applications is necessarily small due to the fact that only a part of the entire content is generated at the time of playback. Consequently, live streaming has stricter deadlines. Periodic dispatch trades timeliness for efficiency in buffer management which makes it hardly applicable for live streaming. Contrarily, progressive scheduling is a greedy algorithm that may work well under the assumption that the ungenerated segments are not requested before they are available. Scheduling algorithms have been proposed to cope with deadline-misses in live streaming. The authors of [Kupka *et al.*, 2011] evaluate a series of segment streaming strategies, with each strategy being a possible outcome of a set of the combinations of the following four options (each option has two values so there are 16 possible combinations in the set). *First request:* a client may request the most recent segment or wait for next available one; *Play-out start time:* a client may start play-out immediately or delay the play-out; *Next request:* a client may request a segment before download finishes or before the play-out finishes; *Deadline miss handling:* a client may start playing from the beginning of the download or skip a portion equal to the deadline-miss from the download. The result of extensive experiments shows that the best combination is when a client requests the most recent segment in its first request, starts play-out immediately, dispatches requests before a play-out finishes and starts playing each segment downloaded from the beginning. This combination is found to avoid the synchronisation of client requests resulting in a high-quality content and short delays.

2.4.2.2 Parallel scheduling

The scheduling of requesting segments plays a critical role in the achievable video bitrate of HAS and the utilization of distributed resources. With the recent access of broadband connectivity for end-users, network congestion not only occurs in the last-mile networks but also in the middle of the content delivery networks. Effec-

tively utilizing the distributed network resources is one of the key factors for HAS in providing high streaming QoE to end-users. Obvisouly HAS sequential scheduling methods cannot provide optimum streaming services when network congestion occurs in the content delivery networks. A number of contributions have attempted to improve the delivered throughput to end-users through various parallel scheduling techniques at the transport and at the application layer. While parallel scheduling is employed when a client has multiple network interfaces and uses some or all the interfaces simultaneously to improve the utilization of one or multiple access networks, it is also beneficial when a client is downloading content from multiple servers using one or many connections.

Transport layer based approaches and multi-homing

Many Internet service providers are capable of offering a number of independent paths, intra and inter-domain between two nodes. Also, many end-hosts today have multiple network interfaces (such as cellular and wireless interfaces on mobile devices), offering the opportunity to use multiple endpoints to communicate via multiple paths, allowing multi-homing streaming.

There are several transport protocols that have been developed to use multiple network paths, such as SCTP that uses multiple interfaces for redundancy/fail-over purposes and Multipath TCP (MPTCP) [Ford *et al.*, 2012] that offers parallel usage of multiple paths for resource pooling [Wischik *et al.*, 2008]. Although both protocols are designed to load balance data bulk transfers, MPTCP is gaining interest in the video streaming research community. In MPTCP, a scheduler assigns each packet from the MPTCP output queue to one available TCP sending buffer. Packet losses occurring on one path can generate head-of-line blocking at the client side, potentially leading to streaming buffer starvation. Many papers addressed this issue by focusing on windows congestion management [Kuhn *et al.*, 2014], cross-layer scheduling [Corbillon *et al.*, 2016], bandwidth and buffer management [Kurosaka and Bandai, 2015] and retransmission processes [Raiciu *et al.*, 2012] at the transport layer.

Wang *et al.* [2009] investigate a scenario where a client streams over multiple paths, which may or may not share a bottleneck link. Their goal is to determine under what conditions multipath TCP provides satisfactory performance, and what are the potential benefits of using multiple TCP connections. Their results show that the use of multiple TCP connections provides satisfactory performance when the achievable aggregated throughput is 1.6 greater than the video bitrate with few

seconds of start-up delay. Another result of Wang *et al.* [2009] shows that with proper design the aggregate throughput of the multiple paths can not only equal but also exceed the sum of the throughput of the individual paths.

In a recent work, Corbillon *et al.* [2016] exploited the interactions between the application layer and transport layers for MPTCP [Fairhurst *et al.*, 2017] to support video streaming. Hence, they introduced a cross-layer scheduler, which leverages information from both application and transport layers to re-order the transmission of data and prioritize the most important part of the video. Performance evaluation based on traces aggregated from real MPTCP sessions (on Ethernet, WiFi and cellular accesses) showed that the cross-layer scheduler improves the achieved video bitrate, but still has efficiency limitations. However, the authors assume that the adaptation logic has already selected the video segment representations for delivery over MPTCP. Thus, the issues inherent to quality adaptation are not considered in this work.

Another recent work by Han *et al.* [2016] mitigates this issue by proposing Multipath DASH (MP-DASH), a multipath framework for HAS with awareness of the end-users' network interface preferences. The proposal does not focus on improving the video streaming quality with MPTCP but aims at reducing the overall data consumption in the case of metered cellular network usage. The idea is to schedule data delivery to satisfy user preferences -preferring WiFi over cellular connection-. Their experiments conducted at 33 locations showed that MP-DASH can effectively reduce cellular usage up to 99%, and radio energy consumption up to 85% when compared with using MPTCP in the Linux kernel.

Chen *et al.* [2013] performed experimental measurements on different applications using single-path TCP, two-path MPTCP and four-path MPTCP. They studied the latency distribution, video prefetch size, block size and periodic retrieval time for Netflix and YouTube streaming using both Andriod and iOS devices and showed MPTCP can be reasonably used for video streaming. However, the authors of [Jurca and Frossard, 2007] showed by simulations some inherent issues and impacts on QoS when streaming video over multiple paths.

Using MPTCP for HAS has its downsides and very few contributions focus on running HAS over MPTCP. This is partly due to the performances of MPTCP that are below expectations especially under heterogeneous network characteristics, eventually leading to QoE degradation for the end-users [Deng *et al.*, 2014; Corbillon *et al.*, 2016; Singh *et al.*, 2012; Jurca and Frossard, 2007]. The lack of MPTCP support in middleboxes [Hesmans *et al.*, 2013; Honda *et al.*, 2011] is another expla-

nation.

In [Priyadarshini and Rekh, 2016], the authors investigated the problems of using multi-homed terminals to stream video on mobile devices in a hetereogenous wireless network. Although not directly relevant to the adaptive mechanisms used in HAS, they studied the behavior of mobile video over multiple communication paths. They developed an analytical framework for modeling MPTCP-based video delivery and proposed ADMIT (quAlity-Driven MultIpath TCP) which uses a utility maximization based Forward Error Correction (FEC) coding and throughput allocation to achieve optimal quality for real-time streaming. Their experiment results show that ADMIT improves video quality in terms of PSNR, and the benefits are more obvious when the number of access networks increases.

The work of Evensen *et al.* [2011] is another attempt in efficiently aggregating available bandwidth from multiple heterogeneous network interfaces for quality adaptive streaming. In [Evensen *et al.*, 2010], a similar approach but within a HAS is proposed by extending DAVVI [Johansen *et al.*, 2009] with multi-homing capability. Experimental results show that the scheduler reduces video interruption and improves average video quality, even when using heterogeneous multiple wireless interfaces.

In [Kaspar *et al.*, 2010b], HTTP range retrieval is used to sequentially request segments from a multi-homing capabilities. The HTTP range retrieval request is used to logically divide a video segment into multiple sub-segments of fixed size. The performance of the proposed technique is found to be dependent on the segment size. The authors explain that using small segments increases the overhead, which results in reduction of aggregate throughput, while large segment size is found to significantly increase both the start-up delay and the buffer requirements. The authors conclude on two possible solutions, either to get an optimal segment size -imposing a trade-off between the throughput and the start-up delay- or to parallelise the scheduling. In their follow-up work, Kaspar *et al.* [2010a] study parallel scheduling based on a HTTP pipelining and multi-homing that allows a client to send a request without the need to wait for a response. The proposed technique interleaves byte range requests to then have each interface sending a request immediately after receiving a response. By ensuring that the server is always busy processing and responding to requests the efficiency of throughput aggregation is found to be close to the optimal level. However, the work was done in the context of progressive download instead of HAS. Additionally, optimal performance can only be achieved with a large buffer size to absorb the effects of link variability. In their

subsequent work [Evensen *et al.*, 2012], the parallel scheduling is improved by adjusting the sub-segment size in proportion to the estimated link capacity. By allocating the right amount of data to each link, idle periods are avoided and the buffer size requirement is lowered.

Application layer based approaches

In [Nguyen and Cheung, 2005], multiTCP is proposed as an application layer algorithm that improves resiliency against short-term TCP throughput fluctuations. The authors of the paper demonstrate that for any single packet loss event, the reduction in throughput when two TCP connections are used is four times less than if one TCP connection only is used. In summary, the amount of TCP throughput reduction is inversely proportional to the number of TCP connections employed.

In the work of Kuschnig *et al.* [2010] multiple HTTP/TCP connections are used to stream video content. The proposal is found to be insensitive to packet loss and therefore, reduces throughput fluctuation. The scheme works as follows: the client downloads a fixed number of segments from one server. Each segment is being requested in order of playback, but the segments with closer deadlines are prioritized. When the download of a segment reaches a timeout, the segment is retransmitted on two HTTP connections in order to increase the probability of successful transmission.

Multiple-server based approaches

The underlying principle of the works on MPTCP-based streaming is that by controlling the multiple TCP connections competing for the same resource, the throughput variation observed on each connections can be smoothed out, and the achievable TCP throughput can be improved. In contrast, multiple-source HTTP adaptive streaming with parallel segment scheduling aims at utilizing the distributed infrastructures of servers composing the content delivery networks (CDNs) so as to mitigate the issue of bandwidth bottleneck at the content delivery network side.

In content delivery networks, content is replicated to the multiple surrogate servers distributed in the networks. As the surrogate servers are close to the end users, CDNs have the advantages of saving bandwidth and reducing the delays perceived at the receiver side. However, due to expensive network resources in the CDN, the achievable bandwidth for media segments delivery is still limited. Consequently, parallel and multiple-server HAS is an essential and important feature for enabling clients to effectively make use of distributed network resources.

The work of Adhikari *et al.* [2012c] on understanding and improving multi-CDN delivery concludes that the QoE of end-users would greatly benefit from the advent of a practical HAS that actually utilizes multiple servers simultaneously. However, without appropriate scheduling, using multiple servers does not necessarily guarantee a high quality streaming service.

Originally designed to perform with single-source HAS, the quality adaptation decisions in [Tian and Liu, 2016] impose segments to be retrieved sequentially. As a result, when applied to multiple servers streaming, it causes frequent fluctuations of playback video bitrate. Moreover, fragments of video are requested from multiple servers one after another without considering their completion time. Thus, the expected completion time is generally exceeded due to the channel heterogeneity of multiple-source environemnts. Oppositely, the examined rate adaptation logic in [Zhou *et al.*, 2014] downloads blocks of video data (not segments) simultaneously from different servers. Besides, requests are scheduled to multiple servers according to their playback deadline priorities so as to guarantee their completion time in order. The work of Zhang *et al.* [2015b] presents a streaming proposal using several servers in parallel. The authors define streaming fairness (fair throughput usage between client on the same link), efficiency (highest possible video bitrate delivered) and stability (minimization of the video rebuffering events) before providing a system to improve these three criteria. However, the proposed approach lacks of segment scheduling that leads to low QoE performances when the used paths have heterogeneous conditions. A multi-source evolution of HAS is introduced in [Pu *et al.*, 2011], employing the Scalable Video Coding (SVC) technique that imposes dependency between layers and requires a great care in the design of the layer scheduler to prevent video stalls.

In contrast to the approaches that use the HTTP range retrieval, the contributions [Liu *et al.*, 2011b, 2012a, 2011a] propose a scheduling scheme that enables a client to request multiple segments simultaneously using independent HTTP sessions. The scheduler first sends a HTTP GET request to a server. While receiving the requested segment, it dispatches other requests for other segments. The scheduler aims at maintaining a limited number of parallel HTTP connections and determines when to start the next segment request. No additional request is sent when the following two conditions holds: (1) the maximum allowed parallel sessions have been reached, (2) the buffer level is equal or greater than a pre-defined upper bound. In [Gouache *et al.*, 2011], an attempt to improve the resilience of the system by concurrently downloading segments from multiple servers are made. The scheme

continuously estimates the bandwidth of each stream from all servers. A software agent decides which representation will be requested based on the smoothed version of the bandwidth estimate. The agent requests a slice of the chosen video segment from each server simultaneously in proportion to the estimated capacity of the corresponding server. The result of their experiment shows a reduction in the video rate variability at no extra bandwidth.

2.5 Hybrid P2P/CDN solutions

This section explains the delivery of video content over the Internet with Content Delivery Networks (CDNs) and with adaptive P2P streaming technologies before exposing some of the research and industry work on hybrid P2P/CDN content delivery. Then, we review the different challenges of hybrid P2P/CDN content delivery to improve the quality of streaming services and the end-users' QoE. Although such hybrid solutions have the advantage of high reliability and reduced scalability costs, very few research addressed the challenges of combining HAS with P2P/CDN video delivery to benefit from QoE improvements.

2.5.1 Content delivery networks

Content Delivery Networks (CDNs) have been considered as the main approach for video distribution over the Internet. CDN servers are geographically replica/cache/edge servers positioned as close as possible to the consuming clients. When accessing a content, consuming clients are automatically redirected to one of the best available servers based on proximity -or other parameters- so as to temper network congestion and achieve higher throughput. Figure 2.8 provides a high level understanding of CDNs.

With the ever-increasing amount of video traffic, CDN has been facing several challenges related to managing and administrating the entire CDN infrastructure, such as the support for HTTP-based video delivery, the scalability problem [Balachandran *et al.*, 2013], replica placement [Pathan and Buyya, 2008], content selection [Gao *et al.*, 2015; Jin *et al.*, 2016; Scellato *et al.*, 2011], and content placement [Applegate *et al.*, 2010].

CDNs are available from companies like Limelight, Akamai, and Level 3, but recent years have seen the rise of CDN services hosted by big companies such as Google, Facebook and Microsoft. To provide a cheap pay-as-you-go service to a

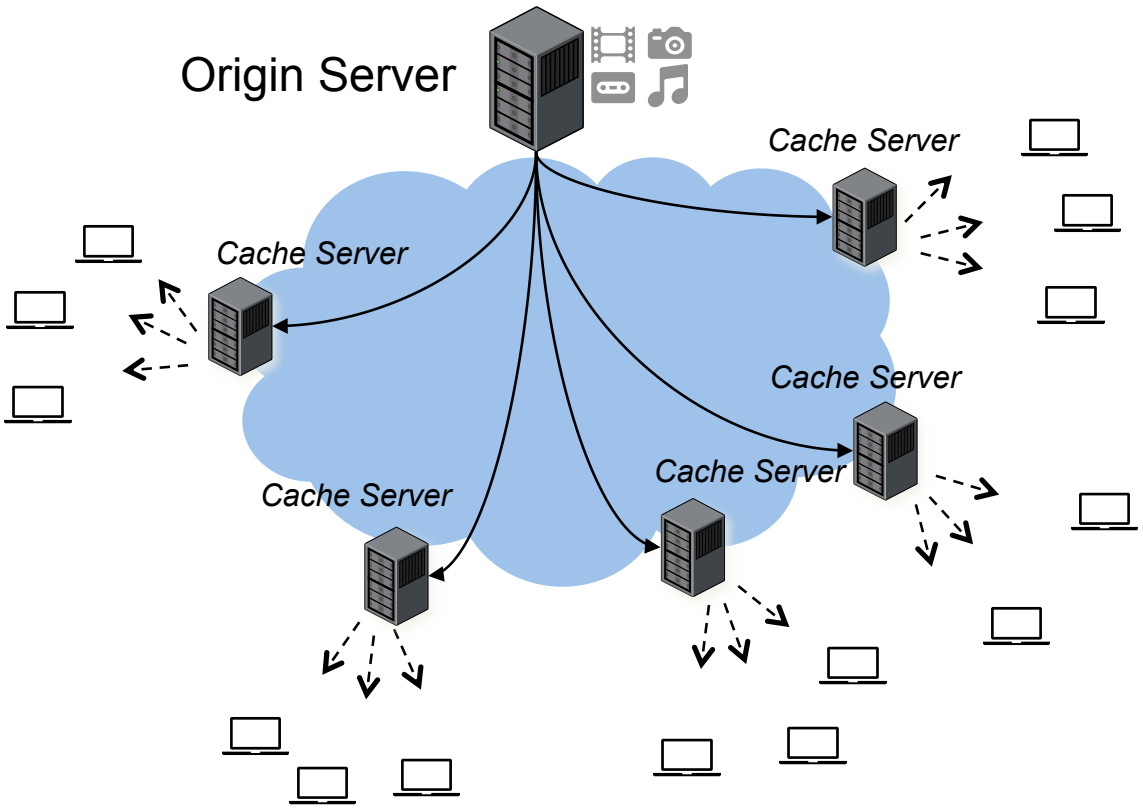

Figure 2.8 – High level understanding of Content Delivery Networks

broad variety of customers some CDNs have adopted cloud technologies which became known in the literature as cloud CDNs [Limelight, 2017]. Futhermore, in order to gain a better control over the data services served to their end-users, many telecom operators (AT&T, Orange, Telefonica, Verizon, etc.) have deployed their private telco-CDNs [Frank *et al.*, 2013]. Overall, Cisco has estimated that content delivery network traffic will handle nearly two thirds of all Internet video traffic by 2020 [Cisco, 2016].

There have been a number of studies on CDN-based VoD systems (e.g. YouTube [Adhikari *et al.*, 2010, 2012b], Netflix [Adhikari *et al.*, 2012c] and Hulu [Adhikari *et al.*, 2012a]). However, several recent studies have reported that CDNs are being stressed by the demands during peak hours [Liu *et al.*, 2012c; Wendell and Freedman, 2011], pointing out the limitation on CDN scalabilty. Subsequently, many efforts have been put on augmenting traditional CDN architectures with P2P networks [Balachandran *et al.*, 2013; Huang *et al.*, 2007; Karamshuk *et al.*, 2015; Ruckert *et al.*, 2014].

2.5.2 P2P and Adaptive P2P streaming

P2P systems [Camarillo, 2009] represent an alternative to CDNs because they do not rely on a highly reliable fixed-size streaming infrastructures. On the contrary, they utilize volatile and heterogeneous resources to achieve video streaming. In a P2P system, peers act as both clients and servers, and participate to the streaming system by sharing their resources -e.g., storage and network bandwidth- with the rest of the P2P community to increase content availability and streaming performance. Because clients dedicate their resources to serve requests from their peers, a P2P system can in principle scale arbitrarily without a need of dedicated servers [Deng and Xu, 2013]. Therefore, in contrast to CDNs, P2P systems permit considerable savings on infrastructure. However, many weaknesses have been identified in P2P networks such as peer churn -peers frequently and suddenly leaving or joining the system due to network failures or based on their own decisions- [Stutzbach and Rejaie, 2006] and heterogeneous resources among participating peers that can rapidly result in poor streaming performance, high startup delay and frequent video playback disruptions [Huang, 2008].

Two main types of architectures are generally considered for video steaming services: tree-based overlay for streaming sessions from media sources to a pool of client peers; and mesh-based overlay for massive parallel content distribution among peers. In tree-based overlays, peers are organized in a tree structure (sometimes multiple-tree structure) where clients are leaf nodes. Content is usually pushed from the root of the tree to the consecutive levels of the tree until it reaches the end leafs. Although the tree-based approach is simple and easy to control, it can be severely affected by peer churn [Stutzbach and Rejaie, 2006]. In addition, the received content quality is limited by the minimum upload bandwidth of the intermediate peers, since each client is connected to the source through a single tree branch. Multiple tree architectures address the latter problems by providing redundancy in network paths. However, designing and maintaining such systems is very challenging and may even lead to solving contradictory issues such as minimizing tree depth, while simultaneously provisioning network path diversity.

In mesh-based systems, peers connect to a random set of neighboring peers that watch the same content. Peers usually exchange information about the previously cached data to then retrieve missing pieces of videos from their neighbors. The advantages of such organizations reside in the low cost and simplicity of design and maintenance. Besides, because each peer maintains a set of neighbors at any point in time, the probability of having multiple available distinct network paths for the

same content is increased. Hence, the mesh-based approach is much less susceptible to peer churn than tree-based P2P.

A multitude of surveys on P2P technology have presented critical analysis on different design features and infrastructural properties of P2P systems and their influence on non-functional aspects such as scalability, resource management, security, fairness and self-organization. Also, several studies have been conducted on real-world deployment of P2P-based VoD systems (e.g. PPLive VoD [Huang, 2008], Joost [Lei *et al.*, 2009], CoolStreaming [Zhang *et al.*, 2005; Xie *et al.*, 2007]). Nevertheless, if peers can simultaneously request missing video pieces from several neighbors, the resources they share with the rest of the P2P network are highly heterogeneous and volatile, which can rapidly result in video stalls and low QoE. A comprehensive survey of various techniques proposed for structured and un-structured P2P networks has been presented in [Lua *et al.*, 2005]. Similarly, the authors of [Liu *et al.*, 2008] provided an overview of different approaches to address chunk scheduling techniques and peering mechanisms. Multiple adaptive streaming techniques have also been proposed in P2P streaming systems [Jurca *et al.*, 2007]. Layered video encoding has been used to adaptively deliver different layers of the video to the clients. Multiple Description Coding (MDC) and network coding has also been used to propose adaptive streaming systems that support a large number of users [Xie *et al.*, 2007; Zhang *et al.*, 2005; Castro *et al.*, 2003; Li *et al.*, 2007]. A lot of efforts have been put into the field of pull-based P2P SVC streaming systems in order to deliver the best possible video quality by optimizing overlay structuring and data scheduling [Medjiah *et al.*, 2014; Moon *et al.*, 2013; Xiao *et al.*, 2009; Liu *et al.*, 2009; Eberhard *et al.*, 2010; Capovilla *et al.*, 2010].

2.5.3 Improving QoE and scalability with hybrid P2P/CDN streaming

The central idea behind hybrid P2P/CDN streaming is to combine the benefits of two different technologies for content distribution: traditional server-based CDNs, and P2P networks. Figure 2.9 illustrates the main idea of hybrid P2P/CDN streaming. Traditional CDNs rely on professionally and geographically distributed high-end infrastructures. CDN servers can therefore be expected to be highly reliable and available, and are engineered to provide a high quality of service, often assisted with service-level agreements (SLAs) between CDN providers and content owners. However, from an economic perspective, traditional CDNs require significant investments

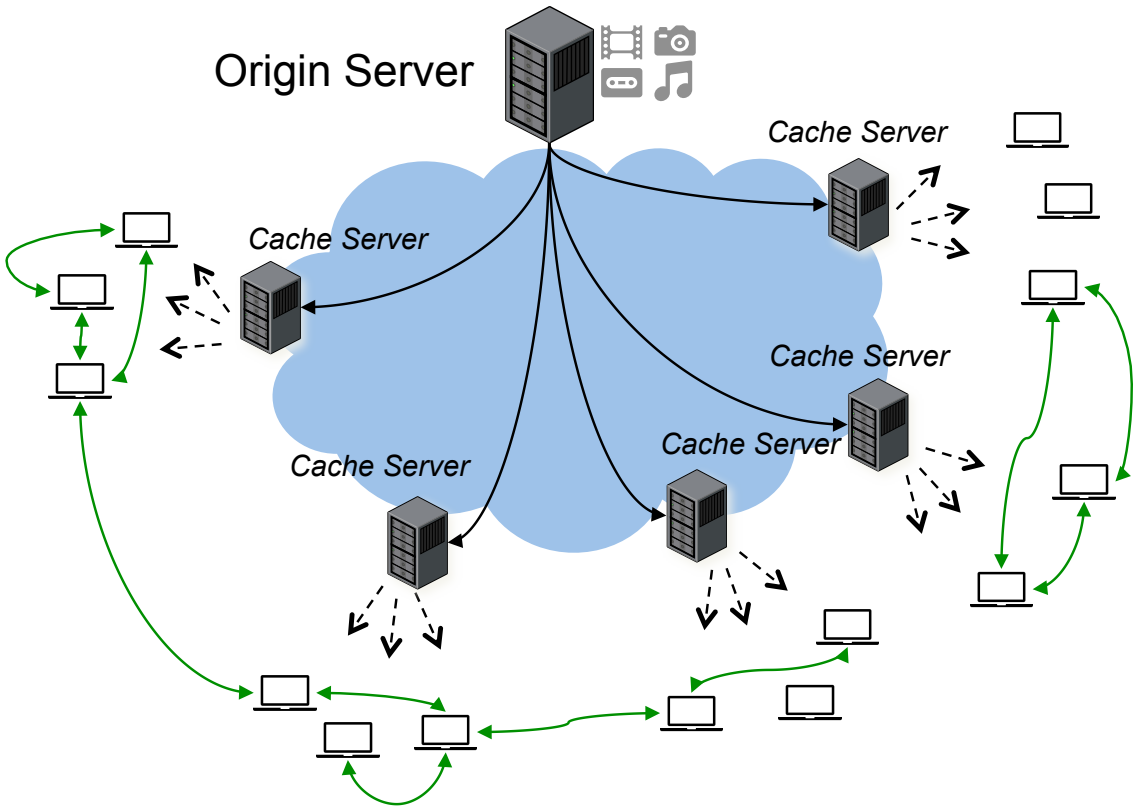

Figure 2.9 – High level understanding of hybrid P2P/CDN

for scaling up, as they require the deployment and management of geographically distributed data centers [Kim *et al.*, 2015]. Scaling up is precisely the strength of P2P content delivery. P2P systems possess the self-scaling property [Menasche *et al.*, 2009; Qui, 2013], as each user downloading content also adds new capacity by acting as a server for other users. However, obtaining content through P2P systems has proven to be unreliable due to peer churn and to peers' resources heterogeneity [Kaune *et al.*, 2010]. The hybrid P2P/CDN streaming approach permits, on the one hand, to offer scalability with the P2P approach and, on the other hand, to compensate the bandwidth imbalance and the churn of P2P systems by relying on the CDN servers. The great advantages of such solutions have been demonstrated analytically [Xu *et al.*, 2006a; Lu *et al.*, 2012], by simulation [Chellouche *et al.*, 2012; Huang *et al.*, 2008], and by large-scale deployment [Zhang *et al.*, 2015a; Yin *et al.*, 2009, 2010].

In most hybrid P2P/CDN streaming, content is delivered from the CDN servers to a minor subset of the peers composing the P2P overlay [Peterson and Sirer, 2009; Roverso *et al.*, 2011; Zhao *et al.*, 2013]. In the mean time, the peers engage in data

exchange with their neighbors in order to distribute the received video segments from the CDN servers. When the capacity of the peers to re-emit the received data is insufficient, the clients are served directly from the CDN servers. Thus, hybrid P2P/CDN solutions can significantly offload the bandwidth requirements of CDN infrastructures, as reported in feasibility studies for a number of large VoD vendors (BBC iPlayer [Karamshuk *et al.*, 2015], Conviva [Balachandran *et al.*, 2013], and MSN Video [Huang *et al.*, 2007] where 50% to 88% of all consumer traffic is achieved with P2P data exchange. Recently, hybrid P2P/CDNs have also been deployed by commercial CDNs, including Akamai [Zhao *et al.*, 2013], ChinaCache [Yin *et al.*, 2010], and Xunlei KanKan [Zhang *et al.*, 2015a].

Reducing scalability costs (i.e., increasing bandwidth savings) while at the same time providing the best possible quality of streaming service or QoE to end-users still expose several technical challenges yet to be addressed if industrial actors are to adopt such hybrid streaming solutions: streaming discontinuity due to peer churn, the lack of mechanisms to cope with the heterogeneous resources of participating peers, the inaccessibility of peers behind NAT and firewalls, the lack of quality adaptation tool. Not surprisingly, the majority of research efforts in hybrid P2P/CDN content delivery literature have focused on developing strategies for improving quality of service in terms of reducing startup delay and playback delay [Xu *et al.*, 2006b; Li *et al.*, 2012b; Lu *et al.*, 2011; Li *et al.*, 2012c; Kim *et al.*, 2011; Hu *et al.*, 2012; Ha *et al.*, 2011].

2.5.3.1 Architectures

Hybrid P2P/CDN is a hybrid architecture which aims to combine the advantages of both CDNs and P2P systems. On the one hand, hybrid P2P/CDNs rely on contributions of content delivery resources from the end-users and, hence reduce the infrastructural costs of traditional CDNs. On the other hand, hybrid P2P/CDNs reduce the drawbacks of pure P2P systems by relying on the guaranteed content and resource availability of CDNs. In this context, the most challenging issues are related to integrating traditional CDNs and P2P while at the same time benefiting from the advantages of both. The variety of the hybrid P2P/CDN content delivery architectures can be classified into two main categories: centralized and decentralized architectures.

Centralized architecture: in centralized hybrid P2P/CDNs, new clients register themselves at the nearest CDN server and store contextual information such as IP addresses, open ports, consumed content. The latter information is stored into a

centralized database. The CDN server then forwards the first video segments to the clients along with a list of randomly selected peers from the P2P overlay. In the event of insufficient number of peers composing the overlay or in case of high peer churn reported by the peers, the server refreshes the list of peers. Hence, this CDN server not only acts as a content server but also as a tracker. Also, if the P2P data exchanges fail to deliver video segments in time, the server is used as a back-up solution. The centralized hybrid P2P/CDN architecture has several advantages over pure P2P systems. The quality of service is highly reliable in centralized hybrid P2P/CDNs due to high content availability at the edge servers. Additionally, in a centralized hybrid P2P/CDN architecture, the CDN is in charge of the P2P overlay, thus easing its management. Centralized architectures have been implemented in the vast majority of commercial hybrid P2P/CDNs [Zhao *et al.*, 2013; Kim *et al.*, 2011; Zhang *et al.*, 2015a].

Decentralized architecture: in decentralized hybrid P2P/CDN solutions, the P2P overlay is managed by elected tracker-peers (or super-peers) that are selected based on multiple criteria, such as their proximity to the CDN server, stability or network and processing capabilities. When a new client joins in the decentralized hybrid P2P/CDN, it contacts an elected super-peer to obtain a list of active peers consuming the desired video content. The request is redirected to the nearest super-peers until a super-peer holding such a list is found. The super-peers also act as entry point for the content to arrive in the P2P overlay. When the amount of peers downloading a given content is unsufficient, the super-peers are responsible for retrieving the consumed content from the CDN servers. Decentralized architectures have the advantage of minimizing the required CDN infrastructure by delegating most of the P2P overlay management functions to the clients. Nevertheless, they become more vulnerable to malicious attacks and are more difficult to manage in comparison with the centralized approach.

2.5.3.2 Heterogeneity of resources

Heterogeneity in resources is one of the main obstacles in implementing hybrid P2P/CDN content delivery. Indeed, the multiple Internet connectivity types that can be found in the large number of distributed end-systems present various upload and download bandwidth capabilities that directly affect the efficiency of the P2P data exchange. Moreover, the asymmetry between upload and download bandwidths of last-mile Internet connections is another issue for video streaming in many P2P applications such as PPlive and Maze networks. For instance, a downstream rate

of ADSL2+ can represent up to 15 times the upload network throughput, exposing a potential bottleneck in uploading bandwidth when streaming a high resolution video content from peers [Baccelli *et al.*, 2013].

The impact of the limited upload bandwidth on hybrid P2P/CDN streaming has been evaluated in [Huang *et al.*, 2007]. The authors demonstrated that although the demand for a high resolution video content is growing over time, its growth rate is in fact smaller than that of the average upload bandwidth among Internet users. This indicates a very positive trend for the future of P2P/CDN streaming solutions in delivering high QoE streaming services, suggesting that the future upload capacities of the end-users' devices would allow for widespread P2P video sharing. In order to improve the quality of video streaming service while optimally exploiting throughput capacity of peers to reduce servers' participation to the streaming, [Huang *et al.*, 2007] suggested two different peer selection policies: Water Levelling (WL) and Greedy Policy (GP). With the WL approach the extra uploading bandwidth is uniformly distributed among all peers. In the GP approach, each client simply dedicates its remaining upload bandwidth to the nearest peer. Simulation results show that in the case of P2P/CDN solution with the greedy policy, the server bandwidth requirements could be reduced from 2.20 Gbps to 0.79 Gbps.

Furthermore, as in P2P streaming, peer churn is also an important factor for the quality of hybrid P2P/CDN solutions. To tackle the negative impact of peer churn on the performance of hybrid P2P/CDN delivery systems, the authors in [Chen *et al.*, 2015] proposed a crowd sourcing-based content distribution system called Thunder Crystal. The *thunder crystal* approach relies on the smart Access Points (APs) distributed among clients. A smart AP is equipped with a large storage for caching high resolution videos and trades content with peers in exchange of some benefits obtained from the content provider (e.g., discounted subscription). Similarly, Youku, one of the largest VoD services in China, has deployed millions of dedicated smart peer routers (i.e., set-top-boxes) with 8GB storage capacity in consumers' homes and offices to assist content distribution [Ma *et al.*, 2016]. Based on a trace driven evaluation of the YouTube traffic, the authors in [Nicolas *et al.*, 2013] proposed a P2PTube approach which relies on exploiting set-top-boxes to assist content distribution. As the results from the [Nicolas *et al.*, 2013] suggest, up to 46% of videos can be served from peers in the proposed system, an advance of 10% with respect to the previous result reported in [Zink *et al.*, 2009]. The authors in [Gramatikov and Jaureguizar, 2015] exposed a mathematical model to evaluate the impact of peer-churn on a hybrid P2P/CDN, when the users of set-top-boxes

are not willing to share their resources. Similarly, in order to deal with peer churn, the authors in [Markakis *et al.*, 2017] proposed a Home Box-assisted approach which relies on exploiting set-top-boxes as proxies between end-users and CDNs.

2.5.3.3 Quality of Service and Quality of Experience in hybrid P2P/CDN solutions

In contrast to CDNs, P2P systems inherently suffer from various problems which impact quality of service, including peer churn, partial participation, and heterogeneity in resources. The delivery of high quality video contents to the end-users is among the most challenging issues for many network and content delivery services [Bobarshad *et al.*, 2012, 2010; Liu *et al.*, 2008; Passarella, 2012].

Video startup delay

To achieve a lower startup delay, the authors in [Xu *et al.*, 2006b] proposed a three-phase streaming process for hybrid P2P/CDN architecture. In this scheme, peers download some initial segments from a geographically close CDN server and the remaining segments from the P2P network. The authors of [Lu *et al.*, 2011; Ha *et al.*, 2011] proposed a strategy for improving the startup delay via an effective buffer management at the client side. In order to minimize the video startup delay, Ha *et al.* [2011] argued that a higher priority should be given to the first video segments to be displayed. In the proposed scheme, the video buffer is divided into two zones. When the buffer level is close to reach the emergency zone, the video segments are retrieved from CDN nodes, otherwise the segments are served from peers. The relative size of the two zones depends on the number of CDN nodes available in the system and on the number of participating peers. A newly arrived peer will firstly approach the CDN server to fill the buffer priority zone as fast as possible before engaging into P2P data transfer.

Similarly, Lu *et al.* [2011] suggested organizing the playback buffer into three different regions with a startup region and a common region equivalent to the ones proposed earlier in [Ha *et al.*, 2011]. However, an additional emergency zone is introduced when a peer fails to download the required segments from neighboring peers before a given deadline. Simulations showed improvements of 2.5 sec in the average startup latency in comparison to the traditional peer-to-peer networks.

The authors of [Hu *et al.*, 2012] suggested to better utilize the peers' resources to provide lower startup delay and continuous data transmission. The authors proposed a resource scheduler relying on three types of queues to deliver video segments from

the servers before their playback deadlines. When a new peer joins in the streaming system, its requests for video segments are placed in a first queue following the first-come-first-serve discipline. The requests from this queue are sent to the server. Then, the server delivers a few initial video segments to the peer for immediate playback while the peer initiates P2P communications to obtain the remaining segments. If a peer cannot receive a desired chunk from its neighboring peers before the playback deadline, its request is handed over to the servers and is placed in a second queue. In this queue, the delivery of each chunk is urgent, but the playback deadlines of these chunks are different. Therefore, the earliest playback-deadline first discipline is employed. In the third queue, all the requested chunks have non-urgent playback deadlines and the first-come-first-serve discipline is adopted as the chunk scheduling strategy. Simulation results showed that in comparison to the previous approaches, the proposed scheduling mechanism can reduce the average startup delay from 21.36 seconds to 20.45 seconds.

Video playback interruption

Aside from minimizing the video startup delay, optimizing the delivered throughput represents another challenging issue to sustain uninterrupted video playback.

A few solutions have been proposed in the literature to minimize the video freezing events [Xu *et al.*, 2006b; Kim *et al.*, 2011]. The authors of Kim *et al.* [2011] proposed a Group Based CDN-P2P (G-CP2P) decentralized hybrid P2P/CDN architecture, which reduces service disruption latency with a location/content-aware peer selection mechanism. In the G-CP2P strategy, peers join different P2P groups based on their latency. Each group is controlled and managed by a special peer refered to the super-peer, selected based on its distance from the CDN edge server. The super-peer exploits a distributed hash table algorithm (DHT) [Ratnasamy *et al.*, 2001] to locate the content in the P2P network. When a peer is selected as a super-peer, its round trip time (RTT) is used as a key such that all super-peers are arranged and partitioned in increasing order of RTT, hence composing several separate clusters. Each new peer joins a super-peer which has the same order of RTT and joins the group within the same cluster. The authors compared their approach to the work of Xu *et al.* [2006b]. Simulation results conclude that the location/content-aware mechanisms permit an average decrease of 0.5 sec in the start-up delay, and allow to reduce the delivery time of each chunk of 0.5 to 4.5 sec.

Ameliorating QoE in hybrid P2P/CDN solutions with HAS techniques

A few other works focus on improving the QoE in hybrid P2P/CDN systems with HAS techniques. The research work of Tian *et al.* [2013] and Lederer *et al.* [2012] are among the first to propose jointly using the P2P and HAS techniques. The contribution in [Lederer *et al.*, 2012] proposes a pragmatic standard compliant solution to integrate peer-assisted streaming in conventional DASH client-server systems. However the flexibility in using neighboring peers is restrained in downloading a given segment from one peer only. In [Lederer *et al.*, 2012], the authors consider the P2P feature as a secondary and beneficial add-on to the system, and do not take into account the heterogeneity of peers' upload and download capacities in consuming and re-emitting their downloaded content. In [Tian *et al.*, 2013], the DASH technique is applied to a P2P architecture and exploits game theory, pricing models and the network resources of each peer to govern the quality adaptation decisions.

A recent work [Merani and Natali, 2016] on quality-adaptive P2P streaming system is among the first to investigate quality adaptation in P2P systems without relying on media or network coding. The proposed bitrate adaptation logic in [Merani and Natali, 2016] is based on global indicators of the streaming system's behavior and on the peers' download capacities. However, Merani and Natali [2016] take into account neither the variation of peers' download/upload capacities in time nor the QoE level obtained at the end-users' side. Additionally, this work mostly investigates the quality adaptation for a system almost entirely composed of peers. In their design, the contributions of servers to the streaming session are very limited and are only required to deliver a few segments to some peers responsible for spreading the data in the P2P overlays.

In summary, our analysis in this section suggests that the most promising trade-off in mixing the advantages of P2P and CDNs to improve QoS or QoE has been boot-strapping video streaming with initial segments downloaded from CDN servers and delegating the remaining work to peers [Lu *et al.*, 2011; Ha *et al.*, 2011]. Similarly, the peer-assisted content delivery system benefits from handing over the delivery of video to the servers in emergency cases when there is no sufficient upload bandwidth available from the peers [Xu *et al.*, 2006b].

2.6 Conclusions

To summarize this chapter, we reviewed the existing technologies used for delivering video content over the Internet to large number of end-users on heterogeneous

Table 2.4 – Summary of streaming strategies and their QoE improvement mechanisms

Content Delivery Strategy	Adaptation Type	Work
CDN	HAS	see sections 2.3 and 2.4
P2P	Media Coding (MDC, SVC, FEC)	[Jurca *et al.*, 2007] [Xiao *et al.*, 2009] [Liu *et al.*, 2009] [Xie *et al.*, 2007] [Zhang *et al.*, 2005] [Castro *et al.*, 2003] [Li *et al.*, 2007] [Medjiah *et al.*, 2014]
	Overlay management	[Medjiah *et al.*, 2014] [Moon *et al.*, 2013] [Xiao *et al.*, 2009] [Liu *et al.*, 2009]
	HAS	[Merani and Natali, 2016]
Hybrid P2P/CDN	Scheduling	[Xu *et al.*, 2006b] [Lu *et al.*, 2011] [Ha *et al.*, 2011] [Ha *et al.*, 2011] [Lu *et al.*, 2011] [Hu *et al.*, 2012]
	Overlay management	[Chen *et al.*, 2015] [Ma *et al.*, 2016]. [Nicolas *et al.*, 2013] [Gramatikov and Jaureguizar, 2015] [Markakis *et al.*, 2017] [Kim *et al.*, 2011] [Ratnasamy *et al.*, 2001]
	HAS	[Lederer *et al.*, 2012] [Tian *et al.*, 2013]

end-systems. On the one hand, P2P and multicast streaming solutions permit to lower the scalability costs. On the other hand, over-the-top content delivery infrastructures such as CDNs, and managed network infrastructure with end-to-end QoS provide high reliability for streaming services. We listed the research contributions to improve the end-users' QoE with content quality adaptation and content delivery adaptation for HTTP Adaptive Streaming. The latter adaptation mechanisms were derived for the three possible points of actions: at the client side, the server side or in the network. We exposed some of the existing work on adaptive P2P streaming and hybrid P2P/CDN solutions that benefit from the reduced scalability costs of P2P and the high reliability of CDNs. We also identified a lack of mechanisms to improve end-users QoE in such hybrid systems, especially when considering the advances made with HTTP adaptive streaming. Table 2.4 summarizes the existing video delivery strategies and their QoE improvement mechanisms.

The work of this thesis focuses on pragmatically extending the HAS capabilities to simultaneously utilize multiple servers and improve QoE. Investigating the adaptation mechanisms involved in such systems is also necessary to understand the

extent to which QoE can be improved. The second goal of this thesis is to further enhance multiple-server adaptive streaming solutions by combining it with adaptive P2P streaming and realize the first proposal of hybrid P2P/multi-server HTTP adaptive streaming. In view of the exposed related work, we can now position the work of this thesis (i.e., MS-Stream, MATHIAS and PMS), illustrated on the Venn diagram in Figure 2.10.

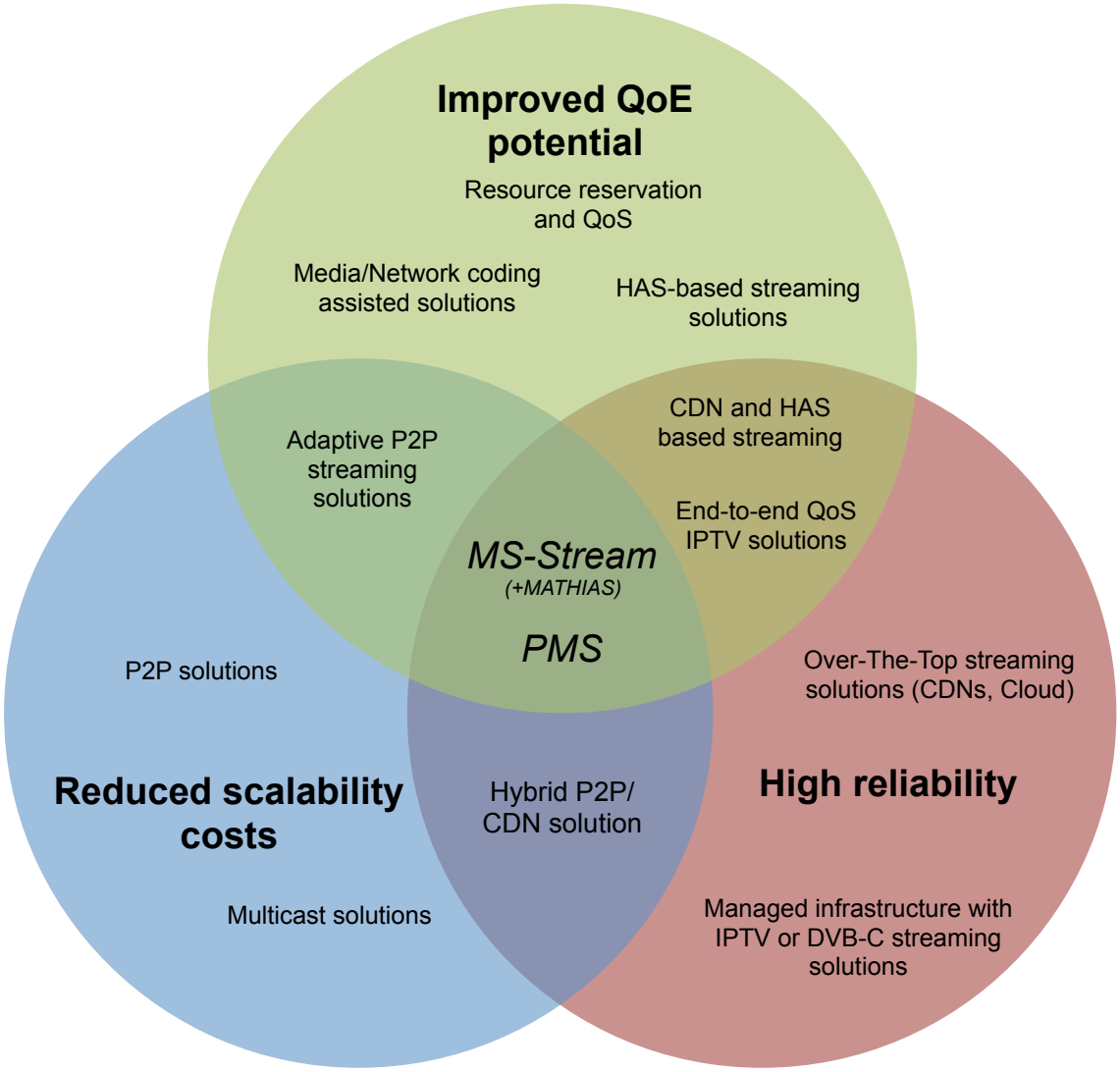

Figure 2.10 – Venn diagram of the streaming features (improved QoE potential, high reliability, reduced scalability costs) and some of the existing solutions to achieve them.

Chapter 3

MS-Stream: Toward a pragmatic evolution of client-centric HAS techniques in supporting multiple-source streaming

"There's a way to do it better - find it"

— Thomas A. Edison

3.1 Introduction: The multi-source adaptive streaming over HTTP approach

The demand on network resources is growing every day driven by the needs of end-users to consume more content over the Internet. According to Cisco's white paper [Cisco, 2016], video traffic will experience a tremendous growth and is expected to exceed 82% of the total Internet traffic by 2020. Most of the time, this increase is not followed by the necessary upgrade of core networks' capacity due to the important costs it incurs. This poses powerful challenges on the quality and scalability of the offered services. Alternative solutions (i.e., CDNs) propose to overcome these challenges by providing a single intermediate node with high throughput capacity and geographically close to clients. When accessing a content, consuming clients are automatically re-directed to the closest server so as to temper network

congestion and achieve higher throughput. In addition to the CDN-based infrastructure, streaming services usually rely on HAS techniques where the consuming client uses a single server and dynamically adjusts the requested content bitrate according to the estimated network conditions. Compared to non-adaptive streaming protocols, the adaptability and flexibility in HAS enable higher end-users' QoE by avoiding video freezing events. However, these solutions are not taking into account the case when a large amount of end-users located under the same geographic area is simultaneously consuming streaming services, the nearest server may become rapidly overloaded. As a result, consuming clients will suffer throughput degradation and thus experience a poor QoE due to content and throughput bottlenecks at the content delivery network side [Khan Pathan and Buyya, 2010]. Although the CDN solutions based on the single-server HAS approach can handle large volumes of requests, they laboriously adapt to the highly dynamic and volatile nature of streaming service audiences. In view of the drastic video traffic growth forecasts, the deployment and maintenance costs of content delivery solutions will rise, eventually leading to tremendous increases of streaming service pricing for content providers and consumers seeking high QoE content delivery.

In order to alleviate these QoE and cost issues, we propose to stop confining HAS to the single-server approach and propose a HAS-evolving streaming solution leveraging on the simultaneous usage of multiple servers in order to aggregate throughput on multiple communications channels with heterogeneous characteristics. Multiple-source HAS deployed in distributed environment has not yet been proposed in pragmatic-enough ways that would propel multiple-source streaming to the industry [Adhikari *et al.*, 2012c]. This chapter is dedicated to the presentation of **MS-Stream** *(Multiple-Source Streaming over HTTP)*, a pragmatic -i.e., easily deployable in practice- HAS solution that takes advantage of multiple-source environments in order to achieve similar, if not higher, QoE compared to existing uni-source HAS streaming solutions.

3.2 MS-Stream: Multiple-Source Streaming over HTTP

MS-Stream is an evolution of HAS solutions that relies and extends the Dynamic Adaptive Streaming over HTTP (DASH) standard. A manifest file listing the available servers is periodically being advertised to clients. For any given video segment

listed in the MPD file, the MS-Stream client simultaneously requests several servers via the HTTP protocol to compose in an online manner video segments (termed sub-segments). The composition sub-segments derive from the concept of Multiple Description Coding (MDC) [Kazemi *et al.*, 2013] that permits the creation of independently-decodable streams, which, when merged together can enhance the visual quality of the video. Sub-segment composition represents very low CPU footprint operations. When retrieved, the requested video sub-segments are merged to reconstruct and display the original requested content quality. In the event of sub-segment loss or late delivery, content playback continuity is not affected due to the received independently-decodable sub-segments, and only content quality is impaired. Additionally, if the considered servers or paths suffer from conditions degradation (abrupt throughput changes, high delay variations, etc.) quality adaptation and delivery adaptation mechanisms can prevent and avoid QoE degradation. Thanks to its codec agnosticism and DASH-compliance, this proposal represents an evolving solution that can be applied to many scenarios: P2P, CDNs, Clouds, Set-Top-Box overlays as well as collaborations of resource providers to achieve higher QoE and create new businesses. Figure 3.1 depicts the overall idea of a resource provider utilizing an MS-Stream-like solution for content delivery compared to currently used uni-source HAS solutions.

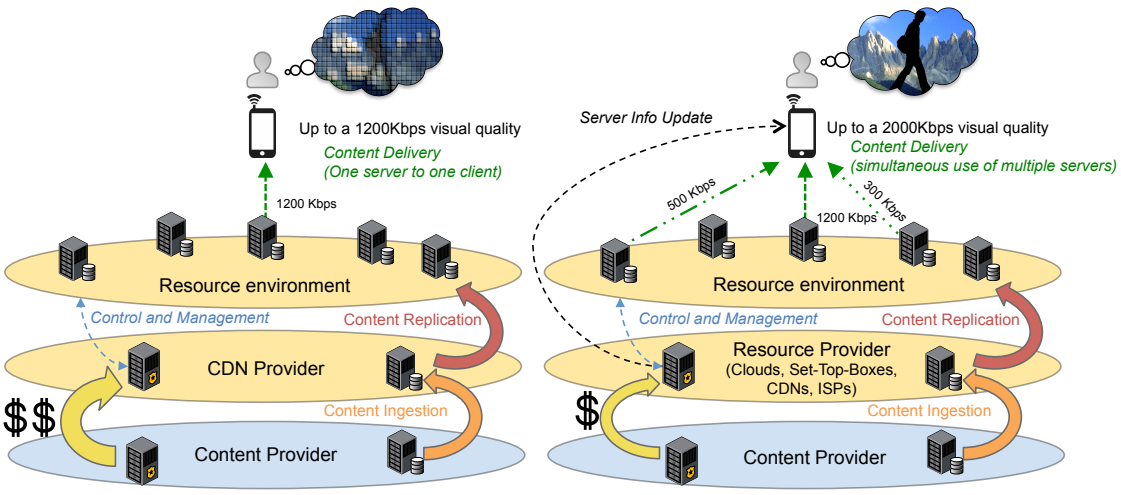

Figure 3.1 – Comparison between CDN-based HAS streaming solutions (on the left) and our MS-Stream solution (on the right)

3.2.1 Multi-source streaming system and architecture

We propose MS-Stream (Multiple-Source Streaming over HTTP) as a practical solution simultaneously exploiting multiple heterogeneous servers for consumers' QoE enhancement. The MS-Stream overall client/server architecture is depicted in Figure 3.2 where additional modules -compared to a classic HTTP Adaptive Streaming architecture- are highlighted in blue. Prior to the streaming session, a manifest file containing information about the available MS-Stream servers and the number of Group of Pictures per video segment (c.f. section 3.2.2) is delivered to the client.

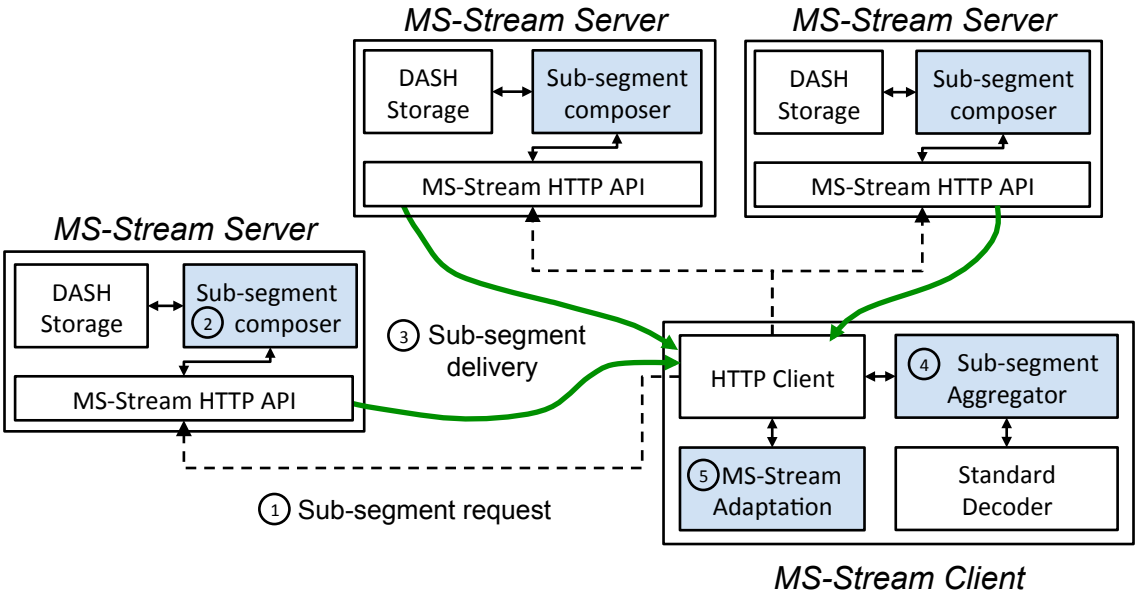

Figure 3.2 – MS-Stream client/server architecture

The MS-Stream content delivery steps consist of:

1. The MS-Stream client simultaneously instructs several MS-Stream servers to generate and deliver sub-segments through the *MS-Stream HTTP API*. The client assigns to each server the creation of a specific sub-segment by explicitely detailing the composition of the latter sub-segment in terms of Group of Pictures (GoPs) sequence and content quality for each GoPs;

2. The *Sub-segment composer* at server-side generates the requested sub-segment from the existing set of content segments made available at Q_{max} different bitrates in the *DASH Storage*. In order to fit any potential use-cases, the provisioning of the DASH storage is intentionally left undefined. For example, the

DASH storage could either be provisioned with content segments in an offline manner (static content) or be provisioned by an external video transcoding unit;

3. Sub-segments transit on the network while the MS-Stream client synchronizes their delivery and takes actions when late sub-segment delivery can negatively impact the delivered QoE;

4. The *Sub-segment aggregator* module merges the received sub-segments so as to reconstruct the original content quality and hands out the results to a *Standard video decoder*;

5. Finally, as content is being delivered over N paths (N sources), a global and per-path adaptation mechanism is required to deal with path heterogeneity, performed in the *MS-Stream Adaptation* module.

Hence, the MS-Stream client is an HAS-evolving client, which incorporates a cost-effective sub-segment aggregation module and an adaptation engine capable of content and server adaptation. Similarly, the MS-Stream server is also an HAS-evolving server that handles video sub-segment composition requests.

3.2.2 Sub-segments composition and aggregation

The sub-segment generation scheme utilized in our approach relies on the principles brought by Multiple Description Coding (MDC) [Kazemi *et al.*, 2013]. A video segment is generated over multiple sub-segments (referred to as descriptions in the MDC concept). Sub-segments are independently decodable and yield refinable information. Each standalone decoded sub-segment provides low, yet acceptable, quality of experience. Part of the original video quality can be reconstructed by aggregating sub-segments together; the full video quality is obtained when all sub-segments are aggregated into one single stream. Unlike layered coding, there is not any sub-segment ordering or dependency. Sub-segments have lower network bandwidth requirements than the original video segment, therefore they can transit more easily than in the case of single-server streaming.

This MDC-based splitting approach is the key lever to collect and aggregate bandwidth on multiple paths. Four major MDC domains have been studied in the literature [Kazemi *et al.*, 2013]: spatial, temporal, frequency, and compression. Following the work of Lu *et al.* [2007] and Tillo *et al.* [2010], and in order to benefit

from codec standard-compliance and from a large amount of possible sub-segments, we focus on a hybrid solution based on temporal and compressed data domains by interleaving the GoPs available at two different bitrate representations of the same segment (but with similar framerate and spatial resolution).

Sub-segments are generated by interleaving the GoPs available at two different bitrates of the same segment. As shown in Figure 3.3, sub-segments that are generated at the MS-Stream server end can be composed of GoPs at a high desired bitrate b_i, and some others at the critically low bitrate and redundant bitrate b_r. b_r is termed *redundant bitrate* because the GoPs at this bitrate contain data redundant with the other GoPs delivered in the other sub-segments. We assume that segments are composed of the same number G_{Tot} of standalone-playable GoPs; the GoP duration is the same for every bitrate b_i. We assume that for any given video segment at a given spatial resolution and encoded at a bitrate b_i ($b_i \in \{b_1, b_2, .., b_{Q_{max}}\}$ with $b_i < b_{i+1}, \forall i \in \{1, 2, .., Q_{max-1}\}$), one video segment at the same spatial resolution is made available in the DASH storage at critically low redundant bitrate b_r. The redundant bitrate b_r is set to such low values (e.g., 150 kbps for a 720p spatial resolution in H.264 baseline profile) in order to provide video playback at the lowest possible network transfer cost. Figure 3.5 depicts the example of three video sub-segments being composed of GoPs at a 720x480 spatial resolution with a 3Mbps bitrate shuffled with GoPs at the same resolution but with a much lower bitrate (150 kbps).

The video sub-segments composed by the MS-Stream server provide uncontinuous video perceived quality with GoP of varying bitrates. However, thanks to the redundant data (i.e., GoPs at the b_r bitrate) copied into sub-segments, the independence between sub-segments provides high reliability in heterogeneous and unreliable network environments. Indeed, complementary sub-segments can be lost without necessarily interrupting the streaming session.

Reconstructing the original content quality is achieved by selecting the GoPs of higher size in the pool of received sub-segments at client-side as depicted in Figure 3.4. Should some sub-segment be missing for content reconstruction, the content is still playable with sub-optimal visual quality, due to loss of information at the GoP level, inducing lower quality on a GoP-duration basis. Figure 3.6 and Figure 3.7 respectively expose the results of the sub-segment aggregation operation with sub-segments n°1 and n°3, and with all the sub-segments.

Targetting ease of adoption of this sub-segment composition/aggregation chain, our contribution benefits from the following three advantages: video-codec stan-

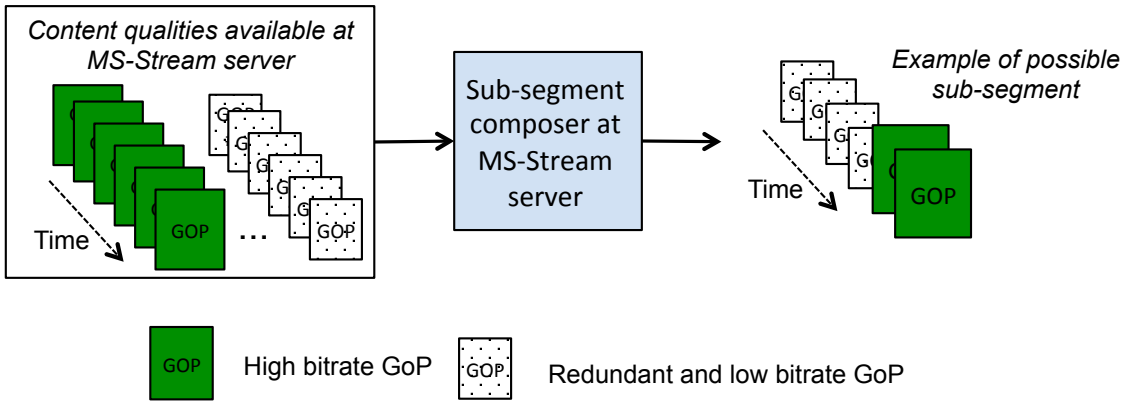

Figure 3.3 – Sub-segment generation scheme

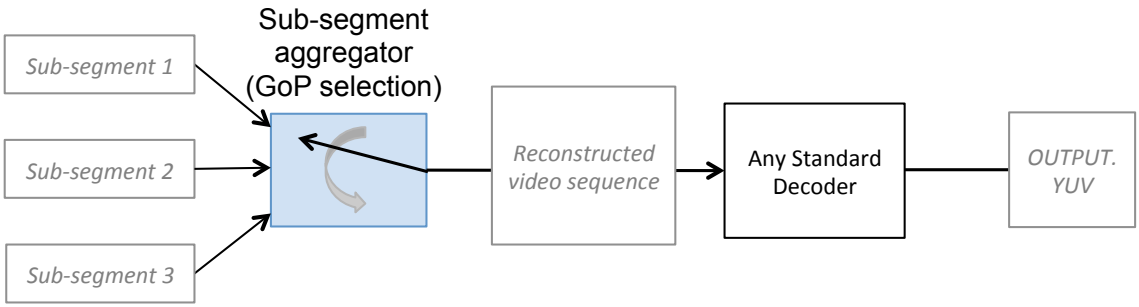

Figure 3.4 – Sub-segment aggregation

dard compatibility, low additional complexity and the possibility to create as many different sub-segments as needed.

1. *Video-codec standard compatibility:* Standard compatibility allows receivers to use their standard decoder modules to decode sub-segments without bringing any changes to their implementation. As our sub-segment composition/aggregation chain does not alter the video bitstream structure but only necessitates to assemble already encoded GoPs available at different bitrates, and since bringing new encoding concepts to standardization organization is a very tedious and time-consuming enterprise, this video-codec standard compliance is of main interest for a quick adoption of this proposal. A simple pre-decoding step is required to aggregate sub-segments, which can be achieved inside the video player itself.

2. *Low complexity:* Due to the rapid growth of battery-powered devices, less computational complexity is still of high interest. In this work, complexity is

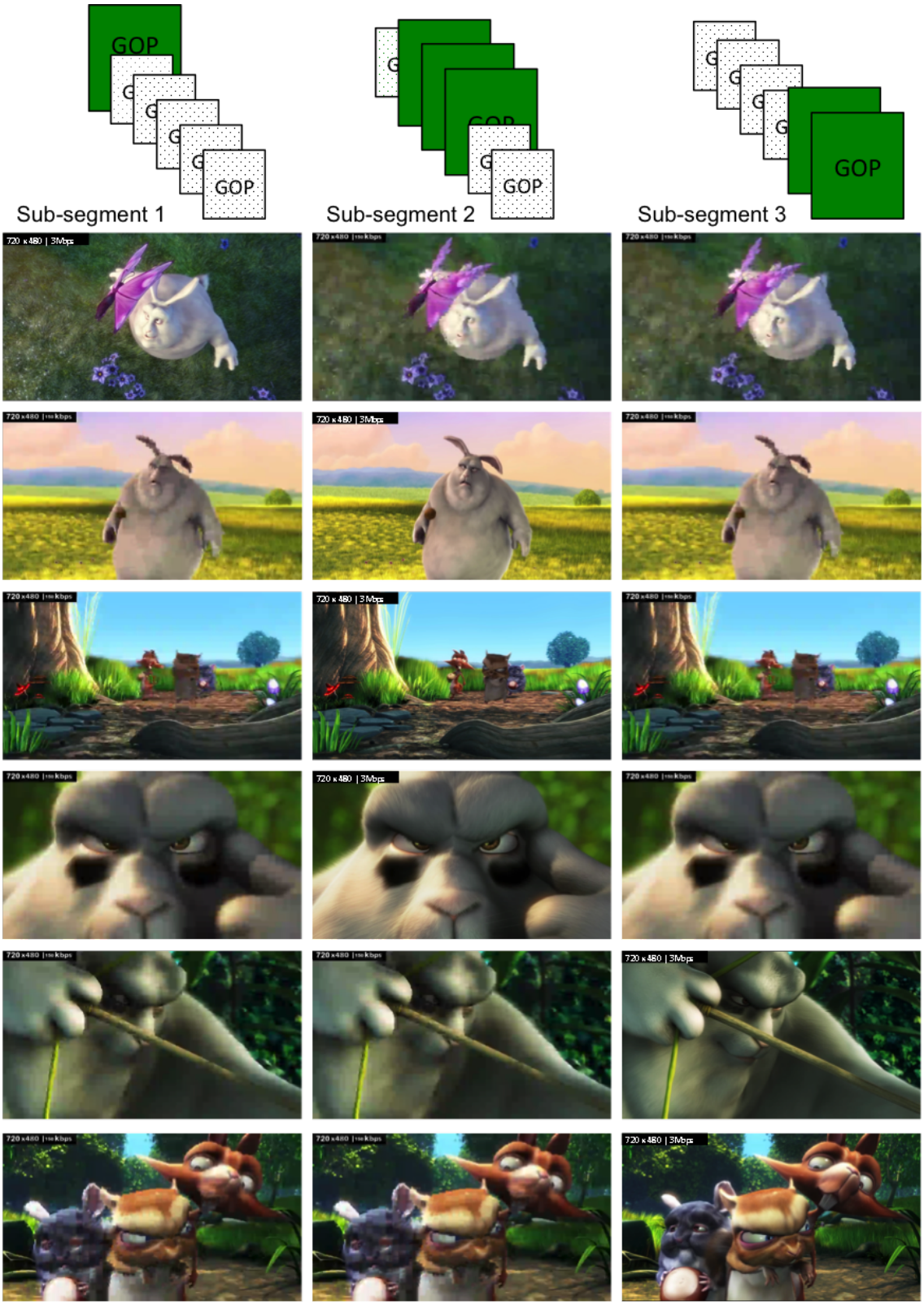

Figure 3.5 – Example of three composed sub-segments exposing sequences of Group of Pictures at different bitrates with the same resolution

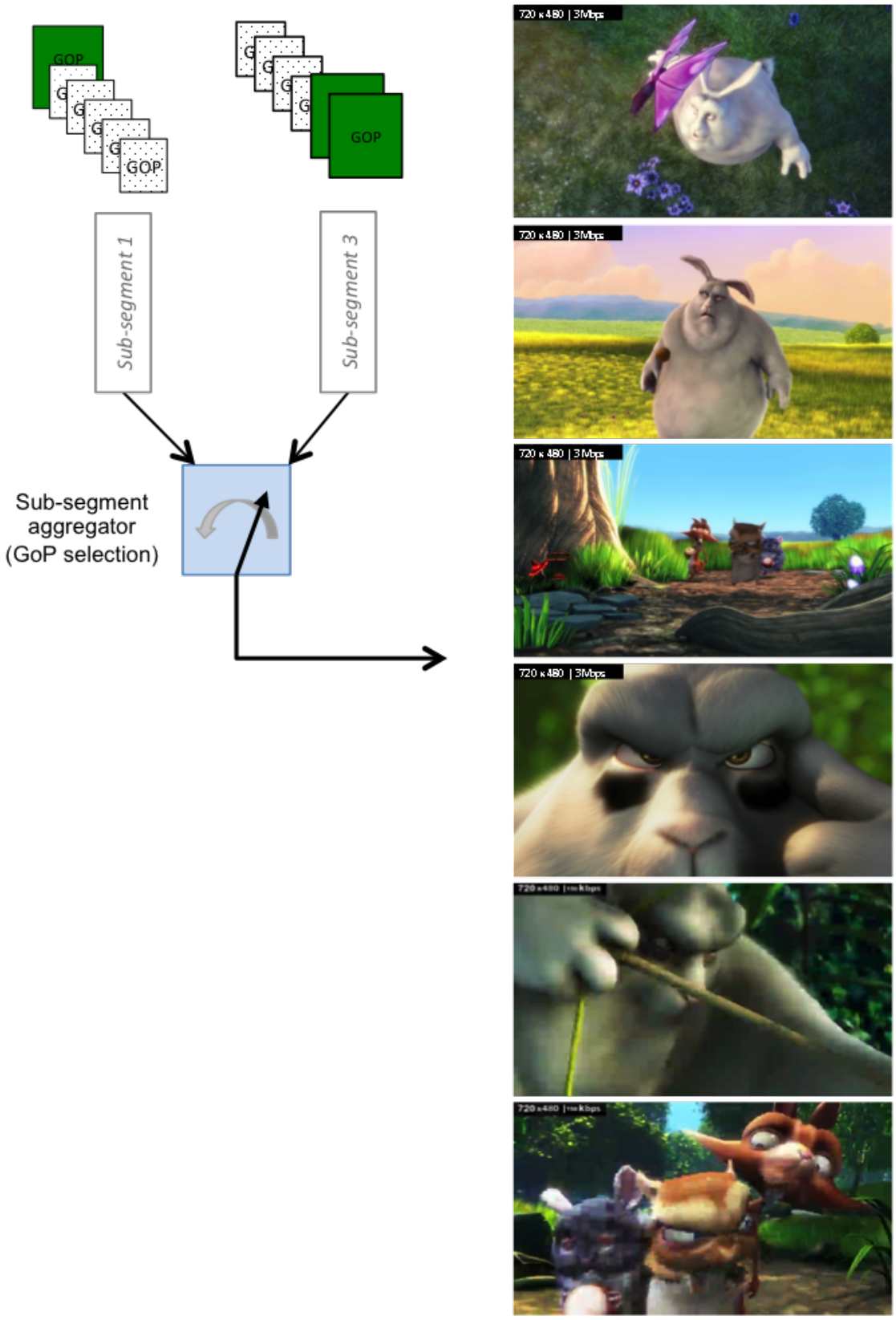

Figure 3.6 – Aggregation of sub-segments number 1 and 3

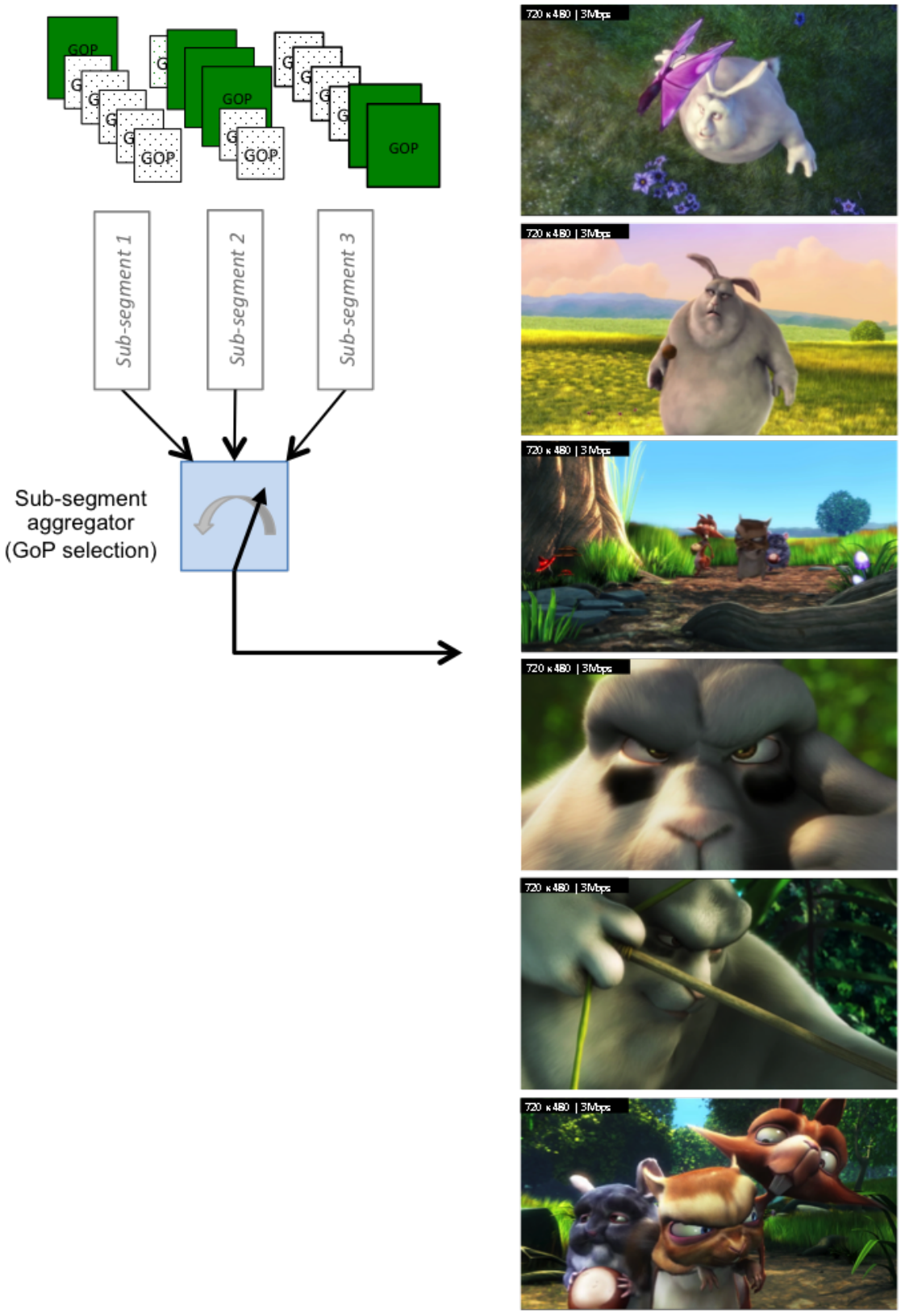

Figure 3.7 – Aggregation of sub-segments number 1, 2 and 3

referred to as the computational overhead compared to the standard encoding and decoding operations. Therefore, it is of major importance to minimize such overhead. In our approach, the composition and aggregation of sub-segments represent movement of small amount of data from one place of the MS-Stream client's process address space to another. As a result, the additional complexity at client's side is insignificant compared to the actual decoding operations often requiring hardware acceleration. Regarding the server-side computational overhead the same applies. However, streaming servers are expected to operate high workloads. The impact of sub-segment creation on the server's ability to handle as many requests as possible is studied in our evaluation (see section 3.3).

3. *Capability to increase the number of possible sub-segments per video segment:* The limited number of possible sub-segments is one of the main drawbacks of most of the proposed multiple-description coding approaches for video strreaming, as exposed in the survey conducted by Kazemi *et al.* [2013]. Generating two sub-segments -or descriptions- is generally suitable for most cases that do not necessarily relate to video streaming over the Internet (such as lossy wireless channels or audio communications) and do not require the usage of 4 or more descriptions. However, in streaming use-cases that leverages on using multiple and separate paths (such as P2P or multiple-source video streaming), high number of sub-segments permit to improve network resource usage flexibility, to increase total aggregated bandwidth, to reduce probability of outage, and to compensate delay variability [Apostolopoulos and Trott, 2004]. In our approach, the number of possible different sub-segments is only limited by the number of GoPs per video segments. Therefore, increasing the number of GoPs per segment will result in increasing the system flexibility, i.e., using multiple sources.

3.2.3 DASH standard compliance

From the standpoint of compliance with streaming standards, the MS-Stream solution relies on the MPEG-DASH standard [ISO/IEC MPEG, 2014] in defining the MPD file that is handed out to the clients. However, the MS-Stream solution goes a step further than MPEG-DASH as it necessitates to incorporate one additional information to the MPD file not initially described in the standard: the number of GoPs per video segment.

From a technical standpoint, MS-Stream ensures DASH-backward compliance: upon the delivery of a regular DASH manifest file, an MS-Stream client can use uni-source DASH protocol. MS-Stream could become DASH-forward compliant quite easily by extending the MPD specification and adding the information of the number of GoPs into it.

3.2.4 Panel of adaptation possibilities

The MS-Stream solution advocates for client-centric decisions to improve the QoE of end-users. We present a spectrum of client-centric QoE-aware adaptation possibilities based on the simultaneous utilization of multiple servers.

3.2.4.1 Sub-segment scheduling

Increasing end-user's QoE by benefiting from the diversity of sources and bandwidth offered by multiple-source HTTP video streaming can be a challenging issue in environments where resource heterogeneity can rapidly become significantly wide. Indeed, when the available throughput capacities of the considered servers are heterogeneous and fluctuate in time, the MS-Stream client can face late sub-segment delivery that does not necessarily impact the playback continuity but impair the displayed video quality.

Balancing the load that the MS-Stream client requests on each server/network path is one way to circumvent this problem. Based on the above-described sub-segment composition/aggregation chain, we present a simple and efficient method enabling MS-Stream client to utilize multiple network paths and conteract the heterogeneity of available bandwidth that characterizes multi-source environments. By dynamically adjusting the number of GoPs at a target bitrate b_i requested from each server, the MS-Stream client can adapt its utilization of each communication channel in accordance with previous observations of throughput available on the latter communications channels.

Prior to the streaming session, the MS-Stream client is handed out a manifest file containing information on the available MS-Stream servers as well as the number of GoPs per video segment. Based on this, the MS-Stream client specifies the GoP composition of each video sub-segment that each server should compose. More explicitly, the MS-Stream client requests, as parameters in the HTTP requests, each server to generate a video sub-segment with some GoPs at a selected target b_i bitrate and the remaining ones at the critically low and redundant b_r bitrate.

Let us consider a video segment with G_{Tot} GoPs that the MS-Stream client wishes to display at bitrate b_i while simultaneously using a set S_M of M servers. Let us define G_s the number of GoPs at bitrate b_i assigned to server s. Let us define $GoP_{j,b_i,s}$ so that $GoP_{j,b_i,s} = 1$ when server s is assigned the delivery of GoP j at bitrate b_i and $GoP_{j,b_i,s} = 0$ otherwise. Therefore, we have:

$$G_s = \sum_{j=1}^{G_{Tot}} GoP_{j,b_i,s} \leq G_{Tot} \tag{3.1}$$

The MS-Stream client requests each server to deliver a playable video sub-segment so that equations 3.1 and 3.2 are satisfied, under the constraint that the delivery of GoP j at the targeted bitrate b_i is assigned to one server only. Thus, servers are being assigned the generation and delivery of sub-segments with different GoPs at the selected high bitrate. Eventually, the MS-Stream client can reconstruct the original segment at the requested bitrate.

$$\sum_{s=1}^{M} G_s = \sum_{s=1}^{M} \sum_{j=1}^{G_{Tot}} GoP_{j,b_i,s} = G_{Tot} \tag{3.2}$$

A simple way of deciding how many GoPs should be assigned to each server is by minimizing the impact of sub-segment late delivery or loss on the perceived QoE by the end-users. Indeed, each sub-segment arriving later than a given deadline will impair the displayed video segment's visual quality by a duration proportional to the number of GoP at the targeted bitrate carried in the sub-segment. Consequently, minimizing the impact of each sub-segment late delivery (i.e., minimzing the number of GoPs at bitrate b_i assigned to each server) is an important issue to maintain high end-users' QoE. In order to address this issue, the MS-Stream client is required to find a solution $\mathscr{S} = \{G_s\}_{s\in[1..M]}$ satisfying equation 3.3, under the constraint of equation 3.4 that the network throughput required to deliver each sub-segment in real-time manner from the assigned server/path does not exceed the throughput previsouly observed -or estimated- from the latter server/path.

$$\mathscr{S} = argmin_{\{G_s\}s\in[1..M]} \left\{ \sum_{s=1}^{M} \left(G_s - \frac{G_{Tot}}{M} \right) \right\} \tag{3.3}$$

$$\frac{G_s}{G_{Tot}} \cdot b_i + \frac{G_{Tot} - G_s}{G_{Tot}} \cdot b_r \leq \textit{Throughput from server } s \; \forall s \in [1..M] \tag{3.4}$$

3.2.4.2 Sub-segment requests synchronization

In our approch, the MS-Stream client waits until it receives all sub-segments before merging and displaying them. By simultaneously retrieving sub-segments from several servers, the probability to receive at least one stream is increased. The MS-Stream client monitors the download progress of sub-segments and takes decisions to cancel the downloads so as to be resilient to network heterogeneity and avoid video blocking events. This mechanism is called sub-segment requests synchronization. A set of three rules is defined for the sub-segment delivery synchronization at client side. For a given segment:

1. If at least one sub-segment is retrieved, then other sub-segment downloads can be abandoned; this reactive rule ensures the delivery of at least one sub-segment before moving on to the next one;

2. If the buffered content playout reaches a given lower threshold t_{thresh}, sub-segment downloads can be canceled by the MS-Stream client. In doing so, uninterrupted video streaming experience is ensured to the end-users, temporarily providing a sub-optimal visual quality to the end-users. This second reactive rule can only be applied if rule (i) is satisfied;

3. If the content playout duration available in the client's buffer exceeds twice the average video segment duration -given as input of the streaming session in the manifest file delivered to the client-, then a timeout value is set on HTTP description segment requests. The timeout value reflects a consumption behavior (aggressive, conservative, etc.) and can be tuned during the streaming session, according to the available buffered content. Once the timeout has elapsed, sub-segment requests can be canceled while satisfying rules (i) and (ii). This proactive rule enables the usage of the buffer to compensate for network characteristic fluctuations on the different paths used. In this chapter, the threshold value of twice the average segment duration is arbitrarily choosen in order to represent the case when the MS-Stream attemps to maintain a buffer occupancy level high when the deliveries of some sub-segments are delayed.

3.2.4.3 Content adaptation and bandwidth overhead consumption

As in DASH solutions and in most HAS, our proposal is agnostic to the implemented bitrate adaptation methods/algorithms, which makes it highly flexible and adapted

to existing or future approaches for video streaming over the Internet. The main objective of this chapter is not on optimal bitrate adaptation methods, but on explaining that bitrate adaptation is feasible in this client-centric approach for the simultaneous usage of multiple servers with heterogeneous capacity. Given a video segment, as soon as all sub-segments are delivered, the client estimates the status of each used path (by measuring the observed throughputs), and computes the global available throughput. Then, a candidate bitrate b_i is selected for the download of the next segment.

Although MS-Stream does not emphasize any quality adaptation mechanisms, one additional information should be taken into account in the design of quality adaptation methods: the bandwidth consumption overhead resulting from the redundant GoPs transmitted on the networks. Indeed, the delivered redundant GoPs may not necessarily take part in the displayed video at the client side. We have:

$$\text{PlayedData} + \text{NonPlayedData} = \text{TransmittedData} \tag{3.5}$$

Therefore, the bandwidth consumption overhead -equation 3.6- is defined as the percentage of data transiting on the network, which does not take part in the video displayed to the end-user:

$$Overhead = 1 - \frac{\text{PlayedData}}{\text{TransmittedData}} \tag{3.6}$$

However, prior to issueing sub-segment requests to the servers, the MS-Stream client is unable to determine which sub-segments will arrive late and which ones will arrive in time, and assumes that all GoPs at the redundant bitrate b_r could potentially be useful to prevent video playback disruptions. Therefore, not only the MS-Stream client ends up over-utilizing the available throughput on each network path to provide smooth streaming, but it should also take into account that the required network throughput to obtain multiple video sub-segment containing all the GoPs sufficient to display a video segment at bitrate b_i is greater than b_i. As the client knows in advance the number of GoPs n_{extra} at the redundant bitrate b_r that will be requested, it can determine the extra throughput δ required to deliver the set of sub-segments compared to delivering one video segment at bitrate b_i in real-time, i.e., with a network throughput of b_i. Therefore, we have:

$$n_{extra} = \Big(M - 1\Big) \cdot G_{Tot} \tag{3.7}$$

and,

$$\delta = \frac{n_{extra} \cdot b_r}{G_{Tot}} = \Big(M - 1\Big) \cdot b_r \tag{3.8}$$

The difference between equations 3.8 and 3.6 is that equation 3.8 shows the extra throughput requested by the client on the network while equation 3.6 exposes the data that transitted on the network and that was not displayed. Therefore, the value of $Overhead$ and δ differs when sub-segment requests are abandonned. Thus, $\delta \geq Overhead$.

The utilization of extra bandwidth represents both a disadvantage and an advantage. On the one hand, the MS-Stream client necessitates an extra δ Mbps throughput in order to display a video segment at bitrate b_i. On the other hand, the flexibility brought by this approach enables the client to use multiple servers with heterogeneous characteristics (available throughput, processing power) while improving QoE by providing increased throughput, resiliency, and uninterrupted video playback in the event of suddenly low performing servers or abrupt throughput drops. From equation 3.8, the greater the number of streaming sources used, the greater the extra bandwidth required. This exposes the limitation of this multiple-source proposal in scaling up the number of used servers. Chapter 4 presents a solution to circumvent this problem.

3.2.4.4 Server adaptation for flexible usage of network resources

As above-mentioned, a manifest file containing information on the available servers (including their IP addresses or domain names) is delivered to the client. This manifest can be updated periodically to let resource providers adapt the pool of available servers. Since the retrieved sub-segments are independent from each other, the MS-Stream protocol also offers clients the opportunities to seamlessly handover the delivery of sub-segments to other servers, or to change the number of considered servers during streaming sessions (by adding, removing servers).

Several types of server adaptation policies can be implemented with regards to observations on the network paths and servers' characteristics (available throughput, delay, outages, etc.) and to QoE objectives (a target video bitrate with limited amount of rebuffering events). For example, priority could be given to visual content quality stability by switching to or adding more stable servers to the session when the considered ones are very unstable and are assigned an important number of GoPs. MS-Stream also supports the use of a large range of different content source types such as sets of set-top-boxes and cloud servers. Finally, when the MS-Stream

client switches to the use of one server, a regular DASH session takes places.

3.3 In-Lab evaluations

We evaluate the above presented MS-Stream solution on several QoE influence factors. The MS-Stream server is implemented in Java8 and the MS-Stream client is implementd on top of the DASH dash.js [DASH-IF, 2017c] video player from the DASH-Industry Forum [DASH-IF, 2017a]. We compare our solution to the DASH reference software player and to the optimum solutions for multiple-source streaming. Throughout our experiments, we keep the same quality adaptation algorithm when comparing streaming solutions. All implemented and simulated streaming players include the same adaptation rules derived from the DASH-IF dash.js player, except for the MS-Stream-based clients that have to take into account the additional bandwidth consumption overhead in the decision to upgrade or downgrade the requested video quality. First, we describe the in-lab evaluation setup as well as the evaluation scenarios before presenting our results on the QoE gain of the MS-Stream approach along with its bandwidth consumption and processing overhead.

3.3.1 Evaluation testbed and streaming applications

3.3.1.1 Testbed

Our in-lab setup is composed of 9 servers connected to a router as depicted in Figure 3.8. Each server has a traffic shaper module on its network interface card to enable tuning the available throughput, delay and packet loss for each network path between the client and the available servers.

Each server is provisioned with the 10-minute Big Buck Bunny movie [Blender, 2017] encoded at 7 different bitrates with a 720p spatial resolution and at 30fps, in H.264 main profile (200kbps, 1Mbps, 1.5Mbps, 2Mbps, 3Mbps, 4Mbps and 6Mbps) and segmented in 6-second segments, each containing 12 GoPs of half a second.

In this evaluation, we consider the perceived quality at the end-user side, i.e., QoE. To this end, we have derived a set of criteria representing the QoE influence factors (each considered essential for video streaming services' QoE): mean displayed bitrate, average number of rebuffering, average start-up delay, average number of quality changes, and quality distribution throughout the streaming session for each client.

The setup also includes a monitoring server that the client periodically contacts in order to report streaming session related metrics. Based on the latter metrics, the monitoring server computes the above-defined QoE influence factors.

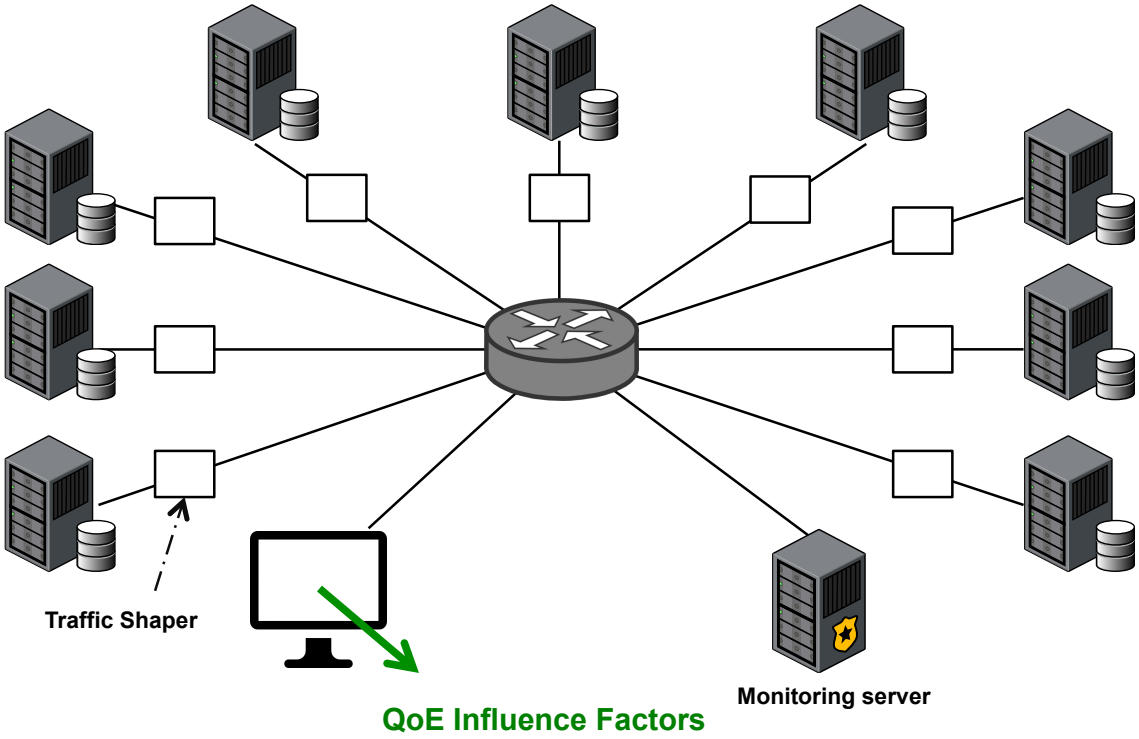

Figure 3.8 – Test-bed setup

3.3.1.2 Evaluated streaming applications

We evaluate three different versions of the MS-Stream client by varying the degree of adaptability each application comprises:

1. MS-Stream-NA (Non-Adaptive): In this MS-Stream Non-Adaptive application, the client uses three servers simultaneously but cannot adjust the requested video quality and cannot handover the delivery of sub-segments to other servers. Additionally, MS-Stream-NA client cannot dynamically adjust the number of GoPs that are assigned to each server, and is constrained to always request each servers for sub-segments composed of 4 GoPs out of 12 at the 6 Mbps bitrate, and the remaining 8 GoPs at the 200 kbps bitrate.

2. MS-Stream-BA: The MS-Stream-BA (Bitrate Adaptation) client uses 3 servers simultaneously and adapts the desired video bitrate by enabling the quality

adaptation algorithm of the dash.js player. The MS-Stream-BA client also adjusts the number of GoPs per video sub-segment based on the method described in section 3.2.4.1.

3. MS-Stream-BSA: the MS-Stream-BSA (Bitrate and Server Adaptation) extends the MS-Stream-BA client by using three servers simultaneously and by having the possibility to handover the delivery of sub-segments to other servers from a list of 9. The implemented server adaptation strategy is simple and basic but demonstrates the advantage of bringing flexibility in the usage of the considered streaming sources. We define the critical condition, where one of the three considered servers is assigned 10 GoPs out of 12 at the target bitrate. This critical condition lets the client identify when the two lightly loaded servers have low throughput capacities and may not be performing sufficiently well to deliver at least one GoPs in a real time manner. When this condition is reached, the client removes the less-performing server (in terms of delivered throughput observations) and replaces it by another server randomly selected from the list of unused streaming sources.

Three other streaming applications were implemented in order to simulate the full potential of both DASH and MS-Stream in multi-source environments. Referred to as *oracle* clients, they know in advance all upcoming throughput variations, delays and packet losses. In our study, oracles are used as reference use cases to allow QoE comparison. The adaptation decisions of the oracle clients rely on actual true predictions of bandwidth availability on each client-server path. Consequently, the oracle clients always take the most appropriate decisions regarding quality adaptation and server adaptation, maximizing QoE by retrieving the highest possible content bitrate while minimizing video freezing events, hence foreseeing the optimum of each solutions.

4. Uni-Source-DASH-oracle: The dash.js player using only one server simultaneously but having the possibility to switch from the currently used server to the best one (in terms of available bandwidth) before the download of each new video segment. Additionaly, this client also incorporates the bitrate adaptation method initially implemented in the dash.js player with modifications brought to the estimation of the available bandwidth as the client knows in advance the bandwidth available on all paths.

5. Multi-Source-DASH-oracle: Relying on the simultaneous usage of 3 servers, this oracle client downloads the first next segment to be played out from the

first server offering the highest throughput. Concurrently, the second next segment to be played out is retrieved from the second server offering the highest throughput, and the same applies for the third next segment.

6. MS-Stream-oracle: The MS-Stream oracle behaves the same as MS-Stream-BSA, with the exceptions of selecting the video quality to be requested for the next video segment based on previous knowledge of the network conditions, and switching to the most available servers (in terms of available throughput) before the download of each new video segment.

Table 3.1 sumarizes the evaluated applications, their consumption and adaptation capabilities.

3.3.1.3 Experiments

The DASH Industry Forum provides benchmarks for various aspects of the DASH standard [DASH-IF, 2014] including six *Network Profiles* (NPs) featuring different bandwidths, delays and packet losses. Each profile spends 30 seconds for each step described in Table 3.2, and then starts back at the beginning. For example, a 10-minute streaming session under the conditions of NP #1 would loop two and a half times on the set of steps composing it.

In each experiment, the outgoing trafic from the servers is shaped according to one pre-selected NP from Table 3.2. The trafic limitation is applied through the linux traffic shaper tool (*tc*) on the server's network interface card. A random time offset is set for the beginning of each assigned NP in order to represent bandwidth diversity and variability in the network. As a result, each server starts the experiment with different delays, throughputs and packet losses. An external module located in the monitoring server controls the transitions between each step of the selected NP through an API controlling the network traffic shaper modules at server side. We purposely left an unlimited throughput capacity on the link to the client. For each experiment, a client consumption style and adaptation capabilities detailed in Table 3.1 is selected. The MS-Stream based clients use the 200 kbps video bitrate quality as the redundant bitrate b_r to be copied into sub-segments.

To sum up, each experiment features: (1) one selected network profile applied at the server network interface card with different start time; and (2) a streaming solution selected from Table 3.1. Each experimental configuration was played 50 times, representing a cumulated playback time of 300 hours.

Table 3.1 – Test-beds and streaming applications

Streaming Applications	**Avai--lable Servers**	**# of used servers**	**Consumption styles and adaptation capabilities**
Uni-Source DASH-oracle	3	1	Bitrate + Server Adaptation *Always downloading from the best path based on prior knowledge of throughput fluctuations*
MS-Stream-NA	3	3	Absence of bitrate and server adaptation Fixed GoP distribution policy (4 GoPs out of 12 at 6Mbs –b_i- and 8 GoPs out of 12 at 200kbps -b_r-) from each server)
MS-Stream-BA	3	3	Flexible GoP distribution policy + Bitrate adaptation *based on throughput estimation*
MS-Stream-BSA	9	3	Flexible GoP distribution policy + Bitrate + Server adaptation *Switching from the less-performing server to another randomly selected server every new segment download only if the GoP distribution step assigns a given server with 10 or more GoPs out of 12, based on throughput estimation*
MS-Stream -oracle	9	3	Flexible GoP distribution policy + Bitrate + Server adaptation *based on prior knowledge of throughput fluctuations*
Multi-Source -DASH-oracle	9	3	Bitrate + Server adaptation *Downloading the first segment on the best path and the two next segments on the two next best paths based on prior knowledge of throughput fluctuations*

Table 3.2 – Network Profiles representing throughputs, delays and packet loss

#1 Mbps (ms,%)	**#2** Mbps (ms,%)	**#3** Mbps (ms,%)	**#4** Mbps (ms,%)	**#5** Mbps (ms,%)	**#6** Mbps (ms,%)
5.0 (38;0.09)	5.0 (13;0.81)	5.0 (11;1.00)	-	-	-
4.0 (50;0.08)	4.0 (18;0.63)	4.0 (13;1.25)	9.0 (25;0.06)	9.0 (10;0.40)	9.0 (06;1.00)
3.0 (75;0.06)	3.0 (28;0.44)	3.0 (15;1.50)	4.0 (50;0.07)	4.0 (50;0.08)	4.0 (13;1.25)
2.0 (88;0.09)	2.0 (58;0.21)	2.0 (20;1.75)	2.0 (75;0.10)	2.0 (150;0.03)	2.0 (20;1.50)
1.5 (100;0.12)	1.5 (200;0.03)	1.5 (25;2.00)	1.0 (100;0.16)	1.0 (200;0.07)	1.0 (25;2.00)
2.0 (88;0.09)	2.0 (58;0.21)	2.0 (20;1.75)	2.0 (75;0.10)	2.0 (150;0.03)	2.0 (20;1.50)
3.0 (75;0.06)	3.0 (28;0.44)	3.0 (15;1.50)	4.0 (50;0.07)	4.0 (50;0.08)	4.0 (13;1.25)
4.0 (50;0.08)	4.0 (18;0.63)	4.0 (13;1.25)	-	-	-

3.3.2 QoE Evaluation

3.3.2.1 Rebuffering per session and average start-up delay

The average number of rebuffering events taking place during a streaming session at each network profile is exposed in Figure 3.9. For all applications except MS-Stream-NA, rebuffering almost never takes place (< 0.1 freezing events per streaming session). As to the MS-Stream-NA client, rebuffering occurs between 0.12 and 1.22 times on NPs #1-#2-#3 and between 2.76 and 4.08 times on NPs #4-#5-#6.
Regarding the video start-up delay, the MS-Stream-BA and MS-Stream-BSA solutions play out the first video segment as soon as they retrieve a sub-segment (and immediately cancel the other description downloads), and have the lowest average start-up delay (1.10 seconds on NPs #1-#2-#3 and 0.87 seconds on NPs #4-#5-#6). This short start-up delay comes at the cost of temporary degraded quality for the first video segment. Uni-Source-DASH-oracle and Multi-Source-DASH-oracle have longer delays (respectively 2.03 seconds and 1.45 seconds on NPs #1-#2-#3 and 1.81 seconds and 0.93 seconds on NPs #4-#5-#6) compared to the MS-Stream clients.

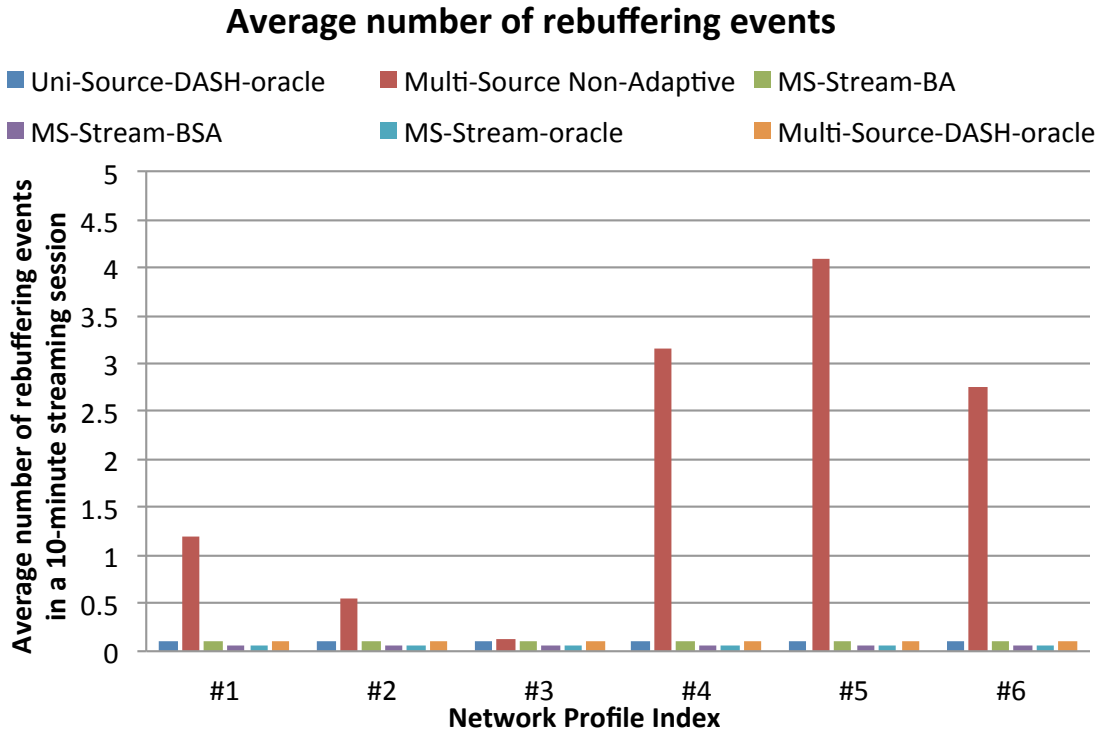

Figure 3.9 – Number of rebuffering events

3.3.2.2 Mean bitrate and number of quality changes

Figure 3.10 and Figure 3.11 respectively represent the average mean bitrate obtained on each network profile and the average number of quality changes that occur during the 10-min streaming session. For all implemented applications, the mean bitrate results on NPs #1-#2-#3 are very similar. The differ of at most 7.6%. on each NP. The same observation is made for NPs #4-#5-#6 with a maximum mean bitrate difference of 6%.

Uni-Source-DASH-oracle reaches a mean bitrate of approximately 3385kbps on NPs #1-#2-#3 and 4416kbps on NPs #4-#5-#6. In comparison with Uni-Source-DASH-oracle, the simultaneous usage of multiple servers allowed MS-Stream-NA to achieve higher video bitrates on NPs #1-#2-#3 with an average 43.3% increase, and with an average increase of 6.1% on NPs #4-#5-#6. Because of the many quality changes that take place during the streaming sessions with MS-Stream-NA (between 27 and 32 changes on NPs #1-#2-#3, 39 and 52 on NPs #4-#5-#6), the apparently high bitrates and low number of rebufferings for MS-Stream-NA are far from being representative of the perceived quality by the end-users.

The basic sub-segment scheduling mechanism implemented in the MS-Stream-BA client enables a better usage of network resources. This results in an increase of the obtained mean bitrate to approximately 5733kbps on NPs #1-#2-#3 and 5540kbps on NPs #4-#5-#6. Opposite the MS-Stream-NA, the resulting QoE perceived by the end-users relying on MS-Stream-BA is significantly higher as the associated number of quality changes throughout each streaming session does not exceed 5 on all NPs. This result emerges from the utilization of rate adaptation to cope with the fluctuations of bandwidth availability on the considered paths.
Finally, by leveraging on server adaptation (switching), the MS-Stream-BSA application increases the mean bitrate up to 5900kbps on NPs #1-#2-#3 and 5845kbps on NPs #4-#5-#6, respectively representing 74% and 32% gains compared with Uni-Source-DASH-oracle. Compared to MS-Stream-oracle, Multi-Source-DASH-oracle and MS-Stream-BA, the average number of quality changes are further lowered to reach no more than 3.4 changes per 10 minutes.
Regarding the multiple-source clients (Multi-Source-DASH and MS-Stream-oracle), they display slightly better performances in terms of mean bitrate and quality changes than MS-Stream-BSA. The latter leads to the conclusion that in a network environment presenting controlled resources with heterogeneous capacities and with one client, multiple-source streaming with simple rate and server adaptation techniques can achieve near optimal performances in terms of QoE influence factors.

3.3.2.3 Content quality distribution

Figure 3.12 plots the average quality distribution obtained by the client on each NP throughout the streaming session. Because the MS-Stream-NA client has to frequently abandon sub-segment downloads in order to prevent the video playback interruptions due to network resource heterogeneity, it displays the top content quality between 88% and 72% of the time on all NPs, and the 200kbps quality for the rest of the time with a visual flickering effects between the 200kbps and the top quality.

The benefits of MS-Stream-BA on QoE are clearly visible in Figure 3.12: the top quality is displayed for a greater percentage of the time compared to MS-Stream-NA. In addition, MS-Stream-BA only relies on displaying the 200kbps quality so as to prevent rebuffering for less than 1% of the time on each NP, whereas MS-Stream-NA has to use the redundant bitrate quality for at least 12% of streaming session's duration. The remaining 12% to 23.5% of the MS-Stream-BA streaming session are left for the play out of the 4Mbps quality.

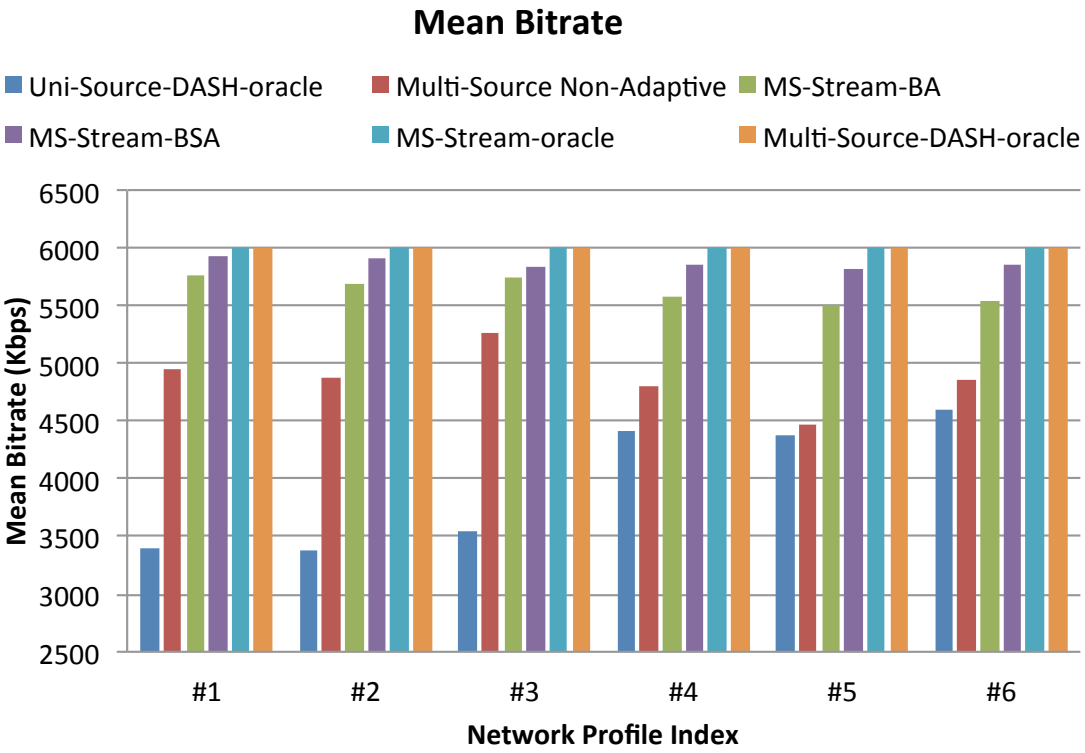

Figure 3.10 – Mean bitrate per session

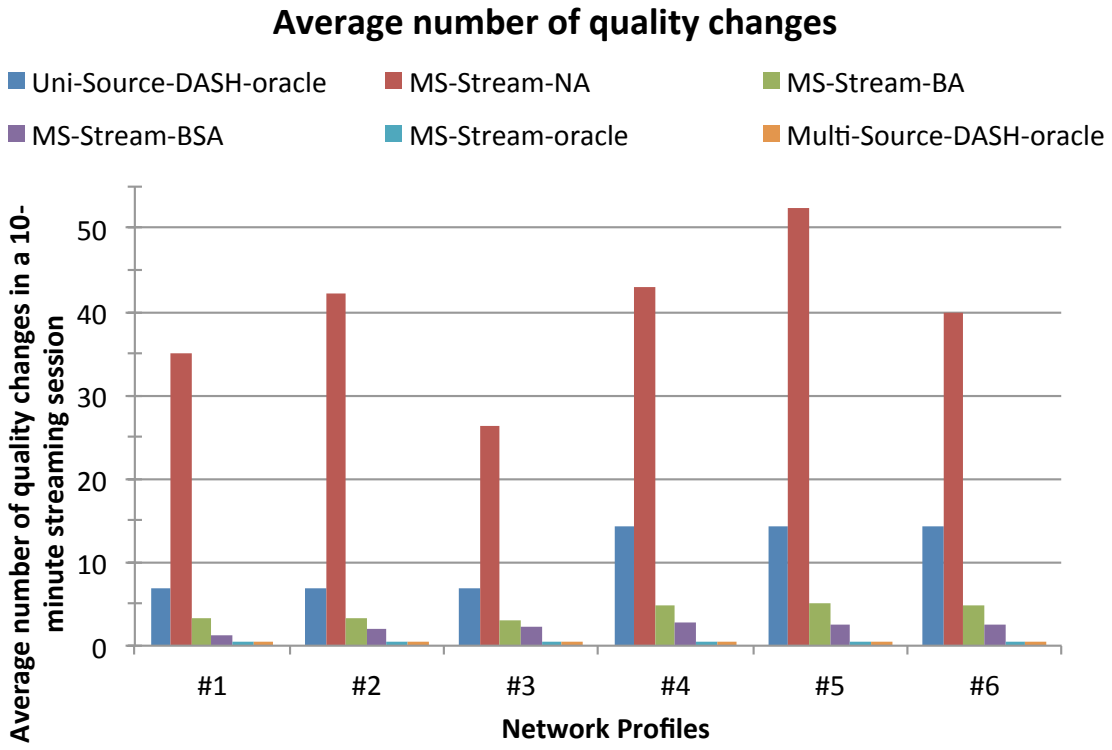

Figure 3.11 – Average number of quality changes

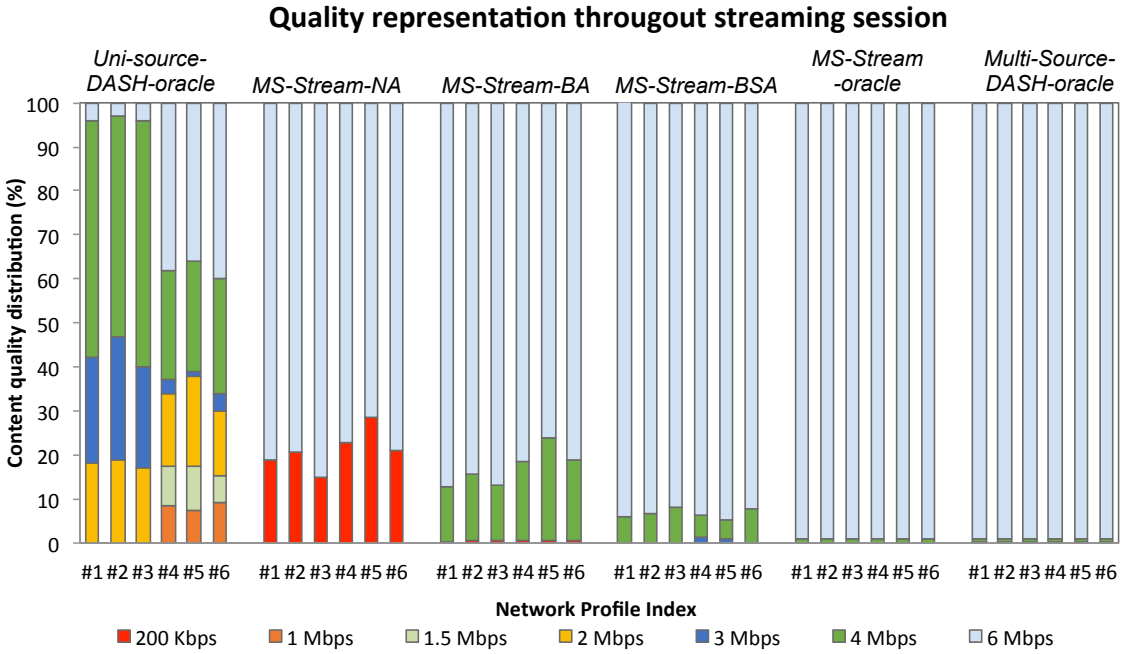

Figure 3.12 – Quality distribution throughout the streaming sessions

Relying on server adaptation, MS-Stream-BSA takes benefit of greater link diversity, which leads to provide the 6Mbps quality for 93% of the time on average on all NPs. For the Uni-Source-DASH-oracle client, although 3 servers are made available and Uni-Source-DASH-oracle had prior knowledge of upcoming throughput fluctuations, the use of one server only was not enough to provide even a similar quality distribution compared to MS-Stream-BA or MS-Stream-BSA. As to MS-Stream-oracle and Multi-Source-DASH-oracle, they performed similarly with 99% of the time at 6Mbps during streaming sessions.

3.3.3 Overhead Evaluations

In this sub-section, the bandwidth consumption overhead (defined in section 3.2.4.3) of the MS-Stream solution is evaluated and quantified. We also investigate the pre-processing required at the server side to compose sub-segments compared to regular static HTTP requests.

3.3.3.1 Bandwidth Consumption Overhead Evaluation

As depicted in Figure 3.13, every multiple-description-based streaming application presented an overall bandwidth consumption overhead representing less than 7% of

the transmitted data on all NPs. Interestingly, MS-Stream-NA had the least amount of overhead (6.07%). This result comes down to sub-segment downloads often being too late -thus frequently cancelled- due to the lack of adaptability of MS-Stream-NA to face heterogeneous network conditions. Hence, the MS-Stream-NA client is often required to display the redundant GoPs at the redundant bitrate so as to give a smooth video experience to the end-user (less than 4.08 rebufferings on all NPs), which leads to lowering the amount of data that is only transmitted and not played out by the client. In the end, the resulting bandwidth consumption overhead decreases and MS-Stream-NA illustrates the benefits of relying on redundant data to prevent video streaming disruptions.

The bandwidth consumption overhead of the MS-Stream-oracle client is 6.25% which exactly corresponds to the percentage of transmitted data at the redundant bitrate. Indeed, MS-Stream-oracle does not necessitate any redundant GoPs as it adapts both the video quality and the used sources according to prior knowledge of the upcoming network condition fluctuations on each path. Opposite to MS-Stream-NA, MS-Stream-oracle directly exposes the disadvantages of relying on redundant data when the adaptation capabilities of the considered multi-source HAS solution are sufficient to avoid video playback interruptions.

Considering MS-Stream-BA and MS-Stream-BSA, they have a slightly greater bandwidth overhead than MS-Stream-oracle, respectively with averages of 6.52% and 6.34% on NPs #1-#2-#3, averages of 6.34% and 6.40% on NPs #4-#5-#6. Compared to the MS-Stream-oracle client, this increased bandwidth consumption overhead is due to the lower displayed video bitrate at the end-user's side resulting from the quality adaptation, which in turn affects the ratio between the amount of transmitted data and the displayed data.

In all these multiple-source applications, the number of servers simultaneously used by the clients is limited to three, which consequently limits the maximum possible bandwidth consumption overhead. However, in a more realistic scenario where the number of servers used by the consuming client for its streaming session is variable in time, the bandwidth overhead could represent a significant part of the transmitted data. As a result, the network throughput overhead of MS-Stream could represent a significant part of the throughput available at the content delivery network side. This issue is adressed in Chapter 4.

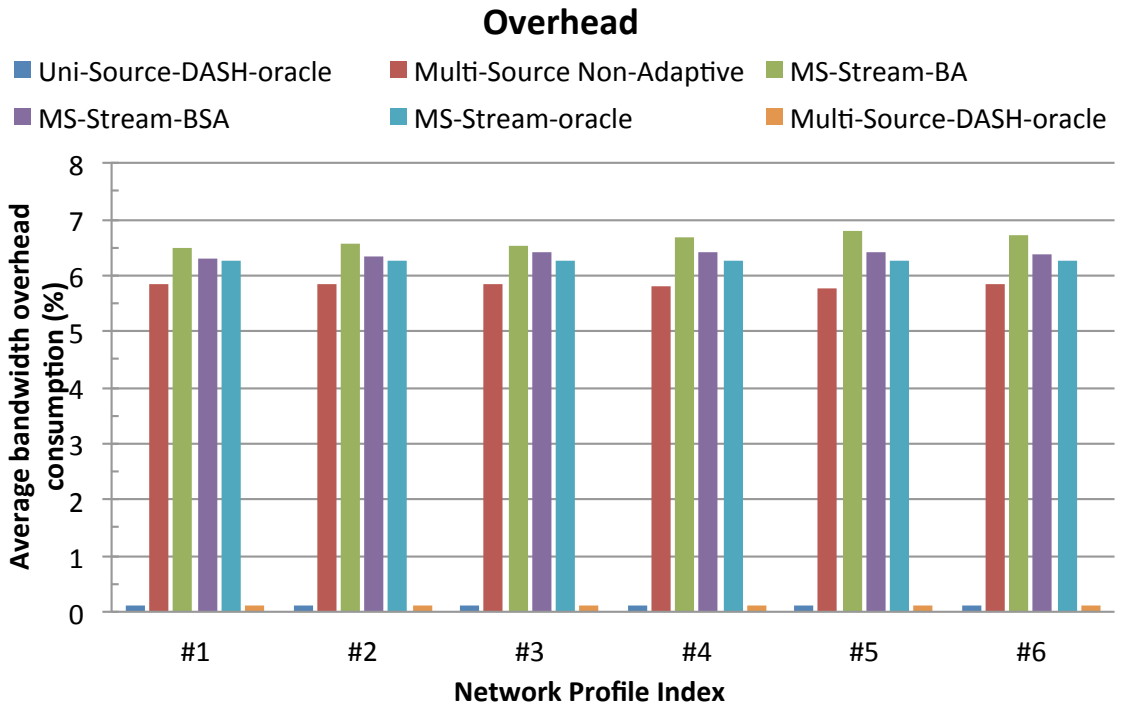

Figure 3.13 – Bandwidth consumption overhead

3.3.3.2 Impact of sub-segment processing on the server load

Regarding the impact of sub-segment generation on the server workload, we evaluated the processing time required to handle simultaneous requests in two scenarios. The first scenario consists of simulating the conditions of an HTTP server receiving multiple client requests for regular video segments at different qualities. In the second scenario, an MS-Stream server exposes its high level RESTful API on top of the HTTP protocol so as to generate multiple video sub-segments with random composition in terms of GoPs, including different qualities.

Both servers were based on the grizzly [Grizzly] implementation of HTTP in Java8, while the video sub-segment processing was implemented on top of the RESTful Jersey [Jersey] framework. Each server was required to handle a total of 10000 requests with a number of simultaneous requests indicated in the x-axis of Figure 3.14. Segment bitrates were uniformly selected between 500kbps and 6Mbps with a 500kbps bitrate steps between each quality. The redundant bitrate b_r was set to 200kbps.

Figure 3.14 reports the required time for each server to process 99% of the

clients requests. This evaluation focuses on the processing overhead only in order to expose the impact that the MS-Stream solution has at the service environment level compared to classic client-centric single-server HAS solutions. An almost linear relation between the MS-Stream and the HAS results is observed with an increase of the processing time in the MS-Stream case of 11% in average.

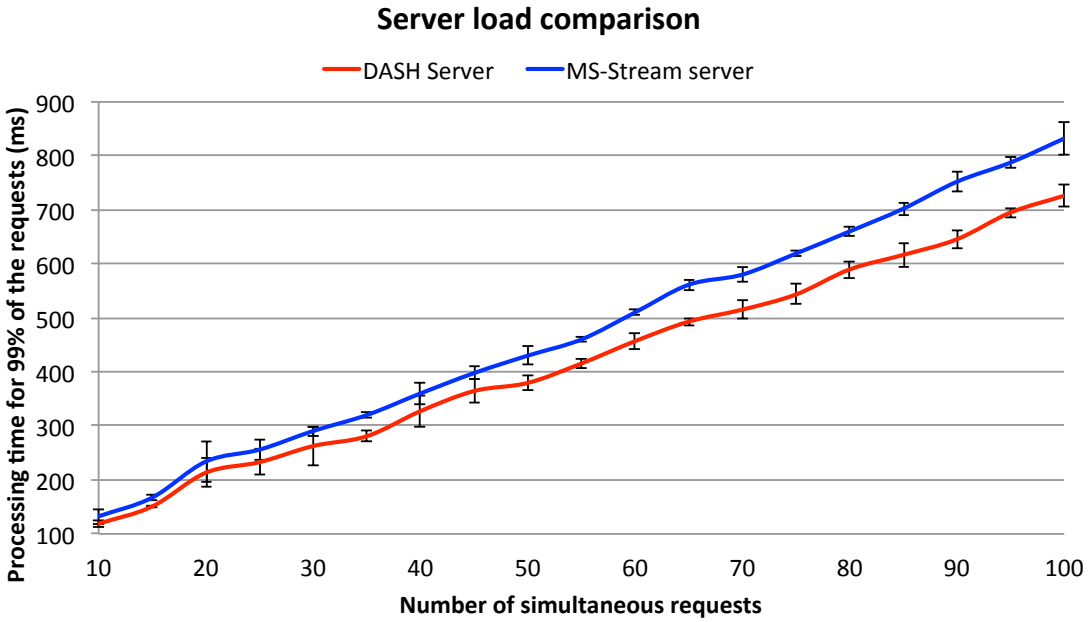

Figure 3.14 – Processing time for 99% of the requests

3.4 Conclusion

To conclude, this chapter introduces our proposal for a pragmatic evolution of client-centric HTTP adaptive streaming techniques toward supporting multiple-source streaming and increasing end-users' QoE. For each video segment, the consuming client requests several video sub-segments from multiple servers concurrently and specifies the composition of the latter sub-segments in terms of quality (i.e., bitrate, spatial resolution, etc...) and at the granularity of independantly playable video GoPs. By copying redundant GoPs into video sub-segments, this streaming solution can benefit from content sources with highly heterogeneous and volatile capacities (set-top boxes, end-users' devices, ISP's home-gateways) while avoiding the video streaming disruptions that represent the main cause of end-users' QoE degradation. The spectrum of adaptation possibilities covers video quality, number

and nature of servers involved in the streaming session, bandwidth consumption overhead, video sub-segment scheduling and sub-segment delivery synchronization. In a nutshell, we demonstrated that MS-Stream has the potential to deliver higher QoE than DASH with significant mean bitrate increase, less rebuffering, shorter start-up delays and less quality fluctuations at the cost of light overheads in terms of bandwidth consumption and processing time at the server side.

Chapter 4

MATHIAS: Multiple-source and adaptive streaming algorithms

"I love deadlines. I like the whooshing sound they make as they fly by"

— Douglas Adams

4.1 MATHIAS: Multiple-source and adaptive streaming algorithms

The MS-Stream solution presented in Chapter 3 is a pragmatic HAS-evolving solution where the simultaneous usage of servers is made possible so as to take advantage of the available bandwidth on multiple network paths and to provide the means to increase the end-user's perceived quality. MS-Stream defines the structure of the video sub-segment as well as the client/server functional architectures and advocates for a client-centric approach but enforces neither the way in which content is delivered from the server(s) to the client nor the consumption and adaptation behaviors that should be implemented. This field is intentionally left opened for other researchers and industrials to come up with their own solutions matching specific use-cases.

In this chapter, we propose to focus on content delivery adaptation mechanisms involved in increasing end-user's QoE for this client-centric multiple-source streaming solution and on the heterogeneity of resources made available at the network

and service infrastructure levels. Several issues arise when considering using multiple sources with heterogeneous capabilities for the delivery of multimedia content to a client. First, when a client attempts to obtain a video quality at a given Y bitrate, two types of bandwidth bottlenecks can prevent the delivery of the latter quality and can result in numerous video freezing events and degraded QoE: bandwidth bottleneck at the client-side -i.e., within the client's environment-, and bandwith bottleneck at the content delivery network side. In the event where the bottleneck takes place at content delivery side, adjusting the number of used sources permits to increase path diversity and to reach the desired network throughput necessary for the delivery of the Y video bitrate.

Secondly, as exposed in Chapter 3, the MS-Stream solution necessitates extra network throughput compared to the bitrate of the video being displayed on the end-user's terminal. This additional bandwidth is required for the resiliency of the streaming session to network outages and to suddenly low performing network paths or servers. Hence the MS-Stream solution overuses the service and network environment's capacities to achieve higher QoE but also impacts on the potential of the streaming system to provide as many clients as possible with the best video quality. Therefore, reducing and minimizing the latter overhead is an essential issue for augmenting the scalability of the MS-Stream solution, thus easing the adoption of the MS-Stream proposal for streaming actors.

Finally, from the point of view of a streaming provider (or content delivery network provider) owning multiple servers with heterogeneous capablities, the full utilization of the servers by the consuming clients is essential to control and minimize the deployment and operating costs. The scope of the MS-Stream solution covers the usage of content delivery sources with various throughput capacities (e.g., set-top-Boxes in end-users' homes with xDSL-like connectivity, virtual private servers, dedicated servers with very high throughput capacities, etc.) that should not be over or under-used. On the one hand, the over-usage of the streaming servers would result in low QoE at the client side. On the other hand, under-usage would result in greater operating costs than necessary. Thus, the open challenge is not only for the MS-Stream client to request the participation of servers in proportion to their available and observable capacities, but also to target a fair utilization of the resources made available in order to slowly converge toward the full utilization of the service infrastructure.

This chapter outlines *MATHIAS* (Multiple-source and Adaptive sTreamIng AlgorithmS) addressing the three above-described issues. MATHIAS is a sequence of

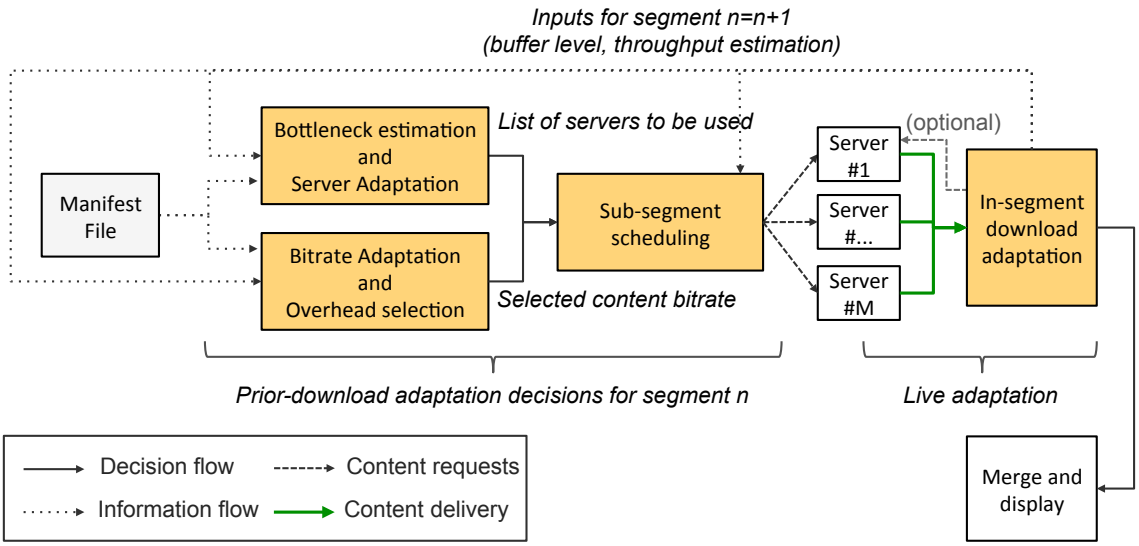

Figure 4.1 – MS-Stream adaptation and consumption algorithm. The *Bottleneck Estimation* and *Server Adaptation* module is only used every two segment downloads for the purpose of bottleneck estimation.

algorithms executed at client side, each solving one of the above-addressed issues. The purposes of MATHIAS are two-fold: (**1**) taking advantage of the multi-source capabilities offered by MS-Stream in order to utilize the most of the resources made available in a client-centric fashion while considering the heterogeneity of the capacities of each used server, (**2**) meeting the end-user's expectations in terms of target video bitrate while achieving as few video rebuffering events as possible (QoE-awareness).

MATHIAS is split into two main phases, as depicted in the MATHIAS decision flow in Figure 4.1, both related to the download of each video segment. The first phase consists of prior-download adaptation decisions composed of three steps: (1) overhead selection and bitrate adaptation to increase QoE and limit the bandwidth consumption overhead resulting from the redundancy of the MS-Stream approach; (2) bottleneck estimation and server adaptation to provide a flexible usage of network resources and to reach the required network throughput matching the target bitrate; (3) sub-segment composition and scheduling decisions in order to adapt sub-segments' bitrate to the observable resource heterogeneity.

In this first phase, MATHIAS decides on the video quality requested to the servers, on the number of servers to be simultaneously used by the client and on the actual composition of the video sub-segments requested to each servers. In chapter 3, the MS-Stream client was using a fixed number M of servers, wherease in MATHIAS,

the client is given to possibility to adjust the number of used servers from 1 to M_{max}. By relying on the information locally available (e.g., bandwidth estimation on each path, buffer occupancy, available servers listed in the manifest file, number of GoPs per video segment), MATHIAS attempts to obtain a target Y video bitrate while dynamically adjusting and minimizing the bandwidth consumption overhead below a given O_{max} percentage of the transmitted data.

The second phase consists in focusing entirely on increasing the end-user's QoE by monitoring the download progress of video sub-segments and performing in-segment download adaptation. To this end, MATHIAS incorporates a set of three in-segment download adaptation rules so as to ensure smooth video playback even in the event of suddenly low performing communication channels.

4.1.1 Prior download: Overhead selection and bitrate adaptation for QoE enhancement and bandwidth consumption overhead limitation

In this step, the MS-Stream client performs quality adaptation and addresses the issue of limiting the bandwidth overhead consumption resulting from the redundancy of GoPs in the sub-segments delivered to MS-Stream clients. Although the video quality adaptation logic is left undefined, MATHIAS enforces that this step is executed before taking decisions on the sub-segments composition and their delivery assignation to the servers, as depicted in Figure 4.1.

In the previous chapter, we exposed the MS-Stream functioning and its trade-off in using additional bandwidth to provide streaming session resiliency to server and in-network impairements so as to improve end-users' QoE. The bandwidth consumption overhead was defined as the percentage of data transiting on the network, which does not take part in the displayed video on the end-user's terminal:

$$Overhead = 1 - \frac{PlayedData}{TransmittedData} \tag{4.1}$$

In addition, the ability of MS-Stream clients to simultaneously use as many servers as possible was identified to be limited by the bandwidth consumption overhead that linearly increases with the number of used servers. Therefore, while a high number of servers used simultaneously increases the throughput delivered to clients, it also leads to critically high levels of overhead that can end up exceeding by far the required throughput to deliver a given video stream. The capacity of a streaming system to accommodate as many clients as possible with a Y Mbps

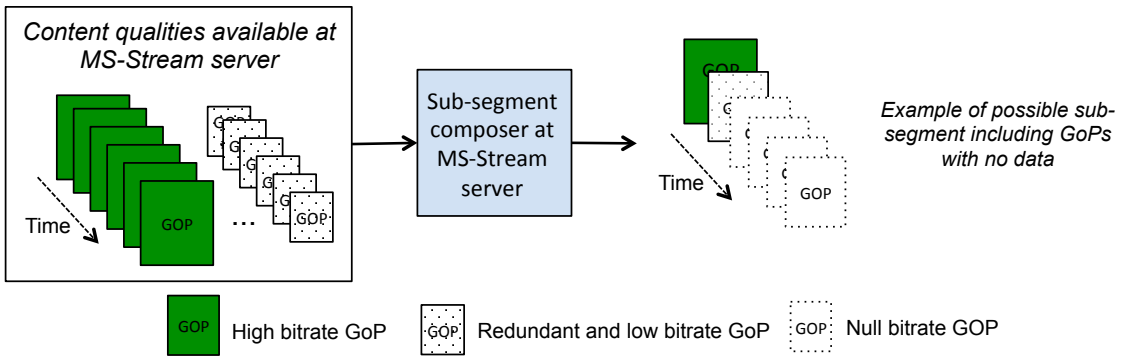

Figure 4.2 – Sub-segment generation with GoPs having no payload data.

content bitrate is a function of its global aggregated upload throughput from all servers. Consequently, the effective utilization of this upload throughput capacity is linearly decreased by the bandwidth consumption overhead, hence impacting the infrastructure scalability, which in turn negatively impacts on the end-users' QoE. Minimizing this overhead is therefore an essential issue for the scalability of the MS-Stream solution and its capability to deliver high QoE to its users.

MATHIAS tackles this bandwidth consumption overhead problem by limiting and minimizing the redundant data (i.e., the GoPs at the redundant bitrate b_r) copied into sub-segments. The approach is based on the fact that providing resiliency to the streaming session is not always profitable in improving the end-users' QoE, especially when the buffered content allows sufficient time to react to impairments that can affect the streaming session's continuity or the displayed video quality.

We propose to limit the overhead by lowering the amount of redundancy according to the buffer occupancy. To that end, the structure of sub-segments is made more flexible than the one described in the previous chapter, and sub-segments can now include GoPs with no payload data composing them, as exposed in Figure 4.2. As a result, sub-segments are not necessarily independently decodable and playable. Therefore, the synchronization rules originally designed for MS-Stream in section 3.2.4.1 are no longer applicable. Thus, we introduce a new set of enhanced rules, further discussed in section 4.1.4.

For every segment n to be retrieved, the MS-Stream client adjusts its bandwidth overhead percentage $O_n\%$ according to the buffer occupancy $bufLevel_n$ before the download of the segment. Figure 4.3 and equation 4.2 expose the proposed relation between the buffered content and the redundant data percentage selection. A maximum level of resiliency is ensured by requiring a maximum percentage O_{max} of redundant GoPs when the buffer level is below a pre-defined lower bound ε. Sim-

ilarly, a minimum percentage O_{min} of redundant GoPs is set when the buffer level exceeds a given upper bound σ, the value of O_{min} can be 0. Finally, when the buffered content duration is between σ and ε, the value of O_n is a decreasing linear function of $bufLevel_n$.

$$O_n = \begin{cases} O_{max} & \text{if } bufLevel_n \leq \varepsilon \\ \dfrac{(O_{min} - O_{max}) \cdot bufLevel_n + \sigma \cdot O_{max} - \varepsilon \cdot O_{min}}{\sigma - \varepsilon} & \text{if } \sigma \geq bufLevel_n > \varepsilon \\ O_{min} & \text{if } bufLevel_n > \sigma \end{cases} \tag{4.2}$$

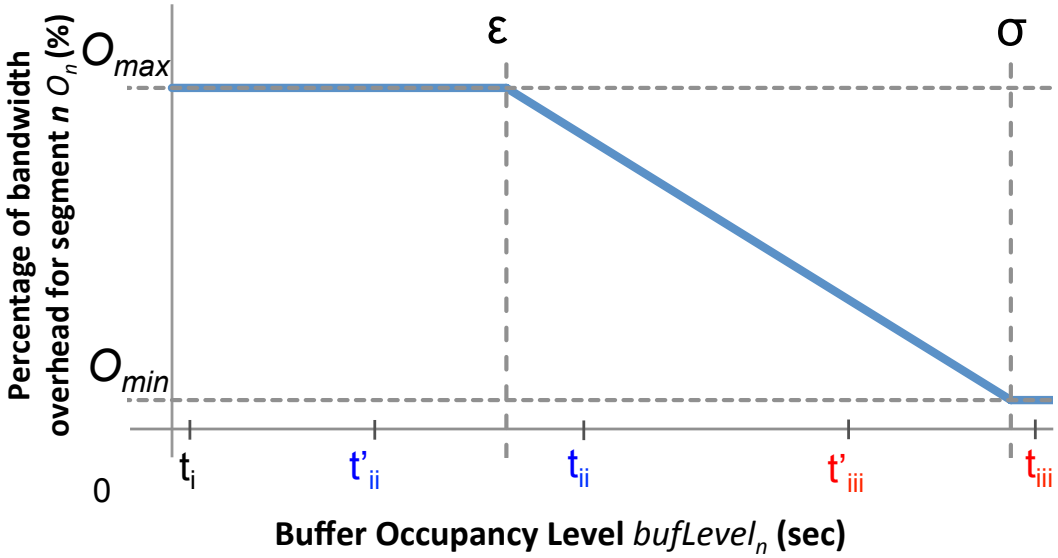

Figure 4.3 – Relation between overhead selection and buffer occupancy level (section 4.1.1); and threshold used for in-segment download adaptation (section 4.1.4)

The sub-segment scheduler is in charge of mapping this percentage to the closest number of redundant GoPs to be copied into sub-segments in order to create the highest number of independently playable sub-segments. Although a greater value of O_{max} increases the maximum overhead, it also provides better resiliency to network outages as it allows MS-Stream clients to request a greater number of independently playable sub-segments.

Regarding the extra throughput δ_n (referred to as δ in chapter 3) required to deliver the video bitrate b_i when using sub-segments with redundant GoPs, the

computation method changes because the MS-Stream client now has control over the percentage of bandwidth overhead O_n it can use. By definition, we have:

$$O_n = \frac{\delta_n}{\delta_n + b_i} \tag{4.3}$$

Therefore, we can derive δ_n in the following equation:

$$\delta_n = \frac{O_n \cdot b_i}{1 - O_n} \tag{4.4}$$

4.1.2 Prior-download: Sub-segment scheduling for adaptability to communication channel heterogeneity

As stated in the introduction of this chapter, the full utilization of the available servers is essential to control and minimize the deployment and operating costs of the streaming infrastructure.

In the MS-Stream solution, the consuming clients retrieve content from servers with highly heterogeneous capacities in terms of available throughput. Although the challenge is for the MS-Stream client to request the participation of the considered servers in proportion to their available and observable capacities, there are unwanted side effects that can lead to the under-usage of servers composing the deployed streaming infrastructure. Indeed, when a given MS-Stream client utilizes a set S_{M_n} of M_n servers simultaneously with heterogeneous capacities for the download of segment n, the least performing servers may expose to the client significantly lower available throughput in comparison with others. As a result, only the best performing servers may be utilized before their available throughput fall sufficently low for the least performing servers to actually be considered as valuable elements to increase the achievable throughput. Hence it can leave idle or under-used some resources that could participate in increasing the perceived quality by end-users.

To tackle this problem, MATHIAS incorporates algorithm 1 which enable the MS-Stream client to act in two different modes: *conservative utilization* (lines 3-16) of the available resources by not exploiting the considered servers up to their full bandwidth capacities and by relying on the low performing ones in the first place; or *prorated utilization* (lines 17-24) of the available resources by requesting the participation of the servers directly in proportion to their capacities. Algorithm 1 decides which mode to operate by detecting whether the MS-Stream client is facing highly heterogeneous capacities with important bandwidth availability. The incentive is to employ the conservative mode when the available network throughput is sig-

nificantly higher than the desired video bitrate and is heterogeneously distributed among the considered servers. In such case, the conservative mode permits to use all the available resources without overloading the low-performing servers and without requesting most of the GoPs at the desired quality from the best-performing servers. Contrarily, when the available resources are sufficient for the delivery of the demanded quality but not sufficiently high to rely on the conservative mode, then the algorithm switches to the prorated mode to enable the usage of all servers up to the proportion of their throughputs. In doing so, the client steers its utilization of the servers in aiming at obtaining the required throughput in a conservative manner.

Table 4.1 – Variables used for segment scheduling in MATHIAS

Symbol	Description
M_n	Number of servers used for segment n: $M_n \in [1..M_{max}]$
$S_{M_n,n}$	Set of M_n server(s) used for segment n: $S_{M_n,n} = \{s_{m,n}\}_{m\in[1..M_n]}$
$X_{M_n,n}$	The set of observed throughput for each server $\in S_{M_n,n}$ for segment n: $X_{M_n,n} = \{x_{m,n}\}_{m\in[1..M_n]} = \{throughput(\{s_{m,n}\})\}$
$G_{M_n,n}$	Set of number of group of pictures $G_{M_n,n} = \{g_{m,n}\}_{m\in[1..M_n]}$ assigned to each server $s_{m,n} \in S_{M_n,n}$ for the download of segment n
$\widehat{S_{M_n,n}}$	Set of M_n server(s) used for segment n, sorted in ascending order of observed throughput: $\widehat{S_{M_n,n}} = \{\widehat{s}_{m,n} \in S_{M_n,n} \vert \widehat{x}_{m,n} > \widehat{x}_{m+1,n} \forall m \in [1..M_n]\}$
thr_n	Observed throughput during the download of segment n: $thr_n = \sum_{k\in[1..M_n]} x_{k,n}$
$bgop_i$	Required network throughput for the delivery of one GoP at bitrate i in one segment $bgop_i = \frac{b_i}{G_{Tot}}$
F	High heterogeneity detection factor (>1)
tol	Tolerance threshold: $1 > tol > \frac{F \cdot b_i \cdot G_{Tot} \cdot bgop_i}{thr_n^2}$

Algorithm 1 Adaptability to heterogeneous network resources

Inputs: $S_{M_n,n}, X_{M_n,n}, thr_n, F, b_i, M_n, M_{n+1}$
Output: $G_{M_{n+1},n+1}$

1: $S_{M_{n+1},n+1} \leftarrow S_{M_n,n}$
2: $G_{M_{n+1},n+1} \leftarrow 0$
3: **if** $thr_n > F \cdot b_i$ **then** ▷ Conservative usage of resources
4: $\widehat{S_{M_n,n}} \leftarrow sorted\{S_{M_n,n}\}$
5: **for** m in 1..M_n **do**
6: **if** $\frac{\widehat{x}_{m,n}}{bgop_i} \cdot tol < 1$ **then**
7: remove $\widehat{s}_{m,n}$ from $S_{M_{n+1},n+1}$
8: $M_{n+1} = M_n - 1$
9: **else**
10: $g_{m,n+1} \leftarrow floor\left[\frac{\widehat{x}_{m,n}}{bgop_i} \cdot tol\right]$
11: $G_{M_{n+1},n+1} \leftarrow g_{m,n+1}$
12: **if** $card\left|G_{M_{n+1},n+1}\right| == G_{Tot}$ **then**
13: break
14: **end if**
15: **end if**
16: **end for**
17: **else** ▷ Prorated usage of resources
18: **for** m in 1..M_n **do**
19: $g_{m,n+1} = floor\left[\frac{x_{m,n}}{thr_n} \cdot G_{Tot}\right]$
20: $G_{M_{n+1},n+1} \leftarrow g_{m,n+1}$
21: **if** $card\left(G_{M_{n+1},n+1}\right) == G_{Tot}$ **then**
22: break
23: **end if**
24: **end for**
25: **end if**

When the cummulated throughput thr_{n-1} observed by the client exceeds by a factor F the requested content bitrate b_i (line 3 of algorithm 1) for the download of segment n, the MS-Stream client decides to conservatively utilize the available throughput of each server. Then, the MS-Stream client sorts the servers by ascending order of observed throughput (line 4) in order to begin with using the lowest performing servers first. Based on the throughput estimation $x_{m,n-1}$ of each server, and based on the throughput $bgop_i$ required to deliver one GoPs at bitrate b_i in real time ($bgop_i = \frac{b_i}{G_{Tot}}$) estimates are made on their ability to deliver at least one GoPs

in real time (line 6 $\frac{\widehat{x}_{m,n-1}}{bgop_i} \cdot tol < 1$). A tolerance factor tol, lower than 1, is introduced in the latter decisions in order to conservatively utilize the low-throughput servers. The value of the tol parameter should be carefully chosen in order to ensure that the G_{Tot} GoPs can actually be delivered throughput-wise. Hence, we define the following conditions:

$$\begin{cases} thr_{n-1} > F \cdot b_i \\ \sum_{m=1}^{M_{n-1}} \frac{x_{m,n-1}}{bgop_i} \cdot tol \geq G_{Tot} \end{cases} \tag{4.5}$$

We can deduce a condition on the value of the tol parameter:

$$tol > \frac{F \cdot b_i \cdot G_{Tot} \cdot bgop_i}{thr_{n-1}^2} = \frac{F \cdot b_i^2}{thr_{n-1}^2} \tag{4.6}$$

The servers that are unable to deliver at least one GoP at the bitrate b_i in real time are removed from the set S_{M_n} of servers to be used for the next segment n (lines 7-8), whereas the other servers are assigned the delivery of $floor\left[\frac{\widehat{x}_{m,n}}{bgop_i} * tol\right]$ GoPs (lines 10-11). In our study, the value of F is arbitrarily chosen, and is set to 2.

Finally, if the client is not facing heterogeneous conditions that necessitate the conservative utilization of available resources, the MS-Stream client operates the second mode of algorithm 1 and assigns to each server a number of GoPs proportional to their throughput: $floor\left[\frac{x_{m,n-1}}{thr_{n-1}} \cdot G_{Tot}\right]$ (line 19). The MATHIAS's sub-segment scheduler is in charge of running the latter algorithm and of designing GoPs compositions of sub-segments.

4.1.3 Prior-download: Bottleneck estimation and server adaptation for a flexible usage of network resources

Providing flexibility in the amount of heterogeneous network resources used to reach a targeted video bitrate (Y Mbps) is the third objective of MATHIAS, and is achieved through this step. The algorithm 2 lets the MS-Stream client in charge of deciding on the number of servers that should be used for the download of the next segment. The main challenge is to properly choose the number of servers when the available throughput in the network is not sufficient to sustain the delivery of the video at the previously selected bitrate b_i. Two main reasons can explain the lack of throughput received by the client: (1) the presence of a bandwidth bottleneck

within the client's environment (over-used WiFi, etc.); (2) the presence of a bandwidth bottleneck at the content delivery network side (network congestion, packet loss, servers reaching their maximum upload throughput capacity).

We propose a method to detect and estimate both the presence and the type of bandwidth bottleneck that prevents the delivery of the video target bitrate. Figure 4.4 provides a high level understanding of the estimation method. Table 4.2 lists the variables used for bottleneck estimation, and Table 4.3 exposes the different cases where bottlenecks are detected and estimated. Then, based on the estimated bottleneck type, the MS-Stream client is able to take decisions regarding the addition or removal of servers to the streaming session.

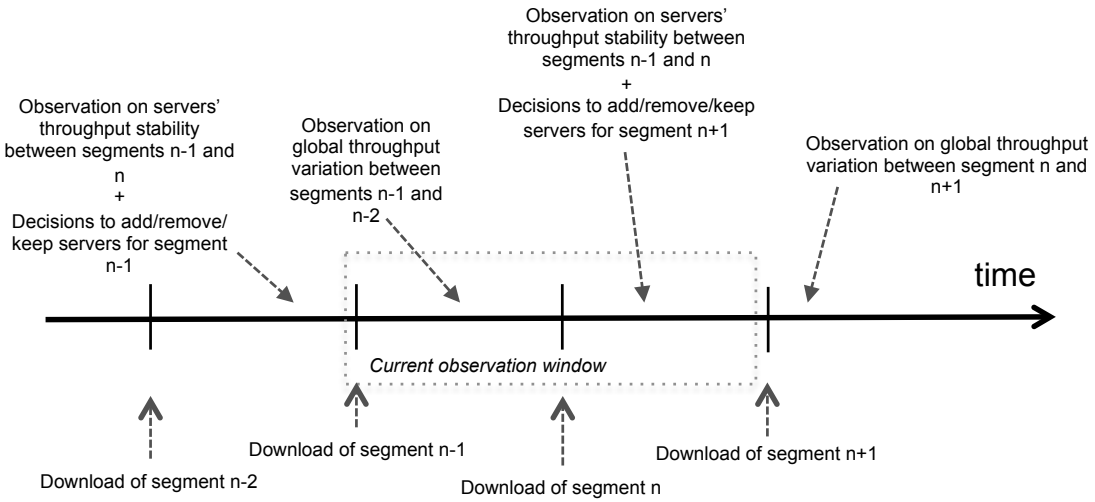

Figure 4.4 – High level understanding of the bottleneck estimation method -relying on an two-segment-download-duration observation window-

The unstable nature of multiple TCP connections competing on the same link with limited throughput capacity is at the basis of the proposed bandwith bottleneck estimation approach. Indeed, in the event of an over-used network link at the client side, the throughput obtained from a given server for the duration of the download of a given sub-segment may highly differ from the throughput obtained from the same server during the download of the next sub-segment. Oppositely, when the available bandwidth at the client side is sufficient to let flow a video stream at the selected bitrate, the throughput obtained from a given server is quite stable between two consecutive sub-segment downloads. Therefore, the proposed bottleneck estimation method relies on monitoring the stability of the throughput delivered by each server during a two-segment-download-duration observation window.

Table 4.2 – Variables used for bandwidth bottleneck estimation

Symbol	Description
δ_n	Allowed extra network throughput consumption (deriving from the selected value of O_n): $\delta_n = \frac{O_n \cdot b_i}{1 - O_n}$
T_n	Minimum throughput required to obtain the targeted video quality ($T_n = Y + \delta_n$)
Δthr_n	Throughput difference between 2 segments: $\Delta thr_n = thr_n - thr_{n-1}$
ΔM_n	Evolution of the number of simultaneously used servers between two segment downloads:$\Delta M_n = M_n - M_{n-1}$
bnd_n	Throughput bound determining whether the global throughput is stable: $bnd_n = \mu * thr_n$
μ	Percentage used in the computation of bnd_n
λ_n	The number of servers which delivered throughput varies highly $\lambda_n = card(\{s_{m,n} \in S_{M_n,n} : \frac{\lvert x_{m,n} - x_{m,n-1} \rvert}{x_{m,n-1}} > \theta\})$
θ	Percentage determining whether the throughput of a given server varies highly in time
λ_{low}	Lower bound threshold for bottleneck detection
λ_{high}	Upper bound threshold for bottleneck detection

Table 4.3 – Bandwidth bottleneck estimation method

Observed ΔM_{n-1}	Observed Δthr_{n-1}	Observed λ_n	Bottleneck estimate
> 0	$\in [bnd_{n-1}; +\infty]$	$< \lambda_{Low}$	Server-side
0	$*$	$< \lambda_{Low}$	Server-side
$*$	$\in [-\infty, bnd_{n-1}]$	$> \lambda_{High}$	Client-side

The duration of this observation window is dimensioned in order to monitor (1) the impact of server add/keep/remove decisions (ΔM_{n-1}) on the difference of throughput globally available (Δthr_{n-1}) between the download of two video segments, and (2) the system's stability in terms of the number of servers (λ_n) for which the delivered throughput varies highly throughout the window's duration.

In the case of bandwidth bottleneck at the content delivery network side, the delivered throughput is most likely limited by the number of used servers and the diversity of used paths rather than by the client's environment. Therefore, the

method to estimate the presence of this bottleneck performs as follows and exposes the two first cases of Table 4.3:

Case 1: If the number of used servers was precendently increased ($\Delta M_{n-1} > 0$) after the download of a video segment $n-2$, and is followed by improvements in the obtained throughput (i.e., Δthr_{n-1} is greater than a lower bound bnd_{n-1}) while the individual throughput $x_{m,n}$ from most of the servers do not vary highly ($\lambda_n < \lambda_{low}$), then the type of bottleneck is considered at the content delivery network side.

Case 2: When the MS-Stream client increases the number of simultaneously used servers after obtaining the video segment $n-1$, but the obtained throughput from all servers as well as their individual throughput keep steady, the bottleneck is most probably not located at the client side, and the content delivery network side is considered responsible by the MS-Stream client for the lack of delivered throughput.

For the case of detecting bandwidth bottleneck at the client-side (case 3 of Table 4.3), regardless of the decisions made on adding/removing servers to the session, if the total throughput declines while the throughput of most servers highly fluctuates between the last two segment requests ($\lambda_n > \lambda_{high}$), there are high chances that the client's environment is responsible for the observed throughput decrease. In this case, a client-side bottleneck is detected. The efficiency of the proposed bottleneck estimation method is evaluated in section 4.2.

The overall server adaptation decisioning is run by the MS-Stream client and is outlined in algorithm 2. The adaptation of the number of simultaneously used servers is only triggered when the obtained throughput does not match the minimum throughput T_{n+1} required to obtain the targeted video quality for segment $n+1$ (lines 1-6 of algorithm 2). Then, the client attemps to identify where the bandwidth bottleneck is occuring by computing the required variables in order to further run the bottleneck estimation method (lines 8-12).

Algorithm 2 AIMD Server Adaptation

Inputs: $\mu, \theta, \lambda_{low}, \lambda_{high}, thr_{n-2}, thr_{n-1}, M_{n-2}, M_{n-1}, M_n, S_{M_n,n}, X_{M_{n-1},n-1}, X_{M_n,n}, bufLevel_{n+1}$

Output: M_{n+1}

▷ Computing bandwidth consumption overhead

▷ and necessary extra network throughput

$O_{n+1} \leftarrow computeAllowedOverhead\%(bufLevel_{n+1})$

$\delta_{n+1} = \dfrac{O_{n+1} \cdot b_i}{1 - O_{n+1}}$

$T_{n+1} = Y + \delta_{n+1}$

if $thr_n < T_{n+1}$ **then**

▷ Computing variables used for bottleneck-type estimation

$\Delta M_{n-1} = M_{n-1} - M_{n-2}$

$bnd_{n-1} = \mu * thr_{n-1}$

$\lambda_n = card(\{s_{m,n} \in S_{M_n} : \dfrac{|x_{m,n} - x_{m,n-1}|}{x_{m,n-1}} > \theta\})$

$\Delta thr_{n-1} = thr_{n-1} - thr_{n-2}$

$bottleneck \leftarrow estimateBottleneck(\Delta M_{n-1}, \Delta thr_{n-1}, \lambda_n, bnd_{n-1}, \lambda_{low}, \lambda_{high})$

▷ AIMD-based adaptation of the number of used servers

if bottleneck == client **then**

$M_{n+1} \leftarrow M_n/2$

else if bottleneck == server **or** M_n is the same for l consecutive segments **then**

$M_{n+1} \leftarrow M_n + 1$

end if

else

$M_{n+1} \leftarrow M_n$

end if

In order to handle network congestion, the standard Additive-Increase Multiplicative-Decrease (AIMD) method is adopted because it has the advantage of locally converging to an optimum resource utilization when facing network congestion. In the presence of a client-side bandwidth bottleneck, the number of servers is halved so as to avoid erroneous sub-segment scheduling decisions due to the unstable and competing multiple TCP connections (lines 14-15). When a server-side bottleneck is detected, the number of servers is incremented (by randomly selecting a server in the server list made available in the manifest file) aiming at reaching a higher delivered throughput (lines 16-17). From the client side point of view, the type of bottleneck cannot always be determined. Therefore, should the bottleneck-type be unknown for a sequence of l segment downloads, the number of servers is incremented to trigger observable behaviors favorable for bandwidth bottleneck-type

detection (line 16-17).

When the decision to add a server is taken, the MS-Stream has no previous record of the throughput provided by this new server. Consequently, the client requests the lowest possible contribution from this server (one GoPs at the redundant bitrate) in order not to negatively impact the end-users' QoE if the associated sub-segment arrives late. It ought to be noted that if a server cannot deliver at least this GoPs in the given delay, it is removed from the list of used servers. Finally, since the observation window used for bandwidth bottleneck estimation is the size of two video segments, the client can only run algorithm 2 every two segment downloads.

4.1.4 In-segment download adaptation: resiliency to heterogeneous network characteristics

The MS-Stream algorithm introduces adaptation actions during the sub-segment download phase in order to ensure smooth video playback experience in the event of network paths or servers not performing as originally expected (with regards to the throughput previously estimated in MATHIAS). MATHIAS defines three in-segment adaptation rules aiming at maintaining the video quality displayed to the end-users, avoiding video stalls, and sustaining the buffer occupancy level to keep the lowest possible amount of bandwidth overhead consumption. Figure 4.5 illustrates that the latter rules are periodically being called by the MS-Stream client until the sub-segment downloads terminate. Then the sub-segments are merged and the MS-Stream client relies on MATHIAS to move on to the next video segment download.

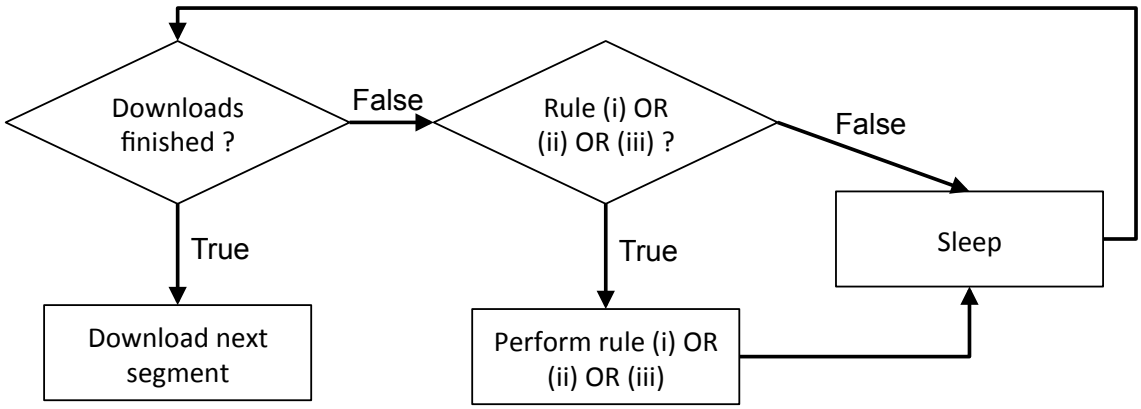

Figure 4.5 – In-segment adaptation rule executions during sub-segment downloads

Once the client sends sub-segment requests to the servers, it is expecting to

be able to reconstruct a video segment composed of G_{Tot} GoPs with each GoPs having a b_i video bitrate. This *expected segment* is denoted $exp = \{b_i, ..., b_i\} = \{exp_k\}_{k\in[1..G_{Tot}]}$. During the download progress of the requested sub-segments, the client progressively re-composes a video segment from the delivered GoPs. We name this segment *current segment* and we denote it $cs(t) = \{cs_k(t)\}_{k\in[1..G_{Tot}]}$. Whenever the client retrieves GoPs from the sub-segment deliveries, it updates $cs(t)$ with the GoPs with the highest available bitrate, i.e., replacing the redundant GoPs at the redundant bitrate if they are present in $cs(t)$. $G_{rcv}(t)$ defines the number of GoPs that are received and that are useful in the re-composition of a playable video segment -i.e., two GoPs available at both bitrate b_i and b_r only count as 1 in the computation of $G_{rcv}(t)$. We clearly define G_{rcv} in equation 4.7:

$$G_{rcv} = \sum_{k=1}^{G_{Tot}} cs(k) * \text{Ш}_k \tag{4.7}$$

where $\text{Ш}_k = 0$ if $cs(k) = 0$ and $\text{Ш}_k = \dfrac{1}{cs(k)}$ otherwise.

Among the sub-segments being delivered, some of them (the ones fully composed of GoPs containing video data) can be independently decoded and displayed while some others require the GoPs of other sub-segments. $M_{n,i}(t)$, $M_{n,d}(t)$ respectively denote the number of servers delivering independent and dependent sub-segments at a given time t. $M_{n,done}(t)$ is the number of servers that are done delivering sub-segments for segment n at time t and $bufLevel(t)$ denotes the buffer level occupancy in seconds at time t.

The three in-segment download adaptation rules introduces 5 time thresholds $t_i < t'_{ii} < t_{ii} < t'_{iii} < t_{iii}$ depicted in Figure 4.3. The rules are designed as follows:

(i) for a given content segment, if at least one playable video segment can be reconstructed before $bufLevel(t)$ falls below a given threshold t_i, i.e., if $G_{rcv} = G_{Tot}$, then all the other sub-segment downloads are abandoned. This reactive rule ensures uninterrupted video experience while providing a temporary sub-optimal visual quality to the end-users. By monitoring sub-segment download progress, estimations are made on sub-segments completion time;

(ii) in the event where there is at least one server delivering an independent sub-segment -i.e., $M_{n,i}(t) \geq$ 1- and $bufLevel(t) > t_{ii}$, the client estimates

the remaining delivery times for the downloads of the sub-segments of the $M_{n,i}(t)$ servers. If the remaining duration of at least one sub-segment exceeds $bufLevel(t) - t'_{ii}$, then the client will attempt to handover the delivery of the missing GoPs in $cs(t)$ to the most performing available server (throughput-wise) $\widehat{s} = s_{m,n} \in \{\{S_{M_{n,i}(t)} \cup S_{M_{n,done}(t)}\} : x_{m,n} = max[X_{M_n}]\}$. First the client determines $\widehat{s}$ and computes the optimal sub-segment *optSeg* that could be delivered from $\widehat{s}$ before $bufLevel(t)$ falls below t'_{ii} -i.e., in $bufLevel(t) - t'_{ii}$ seconds-. This optimal segment *optSeg* can be composed of GoPs at bitrates different from the originally selected one during the bitrate adaptation strategy. This is made possible so that the client is able to compute the composition of the *optSeg* in a way that will maximize the visual quality displayed to the end-users. *optSeg* is computed as follows:

$$optSeg = argmin_{\{seg_k\}k\in[1..G_{Tot}]}\left\{\sum_{k=1}^{G_{Tot}}\left|seg_k + cs_k(t) - exp_k\right|\right\} \tag{4.8}$$

under the following constraint:

$$optSeg_k + cs_k(t) \neq 0, \forall k \in [1..G_{Tot}] \tag{4.9}$$

If such a sub-segment exists, the late sub-segment deliveries from the $M_{n,i}(t)$ servers are abandonned by the client and the new sub-segment is requested from $\widehat{s}$.

(iii) in the event where all servers are delivering dependent sub-segments, i.e., the value of $M_{n,d}(t)$ is $M_n - M_{n,done}(t)$, and $bufLevel(t) > t_{iii}$, the client estimates the remaining delivery time for the downloads of the sub-segments of the $M_{n,d}(t)$ servers. If the remaining duration of at least one sub-segment exceeds $bufLevel(t) - t'_{iii}$, then the client will attempt to handover the delivery of the missing GoPs in $cs(t)$ to server $\widehat{s} = s_{m,n} \in \{\{S_{M_{n,d}(t)} \cup S_{M_{n,done}(t)}\} : x_{m,n} = max[X_{M_n}]\}$. The client determines $\widehat{s}$ and computes the optimal sub-segment *optSeg* that could be delivered from $\widehat{s}$ before $bufLevel(t)$ falls below t'_{iii}. The same method used in rule (iii) is employed to compute *optSeg*. As in rule (ii), if *optSeg* exists, the late sub-segment deliveries from the $M_{n,d}(t)$ servers are abandonned by the client and the *optSeg* is requested from $\widehat{s}$.

These last two reactive rules provide the means to re-assign the delivery of the missing GoPs to the most available server in order to provide uninterrupted streaming and maximize the displayed quality. Moreover, by setting $t_i < t_{ii} < t_{iii}$ (Fig-

ure 4.3) with t_{iii} greater than σ, rule (ii) and (iii) seek to maintain the buffer level in its area and to keep the lowest possible amount of overhead (between ε and σ for rule (ii); and as close as possible to the upper bound σ for rule (iii)).

4.2 MATHIAS Evaluation

MATHIAS is implemented into the MS-Stream client within the dash.js video player of DASH-IF. In order to perform a complete evaluation of MATHIAS, we firstly expose the advantages of each of the proposed algorithms in a controlled environment by playing out a pre-defined scenario. Secondly, we focus on the bottleneck estimation method efficiency. Finally, we empirically evaluate the MS-Stream behavior in a real deployed environment over the Internet, similar to the way an Over-The-Top provider would use it. In our experiments, we use the 10-minute Big Buck Bunny video (with 6-second segments, each containing 12 GoPs) encoded at 8 different bitrates and over different spatial resolutions as described in Table 4.4. Because GoPs of different spatial resolutions cannot be copied into a video segment without leaving unalterned the video codecs standard compliance, one redundant bitrate is defined for each spatial resolution.

4.2.1 MATHIAS functional validation

For the purpose of this study, we use a set of 10 MS-Stream servers provisioned with the video and one client. The setup is located in a controlled network environment as shown in Figure 4.6. The servers and the client have a traffic shaper module on the network interface card to control the inbound and outbound throughput. The equivalent of a 25 Mbps aggregated upload throughput is fairly distributed over the 10 servers, each having a 2500 kbps upload capacity. Regarding the client, it is first set with an unlimited throughput capacity on its link. In the functional scenario described below, throughput limitation is applied on the client's link.

Additionally, a 100ms delay is applied to each packet transiting through the network interface card of each server. The target video bitrate Y is 8Mbps. The maximum and minimum allowed bandwidth consumption overhead percentages are set to $O_{max} = 10\%$ when the buffered content duration exceeds $\sigma = 25$ sec, and $O_{min} = 0\%$ when the buffered content duration falls below $\varepsilon = 6$ sec. We set the in-segment adaptation rules' thresholds $t_{ii} = \varepsilon$ and $t_{iii} = \sigma$. Finally, the thresholds involved in the bottleneck estimation method are set as follows: the normalized

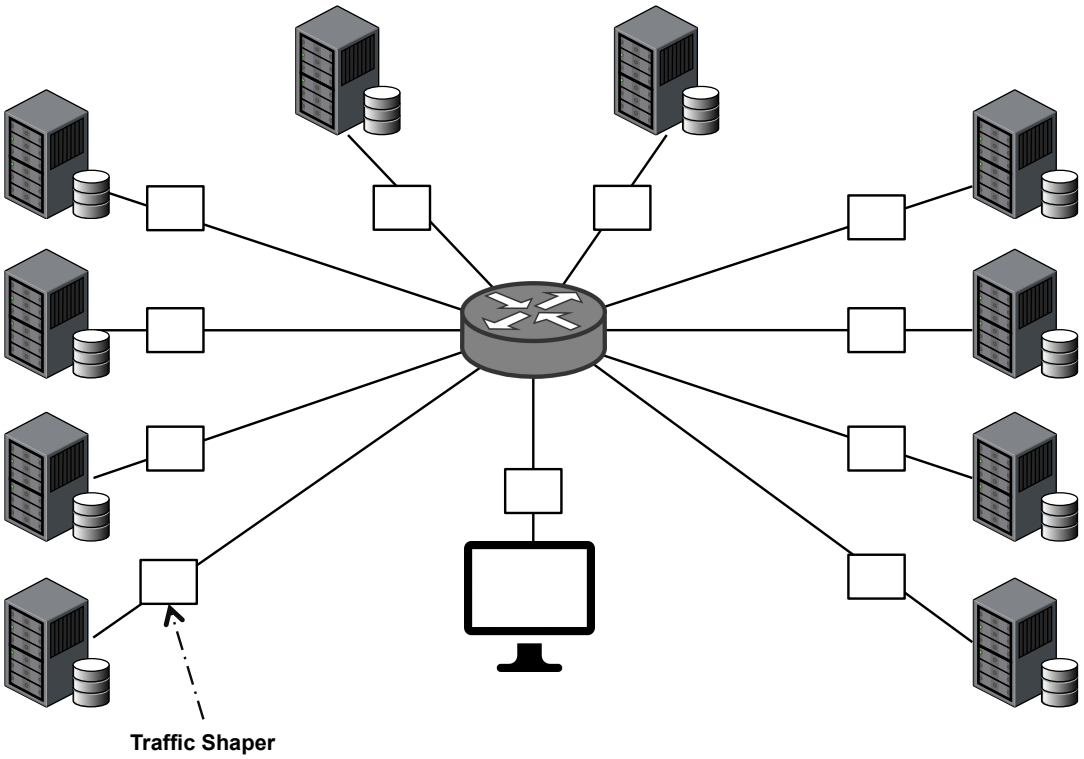

Figure 4.6 – Network setup including 10 MS-Stream servers

throughput difference determining whether the throughput of a given server varies highly in time $\theta = 30\%$; the lower and upper bound on λ_n are $\lambda_{low} = 25\%, \lambda_{high} = 75\%$.

We monitor over time the buffer level in relation with the overhead variation exposed in Figure 4.7. The number of used servers for each video segment is reported in Figure 4.8 along with the ID of each used server and the times when the delivery of sub-segments are cancelled or re-assigned to some servers. The sum of the throughputs observed on each server for the download of each 6-second segment is shown in Figure 4.9. Figure 4.10 presents the displayed video bitrate along with the throughput used for transmitting the redundant GoPs at the redundant bitrate b_r, and the cumulated throughput observed by the client on all servers.

From segment 1 to 8, the client detects server-side bottleneck every two segment downloads and consequently decides to increment the number of used servers (up to 6) until the targeted network throughput T_n is reached. At that point, the number of servers remains steady. In the mean time, as the buffer occupancy level increases and reaches a value higher than σ (25 sec), the amount of overhead data slowly decreases from 10% of the transmitted data to 0.5%.

Table 4.4 – Video evaluation dataset

Resolution	Bitrate	Resolution	Bitrate	Resolution	Redundant Bitrate
1920x1080	8Mbps	1280x720	4Mbps	1920x1080	250kbps
1920x1080	7Mbps	1280x720	3Mbps	1280x720	200kbps
1920x1080	6Mbps	720x480	2Mbps	720x480	100kbps
1280x720	5Mbps	720x480	1Mbps		

At segment 12, we purposely apply a download rate limitation on the client network interface that allows a maximum of 4 Mbps network throughput. As illustrated in Figure 4.7, a direct result of this client side bandwdth limitation is that the client runs the in-segment download adaptation rule (ii) because the amount of buffered content in seconds fell below the lower bound ε (6 sec) due to the suddenly low performing communication medium. Therefore, the MS-Stream client abandons the downloads of two sub-segments (from servers 2 and 5) before re-launching a request for a lighter sub-segment composed of 3 Mbps GoPs from server 2, as illustrated in Figure 4.7. Two segment downloads later (segment index 14), the client detects a client-side bottleneck and halves the number of used servers so as to avoid network congestion and prevent the chaotic behaviors of multiple connections competing for insufficient throughput on the same link.

Until segment 18, the client attempts to determine the reasons for the lack of network throughput that is necessary to obtain the 8 Mbps content quality. At segment 18, we remove the client download rate limitation, which increases the throughput observed at client side, enabling the detection of a server-side bottleneck. Hence, the increase in the number of used servers for the download of segment 19 , depicted in Figure 4.8. Then, the client keeps on attempting to detect the presence of a bottleneck and slowly increments the number of simultaneously considered servers to attain 6 at segment 24.

When the buffer level reaches 50 sec (175 seconds after the beginning of the experiment), we apply drastic throughput limitations on the network interfaces of 9 out of the 10 servers for the duration of three segment downloads. Except for server 3 that keeps its 2500 kbps throughput, all the other servers can send at most 50 kbps. Because of the 2-segment download observation window used for bottleneck detection, the client cannot determine the type of bottleneck that lead to the large drop in the observed throughput (Figure 4.9). Subsequently, the client does not increase the number of used servers and continues the streaming session without modifying the used servers. However, since the throughput from most of the servers

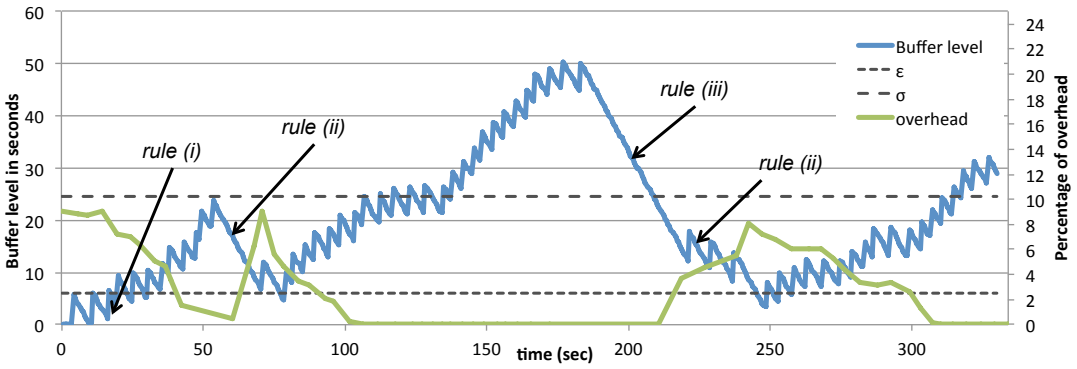

Figure 4.7 – Buffer Level and % of overhead

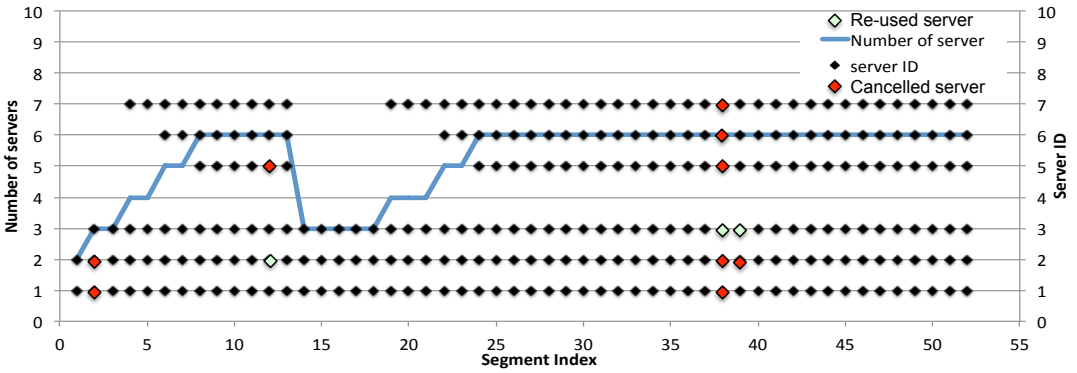

Figure 4.8 – Used servers

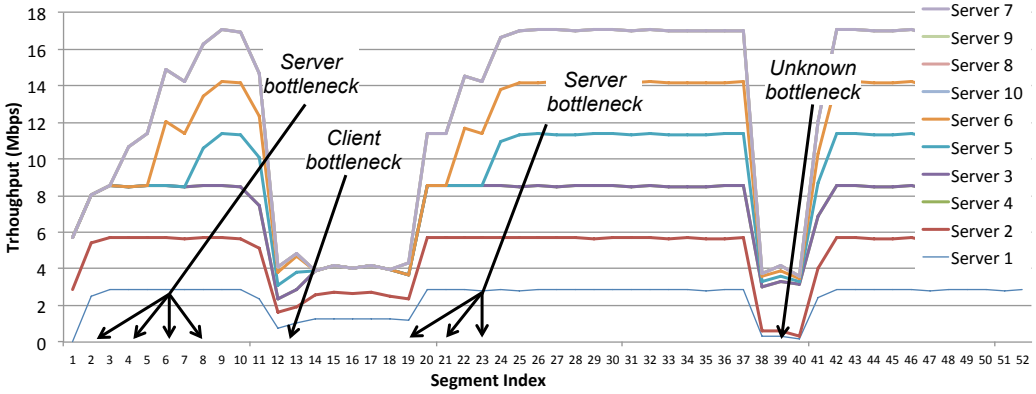

Figure 4.9 – Servers' throughput

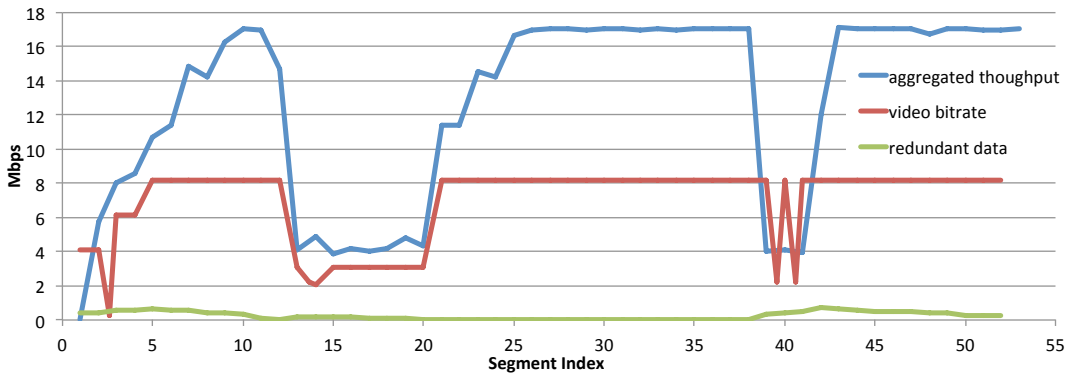

Figure 4.10 – Video and redundant data bitrates, and observed throughput

diminished, the client observes a slowdown of the delivery of sub-segments and a continuous decline of the buffered content level. Therefore, in an attempt to maintain the buffered content above the upper bound σ, the client first performs rules (iii) by cancelling all the sub-segment requests for segment 38 and re-assigning the delivery of the missing GoPs to the one server that has the highest available throughput. The second reaction of the client is to lower down the requested video quality to the 2Mbps video bitrate. The rule (ii) is also triggered for segment 39 because the delivery of the sub-segments from server 2 and 3 are estimated to terminate when the buffer level dives below ε. Thus, the delivery of the missing GoPs are re-assigned to server 3 with at lower bitrate in order to try sustaining the buffer level above ε. From segment 40 until the end of the streaming, throughput limitation is applied neither at the server side, nor at the client side.

4.2.2 Bottleneck estimation performance

Table 4.5 – Bottleneck estimation accuracy

Applied bottleneck	Estimated server bottleneck	Estimated client bottleneck	Estimated unknown bottleneck
Server	92.16%	3.72%	4.12%
Client	20.58%	65.47%	13.95%
None	0.01%	7.14%	92.85%

Regarding the bottleneck estimation method performances, we re-used the same test-bed as in section 4.2.1. We respectively set λ_{Low} and λ_{High} to 50% and 75% of the number of currently used servers. We successively applied and removed 300

client-side download bandwidth variations and 300 server-side upload throughput maximum capacity variations (applied on one or many servers at the same time, but not all of them together). The throughput limitations are uniformly distributed between 100 kbps and 1 Mbps. The number of selected servers which throughput is limited is uniformly selected between 1 and 10. We observe the bottleneck-type estimation made by the MS-Stream client. The results are exposed in Table 4.5. Our method performs well enough for the server-side bottleneck (in 92% of the cases). However, the client-side bottleneck estimation has room of improvement with near 66% of good estimation, and 20% of the cases where server bottlenecks are estimated instead of client ones. Future work will address the issue of variation of λ_{Low} and λ_{High} in time and their impact in the bottleneck estimation.

4.2.3 MS-Stream QoE evaluation for competing clients

In order to fully evaluate performance of MS-Stream with MATHIAS, we decided to set-up a real experiment over the Internet as if MS-Stream was deployed by an OTT live streaming provider on cheap hosting servers such as set-top-boxes and virtual private servers with heterogeneous resources. We deployed 10 MS-Stream servers as set-top-boxes in end-users' homes and universities from the european projects partners DELTA [2015] and DISEDAN [2016] in different locations in Europe: 2 set-top-boxes and 1 university server in Bordeaux, 3 set-top-boxes in Paris, 2 university servers in a laboratory in Heraklion, 1 set-top-box in Warsaw and 1 in Bucharest. The clients are located in Bordeaux.

For these experiments, we limit the maximum buffer level to 30 seconds to represent a real live streaming scenario. We compare MS-Stream + MATHIAS client (referred to as *MS* in the results) to a uni-source DASH solution with a high-end server. The DASH player is the dash.js player from the DASH-Industry Forum used for the evaluations of the previous chapter.

Table 4.6 summarizes the 6 evaluated streaming scenario. The *DASH_ 0*, respectively *MS_ 0*, scenario consists of a single client consuming content from a set of 1 server, respectively 10 servers. In both scenarios, the same total throughput of 10 Mbps is made avaiable to the client with the difference that in *DASH_ 0*, the one server has a 10 Mbps upload rate, whereas in the *MS_ 0*, each of the 10 servers has a 1 Mbps upload rate. In all other scenarios, i.e., *DASH_ 1*, *MS_ 1* *MS_ 2* and *MS_ 3*, two clients are concurently consuming content from the same server infrastructure. Similarly, in the *DASH_ 1* scenario, 20 Mbps are made available at the server side,

while only 2 Mbps are available at each of the 10 servers for the *MS_1 MS_2* and *MS_3* scenarios. Regarding the client, it is set at the University of Bordeaux behind an optical fiber connection with a 300 Mbps download capacity.

We empirically monitor the influence factors of QoE perceived at the consumer's side: mean displayed bitrate, average number of rebuffering events, average start-up delay, average number of quality changes. The resulting metrics are reported into Table 4.7. We also evaluate the quality distribution throughout the streaming session (Figure 4.11) to provide an overview of the different quality displayed to the end-users and their proportion in time. Finally, we monitor the average number of simultaneously used servers along with the average bandwidth overhead (both essential for the efficiency of MS-Stream). We repeat the video 50 times per application, representing a total playback time of 100 hours.

Table 4.6 – Evaluated applications

Eval ID	Available thr. (Mbps)	# of clients	# of servers	O_{max}	ε (sec)	σ (sec)
DASH_0	10	1	1	-	-	-
DASH_1	20	2	1	-	-	-
MS_0	10	1	10	10%	6	25
MS_1	20	2	10	10%	6	25
MS_2	20	2	10	10%	12	25
MS_3	20	2	10	15%	12	25

Table 4.7 – Evaluated applications and their QoE results (per 10 minutes)

Eval ID	Mean bitrate (Mbps)	Avg. quality changes	Avg. rebuff--erings	Avg. start-up delay	Avg. bw overhead	Avg. used servers
DASH_0	7.90	5.10	0	2.21 sec	0%	1
DASH_1	6.87	10.32	7.22	4.11s sec	0%	1
MS_0	7.97	3.21	0.01	1.58 sec	3.91%	9.52
MS_1	7.43	5.32	0.31	1.57 sec	4.37%	7.59
MS_2	7.78	4.11	0.29	1.56 sec	6.43%	7.68
MS_3	7.80	4.69	0.19	1.65 sec	9.12%	7.61

In the *DASH_0* and *MS_0* evaluations, the same amount of throughput is made available for one client. Interestingly, although both application obtain similar QoE

values, MS_0 has a lower start-up delay as the first segment is downloaded from several servers instead of one. In the evaluations *DASH_1* and *MS_1/2/3*, two clients are concurrently competing for the same network resources hence introducing bandwidth diversity and variability in the network. In *DASH_1*, the clients suffer from competition on the same resources. This resulted in lower mean bitrate and many rebuffering events for both clients (7.22 per 10 minutes). In contrast, the *MS_1/2/3* could adapt their usage of the network resources while competing on the same network paths and servers. Additionally, it is worth noticing that *MS_1/2/3* offer a more stable video experience to end-users as they expose a lot fewer quality switches compared to *DASH_1*. Also, the percentage of time spent at the highest quality is significantly higher in the MS-Stream applications than with DASH. We can observe that higher values of $\varepsilon, \sigma, t_{ii}$, and t_{iii} enable a lower amount of rebuffering, a higher mean bitrate, and a better quality distribution (in Figure 4.11). We can also observe that the actual percentage of overhead increases. Interestingly, the MS-Stream clients always minimize the overhead of data with less than 7% of overhead for *MS_1/2* and less than 10% for *MS_3*. We can conclude that a greater value of O_{max} permits the clients to avoid further rebuffering as more servers are allowed to deliver redundant data.

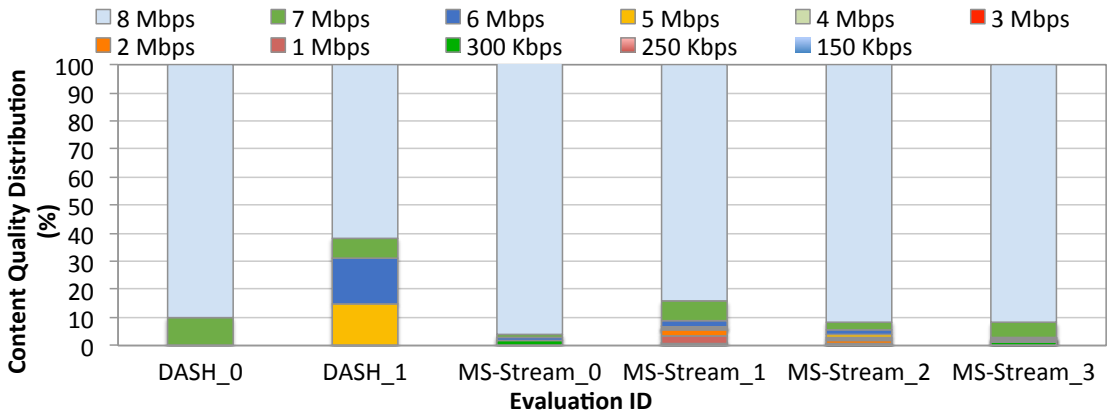

Figure 4.11 – Quality distribution throughout streaming sessions

4.3 Conclusion

In this chapter, we investigated the content delivery mechanisms involved in increasing the end-users perceived quality in the MS-Stream solution presented in Chapter 3. We proposed a set of algorithms called MATHIAS to tackle the chal-

lenges identified: bandwidth overhead consumption minimization; adaptation of the number of used servers; sub-segment scheduling and in-segment download adaptation. MATHIAS was found to improve the QoE of end-users and pragmatically use the resources made available to the MS-Stream clients.

Chapter 5

PMS: Quality and scale adaptive P2P/multi-server streaming

"MS-Stream is the future of video streaming"

— Daniel Négru

5.1 Introduction: Hybrid P2P/Multi-source Streaming

Although the usage of a multi-source streaming protocol has the potential to reach high end-users' QoE, the scale of the considered infrastructure (and consequently, its total upload throughput capacity) is limited. The P2P paradigm for video streaming permits to leverage the cooperation of peers, allowing to serve every video request with increased scalability and reduced costs. Hence, the contribution of each consuming client in transferring its downloaded segments to neighboring peers is a strong asset to further enhance the system's scalability at the best possible QoE.

We propose to combine the approaches detailed in Chapter 3 and 4 with the P2P paradigm in order to present a hybrid P2P/Multi-Server (*PMS*) solution for live streaming gathering scalability and quality adaptation capabilities. Indeed, we wish to benefit from the QoE and scalability potential of both P2P and multi-source streaming approaches. For PMS, we assume the scenario of a video streaming provider delivering a single video content available in Q_{max} different bitrates and consumed by a population composed of $N(t)$ peers. The provider's objective is to deliver the highest possible QoE by taking into account the heterogeneous and volatile

connectivity capacity of consuming peers as well as the limited upload throughput capacity of the fixed-size server infrastructure.

The PMS system and its architecture are presented along with the streaming signalling, the peer selection and the proposed P2P network impairment resiliency mechanisms. Peers are placed in mesh-based overlays, which are identified by the retrieved content quality that can be re-emitted to neighbors. Each peer requests part of the data composing the current video segment from the P2P application overlay it belongs to and the remaining data from the server infrastruture by relying on MS-Stream and MATHIAS. Additionally, we put forward a distributed and decentralized quality adaptation algorithm relying upon local and global indicators of the PMS functioning to control the transitions of peers from one overlay to another. This new quality adaptation strategy strives to enhance the end-users QoE while concurrently aiming at the successful functioning of all P2P overlays. To that end, the quality adaptation running at peer site takes into account the current network conditions, the capacity and efficiency of the P2P overlays, the peer's device resources and its contribution to the good functioning of overlays. Furthermore, we discuss the scalability limitations of our solution, and present a local optimization method circumventing them where every peer locally minimizes its utilization of the fixed-size server infrastructure without sacrificing the achieved QoE gains.

5.2 PMS system and architecture

We first present the hybrid streaming architecture before detailing the streaming signaling, the peer selection and the proposed P2P network impairment resiliency mechanisms.

5.2.1 Hybrid streaming architecture

As shown in Figure 5.1, the PMS system is composed of three major components: (**1**) the streaming portals including management services and trackers. The management services are in charge of delivering MPD files listing available servers hosting the content at the different qualities. As to the trackers, they act as rendez-vous points for all clients to obtain lists of candidate neighbors and to notify themselves as available resources for other peers. Additionally, the trackers are periodically computing global indicators on the current health of each overlay based on metrics reported by the peers. The latter indicators are forwarded to the peers and are

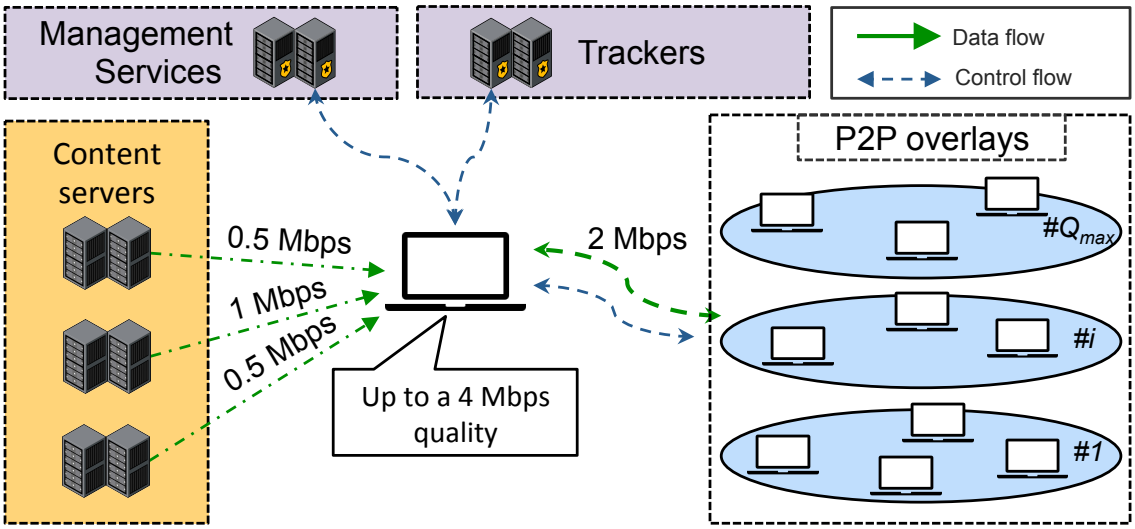

Figure 5.1 – PMS high level solution overview

then used in the decentralized quality and scalability adaptation mechanisms; (**2**) content servers that are dynamically provisioned with the live video flow at multiple qualities; (**3**) peers, which are organized in Q_{max} application-layer mesh-based overlays implementing a pull-based content consumption protocol, obtaining content from content servers and engaging in P2P data transfers. Within each distinct overlay i composed of $N_i(t)$ peers, the i-th quality (i.e., bitrate b_i) is actively being consumed and re-emitted by peers. At any given moment in time, a peer is part of one overlay only and can contribute to the streaming process up to its maximum upload throughput capacity. We rely on a multiple-overlay architecture in order to better understand and design the proposed quality and scalability adaptation algorithms. Ultimately, this multiple-overlay architecture does not overburden the classical design of application P2P networks.

In addition to the details of our system architecture, we also provide the functional software architecture of peers in Figure 5.2. The additional modules compared to the previously proposed MS-Stream client/server architecture (c.f. Figure 3.2 in section 3.2.1) are highlighted in plain blue. We adopt the HTTP protocol in both the signaling and the video data streaming from the content servers. The P2P communications are performed over the WebRTC technology that enables the contribution of peers behind NAT. The P2P data exchanges are performed over the WebRTC's data channel on top of the SCTP protocol that provides congestion/flow control along with reliability of message delivery.

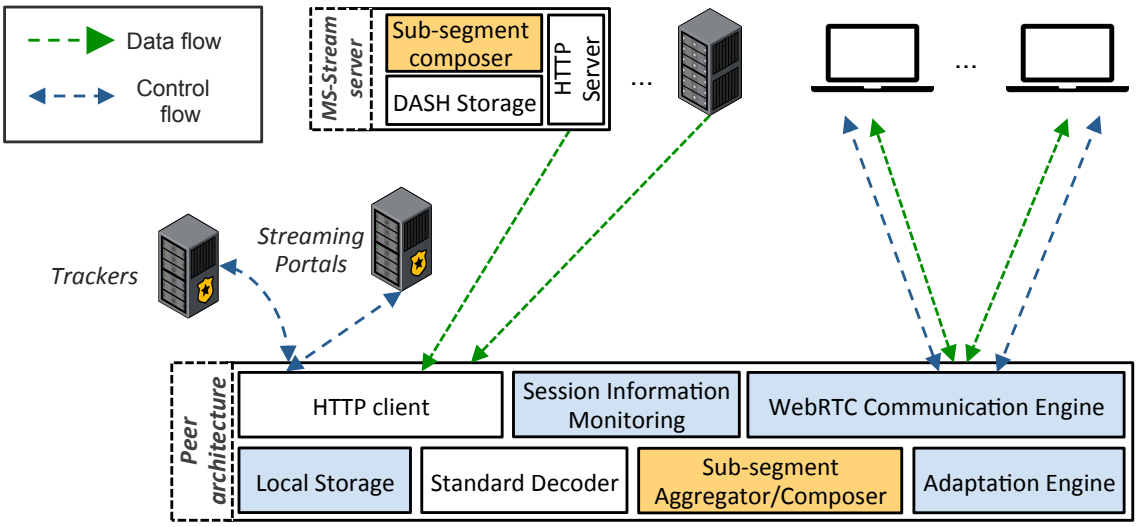

Figure 5.2 – Peer's software architecture

5.2.2 Streaming signaling

The signaling procedures between a new peer j, the servers, and the remote peers are depicted by the blue arrows in Figure 5.1. When a new client pops in, it proactively and periodically requests the Management Services for MPD files listing the nearest servers. The latter MPD file also contains a list of tracker URLs that the peer contacts to register its ID and to notify itself as an available resource for the overlay i it currently belongs to. The peer periodically requests the trackers to obtain a list of K peers in its overlay selected geographically close based on IP geolocation services. The peer regularly asks its K neghbors for two key information items used for the peer selection and for the adaptation algorithms controlling the quality and the usage of the server infrastructure: (1) their buffer maps listing the available segments and GoPs cached in the peers' local storage; and (2) their estimated upload rates $\alpha_{k,\Delta}(t)_{k\in[1..K]}$ computed by the peers themselves. Every peer j computes its upload rate $\alpha_{j,\Delta}(t)$ by taking the highest value between the mean throughput resulting from the number of packets transmitted to remote peers within the last Δ seconds and the highest delivery rate $rmax_{j,\Delta}(t)$ observed for one packet within Δ seconds. Moreover, the peer j repeatedly sends keep-alive messages to the trackers so as to maintain its visibility among the current overlay and simultaneously reports several metrics: its upload rate $\alpha_{j,\Delta}(t)$, the actual amount of data $a_{j,\Delta}(t)$ it delivered within the last Δ seconds, as well as the amount of data $\phi_{j,\Delta}(t)$ it received from the servers. In this way, the tracker follows the evolution

of each overlay's population $N_i(t)$ and computes global indicators on each overlay's functioning, further explained in section 5.3.

5.2.3 Fast-start

In order to minimize the initial video playback delay and enhance the end-user's QoE, without suffering from the delay induced by the communications with the tracker and the neighboring peers, the client relies on MS-Stream servers only for the first video segments to be downloaded. Regarding the very first video segment to be downloaded, the client requests two sub-segments with an equal number of GoPs at the bitrate b_1 (i.e., the lowest bitrate listed in the MPD file) and the other GoPs at the redundant bitrate b_r from two MS-Stream servers randomly selected in the list of servers listed in the MPD file. Once the P2P communciations are established, the peer simultaneously requests the servers to deliver a pre-determined percentage X_0 of the GoPs composing the video segments to be downloaded at the video bitrate b_i. The value of X_0 is an essential parameter that determines the system's scalability and its QoE capabilities. We discuss this parameter in section 5.3.

5.2.4 Peer selection

Periodically, every peer asks the trackers for a list of K neighbors in order then to obtain from them their buffer maps and upload rates. These neighboring peers are then requested to deliver the remaining percentage $1 - X_0$ of GoPs that are not delivered from the MS-Stream servers. Due to the high heterogeneity of peers' upload capacity, it is essential not to rapidly choke the low upload rate peers or under-utilize the ones with high upload rates. Choking low upload rate peers will lead to late request delivery for the neighboring peers using them, resulting in degraded QoE for many end-users and a handicap for the global video delivery system performance. Oppositely, leaving the high upload rate peers poorly used or inactive will induce a poor utilization of the available resources and greater operating costs supported at the server infrastructure level. Instead, every peer advocates for a fair usage of the available resources and accordingly locally optimizes the usage of neighboring peers' upload rate capacity for each segment delivery. Ideally, the greater the upload throughput of a neighboring peer k, the higher its chances to be selected for the delivery of GoP c. Consequently, the assignment process follows a discrete random variable and we set the probability $p_{c,k}$ of peer k to be assigned the

delivery of GoPs c as:

$$p_{c,k} = \frac{\delta_{c,k} \cdot \alpha_{k,\Delta}(t)}{\sum_{l=1}^{K} \delta_{c,l} \cdot \alpha_{l,\Delta}(t)} \tag{5.1}$$

where $\delta_{c,l} = 1$ if peer l has previously cached the chunk c in its local storage and $\delta_{c,l} = 0$ otherwise. As $\sum_{k=1}^{K} p_{c,k} = 1$, the assignment of each chunk c is ensured.

5.2.5 Resiliency to P2P network impairments and peer churn

During the delivery of sub-segments from remote peers, unexpected events can occur such as suddenly low performing communication channels or remote peers that become unavailable because they are no longer in the current overlay or because they disconnect from the streaming session. In PMS, the client consumption protocol incorporates the in-segment download adaptation rules of MS-Stream and extends this resiliency mechanism to P2P data transfer in order to avoid streaming session disruption. To that end, three additional client-side in-segment download adaptation rules have been designed for P2P data exchange only. The first two rules permit to handover the delivery of GoPs to the MS-Stream servers due to low-performing remote peers, and to avoid a too fast buffer depletion that could eventually lead to video glitches. As to the third rule, it allows to display a temporarily low visual quality by cancelling the sub-segment from the low-performing peers and relying on the redundant GoPs at the redundant bitrate delivered by the MS-Stream servers. The three rules consist in:

(1) If a utilized remote peer becomes unavailable or if the available amount of buffered content $bufLevel(t)$ falls below a given threshold t_1, then the peer estimates the remaining duration of the sub-segment delivery from this peer. $bufLevel(t)$ denotes the buffer level occupancy in seconds at time t. If the estimated remaining duration exceeds another threshold t'_1 (with $t_1 > t'_1$) then the consuming client cancel this download. In the mean time, the client re-assigns the not yet delivered GoPs from this peer to the servers at the currently selected video quality. The value of t'_1 defines the limit under which the remaining playback time is considered insufficient to ensure smooth video with regards to the estimated arrival time of sub-segment requests.

(2) Let us consider a second threshold $t_2 < t'_1$. If a utilized remote peer suddenly becomes unavailable or if buffered content $bufLevel(t)$ falls below a given threshold t_2, then the client cancels the ongoing sub-segment downloads from the

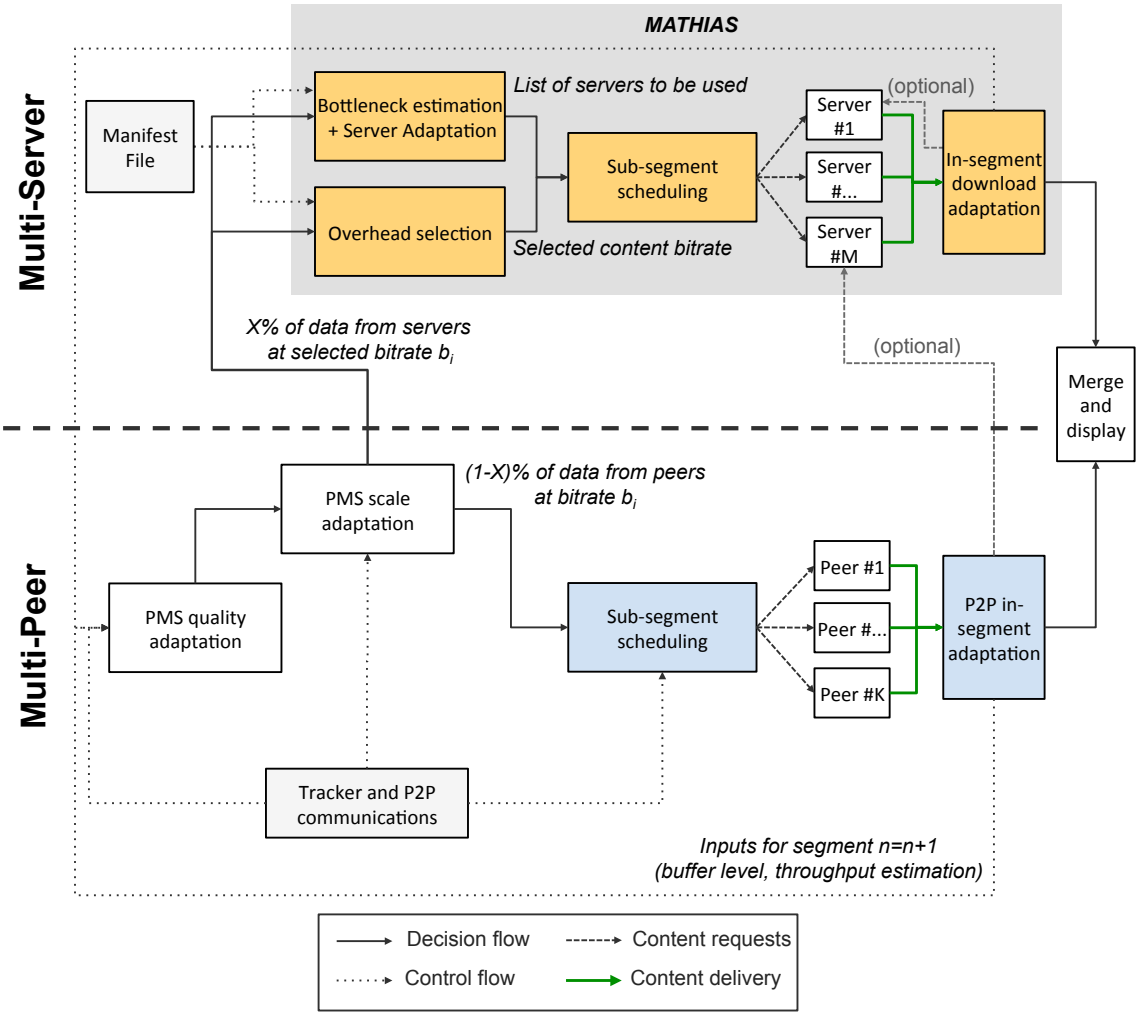

Figure 5.3 – PMS consumption decisioning for quality and scale adaptation

remote peers and launches a new request for the missing GoPs to a MS-Stream server at the redundant bitrate.

(3) Let us consider a third threshold $t_3 < t_2 < t'_1$. If $bufLevel(t) < t_3$, and if the client has already received some GoPs at the redundant bitrate that can replace the missing GoPs from the current sub-segment request at the desired bitrate, then the client abandons the current P2P downloads, merges the available sub-segments and displays content to the end-user. This last resort reactive rule attempts to avoid interrupting the video playback by displaying a temporary sub-optimal visual quality to the end-users. In the event of the above-described server fallback mechanisms, a greater percentage of data is being consumed from the servers.

5.3 The proposed quality and scale adaptation algorithm for PMS

PMS is an extension of MS-Stream that combines the use of peers and servers together to increase both QoE gains and system's scale. Figure 5.3 exposes the consumption and adaptation decision flow chart of a PMS client. For every segment download, the PMS client first runs the *PMS quality adaptation* decision to determine whether to stay in the current overlay or to migrate to another overlay. Second, the *PMS scale adaptation* algorithm locally optimizes the ratio of GoPs (hence the required network throughput) that will be requested from the servers and from neighboring peers. Once both PMS adaptation algorithms are executed, the MATHIAS algorithms decide on how to use the available servers to reach the required network throughput while the PMS client engages in P2P communications and retrieves GoPs from its neighbors. Finally, during the download of GoPs from servers and peers, the PMS client runs the MATHIAS and PMS in-segment download adaptation rules to attempt avoiding any video rebufferring events.

5.3.1 The proposed P2P/multi-source quality adaptation algorithm

In most of the current adaptive streaming solutions over HTTP, the client-centric decisions to adjust the desired quality are based on observable local parameters (observed throughput, buffer depletion speed, etc.). Hence, the client selfishly attempts to maximize the end-user's QoE by consuming the network resources made available for its streaming session. Considering the heterogeneous and highly unreliable nature of the P2P components in the PMS system, consuming peers can no longer perform quality adaptation based on local parameters only. Indeed, such consumption behaviors could lead to resources starvation (namely the upload throughput) first in the P2P overlays and second in the server infrastructure. We propose a distributed quality adaptation algorithm running at peer site, where each peer has to be fair to all the others by moving upwards or downwards in the overlay system according to both local and global indicators on the system functioning. Table 5.1 references the indicators and variables used in the PMS system for quality adaptation.

Two global indicators concluding on the system's health are computed by the trackers based on the metrics reported by the peers within the last τ seconds: the overlay's *capacity index* [Wu *et al.*, 2009] $\kappa_{i,\tau}(t)$ to deliver the consumed quality at

bitrate b_i and the *system delivery efficiency* $\eta_{i,\tau}(t)$ for each content quality i. The two global indicators are computed as follows:

$$\kappa_{i,\tau}(t) = \frac{\sum_{j=1}^{N(t)} \gamma_{i,j}(t) \cdot \alpha_{j,\tau}(t)}{N_i(t) \cdot b_i} \tag{5.2}$$

$$\eta_{i,\tau}(t) = \frac{\sum_{j=1}^{N(t)} \gamma_{i,j}(t) \cdot (a_{j,\tau}(t) + \phi_{j,\tau}(t))}{N_i(t) \cdot b_i \cdot \tau} \tag{5.3}$$

with $\gamma_{i,j}(t) = 1$ if peer j is in overlay i at time t, and $\gamma_{i,j}(t) = 0$ otherwise. The overlay's capacity index is a high level indicator of the overlay's capacity to deliver content at bitrate b_i to its demanding peers. It represents the ratio of the achievable upload throughput of the peers composing the overlay to the global throughput demand of the peers for the quality consumed. When $\kappa_{i,\tau}(t)$ is greater than $1 - X_0$, the overlay i is supposedly capable of providing enough throughput for all its peers. If $\kappa_{i,\tau}(t)$ falls below $1 - X_0$, the overlay attains a critical regime.

The delivery efficiency $\eta_{i,\tau}(t)$ is a more precise indicator of the entire system's functioning when compared to the overlay's capacity. Indeed, it represents the ratio of the delivered data from both remote peers and servers to the demanded data, computed every τ seconds, for the last τ seconds. The overlay's capacity shows the potential of the overlay to sustain a global throughput to the peers composing it, whereas the delivery efficiency captures the state of the video quality delivery within the overlay. The closer $\eta_{i,\tau}(t)$ is to 1, the better the delivery of quality i is carried out by the entire system.

Moreover, every peer j monitors respectively the estimated download throughput $\lambda_{j,n}^{P2P}$ and $\lambda_{j,n}^{server}$ by adding up the estimated throughputs on each peer-to-peer and client-server communication channels. These local indicators reflect the P2P streaming system's and the server system's throughput performance according to the peer's point of view and are respectively referred to as *local P2P throughput* and *local server throughput*.

The quality adaptation process allows the peers to move from one overlay to an adjacent one only, aiming at minimizing the negative effects of high quality variation amplitudes on QoE, as reported by Yitong *et al.* [2013a]. During the download of segment n, the peer j located in the overlay i retrieves the global indicators $\eta_{i,\tau}(t)$, $\kappa_{i,\tau}(t)$, $\eta_{i+1,\tau}(t)$ and $\kappa_{i+1,\tau}(t)$ of both the current and the above overlays. At the end of the download of every segment n, the peer runs the quality adaptation process detailed in algorithm 3.

Table 5.1 – Variables used for quality adaptation in PMS

Symbol	Description
$\lambda_{j,n}^{server}$	*Local server throughput:* Network throughput from the server infrastructure estimated by peer j
$\lambda_{j,n}^{P2P}$	*Local P2P throughout:* Network throughput from the neighboring peers estimated by peer j
$\kappa_{i,\tau}(t)$	Overlay capacity index to deliver the consumed quality at bitrate b_i during τ
$\eta_{i,\tau}(t)$	Actual delivery efficiency of quality i during τ
η_{thres}	Efficiency threshold
$\alpha_{j,\Delta}(t)$	Upload rate computed by peer j during Δ
$rmax_{j,\Delta}(t)$	Highest delivery rate observed from one packet during Δ
$a_{j,\Delta}(t)$	Data delivered by peer j to remote peers during Δ
$\phi_{j,\Delta}(t)$	Data delivered from the server infrastructure to peer j during Δ
X_0	Percentage of data initially requested by each peer from the server infrastructure for every video segment

Algorithm 3 leads the peer's movements among overlays by having it preserve the overlays' capacity and the delivery efficiency of each quality according to the pre-defined and enforced utilization of the server infrastructure (i.e. X_0). Downward movements in the overlay architecture are aggressively influenced by local indicators. Indeed, such indicators provide the peer with a local view of the P2P system's behavior toward its streaming session, and let it know whether it should keep or downgrade the requested quality in order to avoid buffer starvation and to maintain satisfying QoE. Alternatively, upward-movement decisions are conservatively influenced by both the peer's local indicators and the global indicators of the current and target overlays. The global indicators inform the peer whether an upgrade decision in the requested quality is harmless for the functioning of the current and target overlays, while the local indicators permit the peer to understand whether the used MS-Stream content servers can sustain the required throughput for the delivery of the quality $i + 1$.

First, the peer verifies whether its local view of P2P system's throughput $\lambda_{j,n}^{P2P}$ and its local view of server system's throughput capacity $\lambda_{j,n}^{server}$ can respectively sustain its demand for P2P throughput in the current overlay and its demand for server throughput (line 1 of algorithm 3). In this case, the peer may have reached either its maximum download capacity or the maximum throughput capacity of

server and neighboring peers. Consequently, the peer is allowed to stay within the current overlay i.

Algorithm 3 PMS content quality adaptation

Inputs: $\lambda_{j,n}^{P2P}, O_{max}, \lambda_{j,n}^{server}, \alpha_{j,\Delta}(t), \eta_{i+1,\tau}(t), \eta_{thres}, \kappa_{i,\tau}(t), \kappa_{i+1,\tau}(t), \delta_{i,max}, \delta_{i+1,max}$

Output: overlay migration decision

1: **if** $(1 - X_0) \cdot b_i < \lambda_{j,n}^{P2P} \leq (1 - X_0) \cdot b_{i+1}$ **and** $X_0 \cdot b_i + \delta_{i,max} < \lambda_{j,n}^{server} \leq X_0 \cdot b_{i+1} + \delta_{i+1,max}$ **then**
2: Stay in overlay i
3: **else if** $\lambda_{j,n}^{P2P} > (1 - X_0) \cdot b_{i+1}$ **or** $\lambda_{j,n}^{server} \geq X_0 \cdot b_{i+1} + \delta_{i+1,max}$ **then**
4: **if** $\kappa_{i,\tau}(t) \leq 1 - X_0$ **and** $\alpha_{j,\Delta}(t) \geq (1 - X_0) \cdot b_i$ **then**
5: Stay in overlay i
6: **else if** $(\alpha_{j,\Delta}(t) \geq (1-X_0) \cdot b_{i+1})$ **or** $(\kappa_{i+1,\tau}(t) > 1-X_0$ **and** $\eta_{i+1,\tau}(t) \geq \eta_{thres})$ **then**
7: Upgrade to overlay $i + 1$
8: **else**
9: Stay in overlay i
10: **end if**
11: **else**
12: Downgrade to overlay $i - 1$
13: **end if**

In algorithm 3, the demand for server throughput is $(X_0 \cdot b_i + \delta_{i,max})$, where $\delta_{i,max}$ is the maximum allowed extra throughput that can be utilized for bitrate b_i:

$$\delta_{i,max} = \frac{O_{max} \cdot b_i}{1 - O_{max}} \tag{5.4}$$

In the event where the local P2P throughput or the local server throughput are high enough for the peer to reach a better video quality (line 3), the peer first checks whether it can move to the upper overlay safely without harming the overlays' health and their global indicators. Although the peer may have the download capacities to move upward, algorithm 3 enforces the peer to remain at its current quality i if the current overlay's capacity is critically low (i.e.,$\kappa_{i,\tau}(t) \leq 1 - X_0$) and if the peer's upload capacity contributes significantly to the overlay (i.e., $\alpha_{j,\Delta}(t) \geq (1 - X_0) \cdot b_i$ at line 4). If not, if the peer can significantly contribute to P2P transfer or if the overlay's capacity is high enough and the $i + 1$ quality delivery efficiency is greater than a threshold η_{thres} (line 6), then the peer joins the upper overlay Finally, when the requested quality can neither be maintained nor upgraded, the peer moves down to overlay $i - 1$ (line 12).

5.3.2 Increasing scalability without ceding on QoE: A local optimization method

The incentive in trading off the obtained consumers' QoE for a higher system scale is led from the streaming providers' objective to serve as many end-users as possible. This multi-criteria optimization problem is to be solved in order to satisfy both the providers and its clients. However, the great demand on streaming QoE from end-users is also a major constraint that the provider must target for the sake of its streaming platform popularity. To this end, we propose a scale adaptation algorithm running at peer site and locally optimizing the value of X_0, now referred to as $X_{j,n+1}$ (i.e.: the percentage of GoPs $X_{j,n+1}$ that is to be requested from servers for the next video segment $n+1$).

The value of X_0 is an essential parameter that can determine the system's scalability and its QoE capabilities. It globally represents the rate at which data are transferred from the MS-Stream servers to the overlays for every new segment download. Although a great value of X_0 could give a potentially high end-user's QoE due to the stability and efficiency of the server system; it could also lead to an overuse of its upload capacity, severely reducing the entire system's scalability and under-utilizing the P2P system. On the other hand, while a low value of X_0 would highly offload the server system and permit a high streaming system scalability, the P2P system could suffer from starvation due to insufficient data availability in the overlays, leading to video disruption events or low delivered video quality. The objective of the local optimization method is first to preserve the QoE level obtained when using the static value of X_0, and second to minimize utilization of the server system to increase the entire system's scalability (under the constraint that $X_{min} < X_{j,n+1} < X_0$ with X_{min} being the minimum allowed value).

The scale adaptation method is detailed in algorithm 4. The rationale behind this algorithm is to have every peer slowly moving toward an increase in the P2P system utilization when the P2P system provides sufficient resources to allow it. In doing so, the scale adaptation algorithm reduces the likelyhood of peer suddenly loosing on QoE (with video stalls) if the P2P system is not stable. The peers observe the evolution of the P2P system functioning by retrieving a new global indicator ,termed *global P2P system efficiency*, from the trackers and by computing a *local P2P capacity* indicator at the end of every segment download.

Deriving from the above-mentioned system efficiency $\eta_{i,\tau}(t)$, we define the *global P2P system efficiency* $\eta_{i,\tau}^{P2P}(t)$. $\eta_{i,\tau}^{P2P}(t)$ is the ratio of the amount of data delivered

by the peers composing the overlay i during the last τ seconds, to the amount of data demanded by the latter peers in this period. This indocators depicts the efficiency of the P2P overlay i to deliver data to its population.

$$\eta_{i,\tau}^{P2P}(t) = \frac{\sum_{j=1}^{N(t)} \gamma_{i,j}(t) \cdot a_{j,\tau}(t)}{N_i(t) \cdot b_i \cdot \tau} \tag{5.5}$$

We further define the *local P2P capacity* indicator $\mu_{j,n}^{P2P}$, that reflects the peer's view of the P2P system capacity. $\mu_{j,n}^{P2P}$ is the minimum value between 1 and the ratio of the local P2P throughput observed by peer j for the download of segment n to the requested video bitrate b_i. It permits to locally capture the extent to which the P2P system is delivering enough throughput to the peer for the current bitrate b_i. When the value of $\mu_{j,n}^{P2P}$ is close to 1, it shows that the utilized neighbors can actually stream the video segment almost entirely in a real-time manner. On the contrary when the value of $\mu_{j,n}^{P2P}$ collapses, the neighboring peers can hardly re-emit the chunks they downloaded at sufficient throughput.

$$\mu_{j,n}^{P2P} = min \left[1; \frac{\lambda_{j,n}^{P2P}}{b_i} \right] \tag{5.6}$$

Table 5.2 references the indicators and variables used in the PMS system in the scale adaptation algorithm.

Table 5.2 – Variables used for scale adaptation in PMS

Symbol	Description
$\eta_{i,\tau}^{P2P}(t)$	*Global P2P efficiency:* Delivery efficiency of the P2P overlay i during τ
$\mu_{j,n}^{P2P}$	*Local P2P capacity:* Delivery capacity of the P2P overlay i during the download of segment n observed by peer j
$X_{j,n}$	Adjusted value of X_0 for every new video segment n to be downloaded $X_{j,n} \geq X_{min}$
X_{min}	Minimum value that $X_{j,n}$ can take

In algorithm 4, the peer first compares its local view of the P2P system capacity to the global P2P system efficiency handed out by the trackers. When the local P2P capacity is lower than the global P2P efficiency (line 2 of algorithm 4, the peer can understand that its neighbors could not even provide throughput comparable to what is delivered to the other peers constituting the overlay. Therefore, as preserving

QoE prevails over the system's scalability, the peer adapts $X_{j,n+1}$ to the local P2P capacity (line 3). In the event where the local P2P capacity is greater than the global P2P efficiency, the peer strives to maintain its demand higher than what the P2P system delivered during the last τ seconds. However, should the observed local P2P capacity be lower than the initially demanded video data ratio from the P2P system (i.e.: $1 - X_{j,n+1}$ line 5), then the peer decreases its request for P2P throughput, but maintain it higher than the global P2P efficiency (line 6). This is done in order to continue benefiting from the fact that the local P2P capacity exceeds the global P2P efficiency. Finally, when the local P2P capacity indicates that the obtained throughput from neighbors exceeds even the demanded data ratio from the P2P system, then the peer proceeds to carefully increase its P2P transfer demands with regards to the maximum value between the last $1 - X_{j,n}$ and the average between the local P2P capacity and the global P2P efficiency (line 8).

When the peer moves upward or downward in the overlay architecture, the value of X_n, j is automatically set to $\eta_{i,\tau}^{P2P}(t)$.

Algorithm 4 PMS scale adaptation

Inputs: $\eta_{i,\tau}^{P2P}(t), X_{j,n}, \lambda_{j,n}^{P2P}$
Output: $X_{j,n+1}$

1: $\mu_{j,n}^{P2P} = min\big[1; \frac{\lambda_{j,n}^{P2P}}{b_i}\big]$
2: **if** $\mu_{j,n}^{P2P} \leq \eta_{i,\tau}^{P2P}(t)$ **then**
3: $\quad X_{j,n+1} = 1 - \mu_{j,n}^{P2P}$
4: **else**
5: $\quad$ **if** $\mu_{j,n}^{P2P} \leq 1 - X_{j,n}$ **then**
6: $\quad\quad X_{j,n+1} = 1 - \frac{\mu_{j,n}^{P2P} + \eta_{i,\tau}^{P2P}(t)}{2}$
7: $\quad$ **else**
8: $\quad\quad X_{j,n+1} = max[X_{j,n}; 1 - \frac{\mu_{j,n}^{P2P} + \eta_{i,\tau}^{P2P}(t)}{2}]$
9: $\quad$ **end if**
10: **end if**

5.4 Evaluation

5.4.1 Test bed set-up

The PMS streaming system was implemented and deployed in France for evaluation purposes over the Internet. For our experiments, we instantiated fifteen cloud

Table 5.3 – Internet testbed with emulated rate limitations

Emulated limitation	Up/Download capacities	# of peers	Ratio of peers
aDSL	800Kps/8Mbps	105	35%
vDSL	8Mbps/20Mbps	60	20%
Fiber1	5Mbps/20Mbps	60	20%
Fiber2	20Mbps/40Mbps	30	10%
Cable	5Mbps/20Mbps	45	15%

servers with 25Mbps of upload throughput, representing a total throughput capacity of 375 Mbps. Our management services and tackers were hosted in the cloud as well. We deployed 300 peers in end-user homes located in different cities in France (Paris, Bordeaux, Poitiers, Versailles) and in the LaBRI laboratory in Bordeaux. Peers were running in docker containers, which allowed us to set bandwidth limitations on the docker network interfaces to emulate isolated access networks and prevent almost unlimited bandwidth between peers under the same roof. Table 5.3 details our Internet testbed and the emulated bandwidth limitations. The Big Buck Bunny video (with 10-second segments, each containing 20 GoPs) was encoded over 4 bitrates (1, 2, 4 and 6Mbps) at a 720p spatial resolution. The redundant bitrate was set to 200kbps. The pulsation τ value of the tracker is set to 10 seconds and the trackers compute the global indicators every 10 seconds. Peers request the trackers for a list of $K = 15$ peers. The threshold governing the PMS resiliency mechanisms are $t_1 = 15secs, t_1' = 12secs, t_2 = 3secs$ and $t_3 = 1sec$. The efficiency threshold used in the bitrate adaptation method is conservatively set to $\eta_{thres} = 0.90$. The client buffer size was limited to 30 seconds (i.e., 3 video segments) for the live streaming.

As to the MATHIAS parameters, the maximum allowed bandwidth consumption is set to $O_{max} = 10\%$ when the buffer occupancy exceeds $\sigma = 25$ sec, and its miminum O_{min} is set to 0% when the buffer level falls below $\varepsilon = 6$ sec. We set the resiliency rules's thresholds $t_{ii} = \varepsilon$ and $t_{iii} = \sigma$. Regarding the threshold involved in the bottleneck estimation method, the normalized throughput difference θ determining whether the throughput of a given server varies highly in time is 30%. The lower and upper bound of the number of servers which delivered throughput varies highly are $\lambda_{low} = 25\%$ and $\lambda_{high} = 75\%$.

5.4.2 Evaluated streaming applications

We propose to gradually modify the level of adaptability among the evaluated streaming applications in order to clearly identify the contributions of each proposal: the hybrid P2P/Multi-Server consumption protocol, the distributed content quality adaptation algorithm, and the scale adaptation method. Table 5.4 summarizes the evaluated applications.

First, the *P2P/Multi-Server Non-Quality-Adaptive and Non-Scalability-adaptive* (PMS-NQNS) application is the most basic hybrid P2P/Multi-Source streaming solution that offers the least level of adaptability. PMS-NQNS only implements the streaming protocol detailed in section 5.2 without using the proposed resiliency mechanisms to P2P network impairments and without using redundant data into the sub-segments requested to the servers. Naturally, PMS-NQNS utilizes the MATHIAS algorithms to consume content from the servers. A fixed-value of server-infrastructure usage X_0 is enforced for the entire streaming session.

For all other streaming applications, the MS-Stream solution with redundant GoPs and with the MATHIAS algorithms are enabled. The second evaluated application is *P2P/Multi-Server Adaptive-Quality and Non-Scalability-adaptive* (PMS-AQNS) that implements the entire consumption protocol of PMS (section 5.2), the MS-Stream solution with redundant GoPs, as well as the proposed content quality adaptation. In PMS-AQNS, MATHIAS is employed for the utilization of the server infrastructure and a fixed value of X_0 is enforced. PMS-AQNS permits to directly observe the effects of the proposed quality adaptation algorithm on the system compared to PMS-NQNS.

The *P2P/Multi-Server Non-Quality and Adaptive-Scalability* (PMS-NQAS) streaming solution enables the use of the scale adaptation method run at peer site without the distributed quality adaptation. Therefore, each peer can only consume the 6 Mbps content quality while locally striving to reduce its utilization of the server infrastructure (variable value of $X_{j,n}$ in time, $X_{j,n} < X_0$) and to increase the global P2P efficiency. In doing so, we can exhibit the impact of scale adaptation on the system and on the delivered QoE. The complete PMS streaming solution is exposed with the *P2P/Multi-Server Adaptive-Quality Adaptive-Scalability* (PMS-AQAS) clients.

We compared these PMS clients to a recent proposal for quality adaptive streaming in P2P system (based on DASH) [Merani and Natali, 2016] that utilizes P2P data transfers to achieve high streaming performance but does not rely on a hybrid P2P/CDN architecture. The approach proposed by Merani and Natali [2016] also

Table 5.4 – Evaluated streaming applications

Protocol	Consumption and adaptation characteristics
PMS-NQNS	Non-Quality-adaptive, and Non-Scalability-adaptive: implements only the streaming protocol detailed in section 5.2 without using the proposed resiliency mechanisms, without using the redundant GoPs in the requested sub-segments, and using a fixed value of X_0 for the entire streaming
PMS-AQNS	Adaptive-Quality and Non-Scalability-adaptive: implements the entire hybrid protocol of section 5.2 with a fixed value of X_0 and uses the quality adaptation algorithm of section 5.3.1
PMS-NQAS	Non-Quality-adaptive and Scalability-Adaptive: implements the entire hybrid protocol of section 5.2 and uses the scalability adaptation of section 5.3.2
PMS-AQAS	Adaptive-Quality Scalability-Adaptive: AQNS + NQAS
P2P-DASH [Merani and Natali, 2016]	P2P-DASH: implements the distributed adaptation algorithm proposed by Merani and Natali [2016], wherein the focus is on the P2P streaming more than on the usage of the server platform (the server is used at 4 times the rate of the consumed video)

considers global indicators of the P2P system functioning in order to conduct a distributed content quality adaptation among peers. In their evaluations, the authors use an Integer Linear Programming model and define the end-users' satisfaction function as a global parameter that they attempt to maximize. The satisfaction function is computed as the sum of peers that can reach a target video bitrate. The target video bitrate may be different from one peer to another and is a function of the upload and download capacities of the peer's device, known before the streaming session starts. Finally, the influence factors of QoE (average bitrate, video freezes, quality changes) are not evaluated from the end-users' perspective. Therefore, we re-implemented the approach detailed in [Merani and Natali, 2016] in order to compare our work to their approach on QoE influence factors and on scalability potential.

5.4.3 Experiments

Our primary goal was to first study the trade-offs between scalability and QoE of PMS, and then to determine the feasibility and efficiency of our proposed quality and scale adaptation methods. To that end, each experiment comprises the use of one streaming application from Table 5.4 where every peer of the system has to use the same initial value of X_0. Then, according to the evaluated streaming

application, the value of X_0 can be modified by the peers (i.e., in the PMS-NQAS and PMS-AQAS cases). For each experiment, we varied the value of X_0 in an step-wise manner from 5% to 100% with steps of 5%. It is worth noting that when $X_0 = 100\%$, the MS-Stream/MATHIAS scenario is played out.

Each experiment lasted 30 minutes. During the first 15 minutes of the runs, the peer population was limited to half the pool of deployed clients (i.e., 150). The duration of the streaming sessions followed a log-normal distribution of a mean the duration of our video. When a peer finishes its streaming session, it directly starts a new one. In order to study the impact of flash crowd event on the delivered QoE and on the system scalability, 150 other peers joined in the streaming at the 15^{th} minute.

We evaluated our PMS protocols on the same studied set of QoE criteria as in Chapter 3 and Chapter 4, before the flash crowd event and after the flash crowd event: quality distribution during the peer's streaming session (Figure 5.4 and 5.5), the average number of rebuffering events (Figure 5.6 and 5.7) and the average start-up delay, the mean displayed bitrate for each peer (Figure 5.8 and 5.9), the average number of quality changes per peer (Figure 5.10 and Figure 5.11). In Figure 5.12 and 5.13, we can observe the difference between the initially selected value of X_0 and the actual ratio of data consumed by the clients from the server infrastructure. For each evaluated metrics, we also plot its standard deviation in all figures.

5.4.4 Results

We begin by commenting on the results before the flash crowd event.

Rebuffering per session and average start-up delay

For all PMS applications, the average start-up delay was less than 3 seconds. In contrast, the average start-up delay of P2P-DASH clients exceeded 20 seconds as the most of the peers were required to wait for a subset of them to be able to initiate the P2P data exchanges.

In the original video, there are 6 segments displayed per minute. For NQNS, the number of video stalls per minute is almost at its maximum (i.e., 6) when X_0 ranges from 5% to 35% and from 50% to 100%. On the one hand, when the value of X_0 ranges from 0 to 35%, the overlay of peers does not receive enough data from the servers to sustain the demand of P2P data transfer for the consumption of real-time data, which results in the high number of video stalls per minute. On the other hand, when the value of X_0 is greater than 50%, the utilization of the server

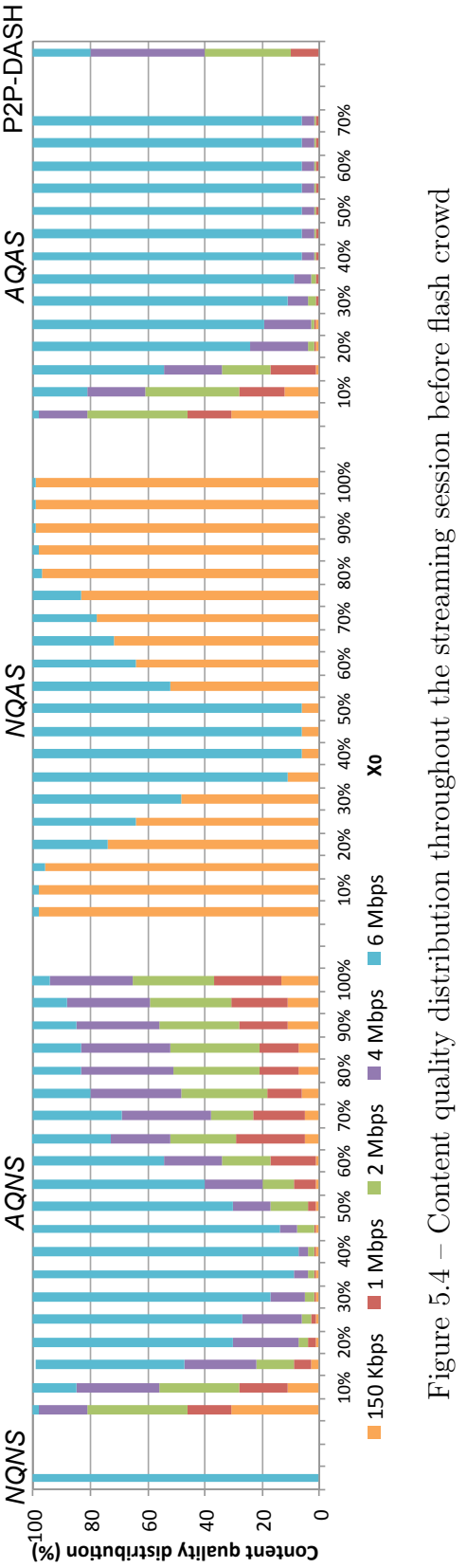

Figure 5.4 – Content quality distribution throughout the streaming session before flash crowd

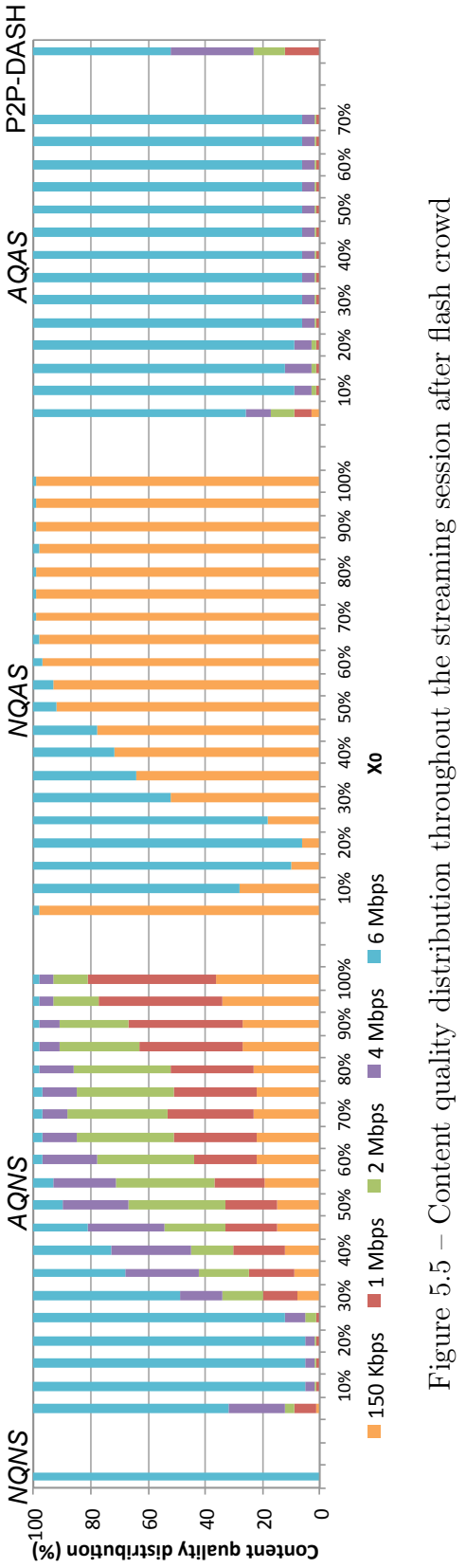

Figure 5.5 – Content quality distribution throughout the streaming session after flash crowd

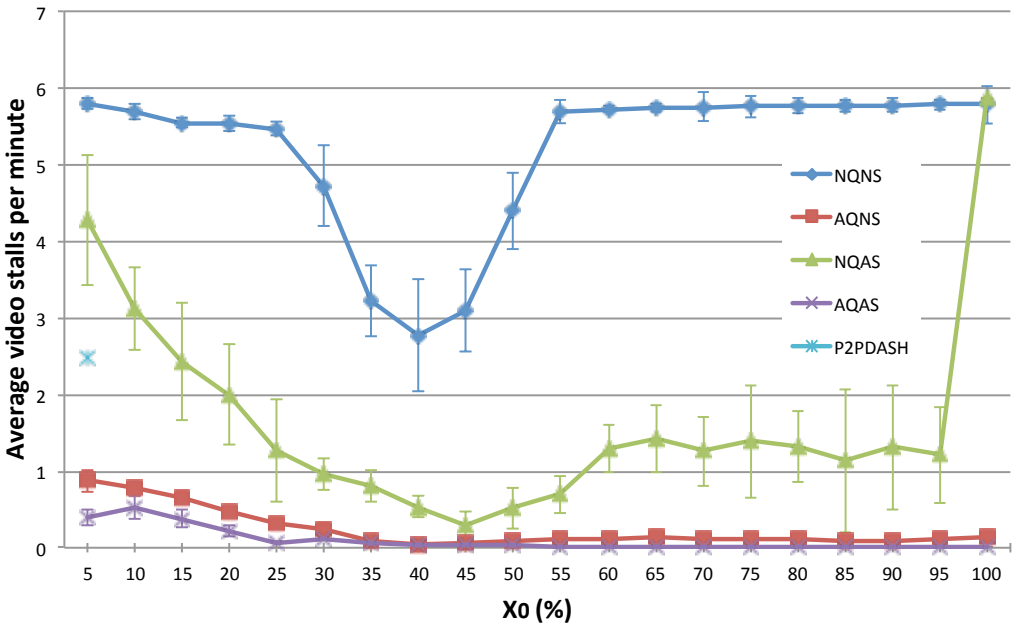

Figure 5.6 – Average video rebuffering per minute before flash crowd

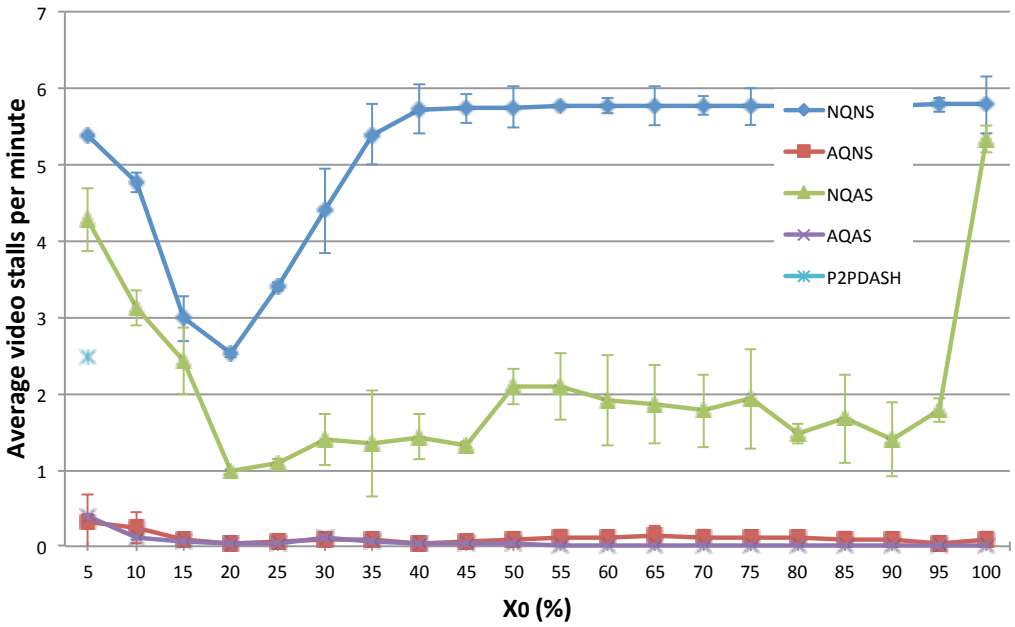

Figure 5.7 – Average video rebuffering per minute after flash crowd

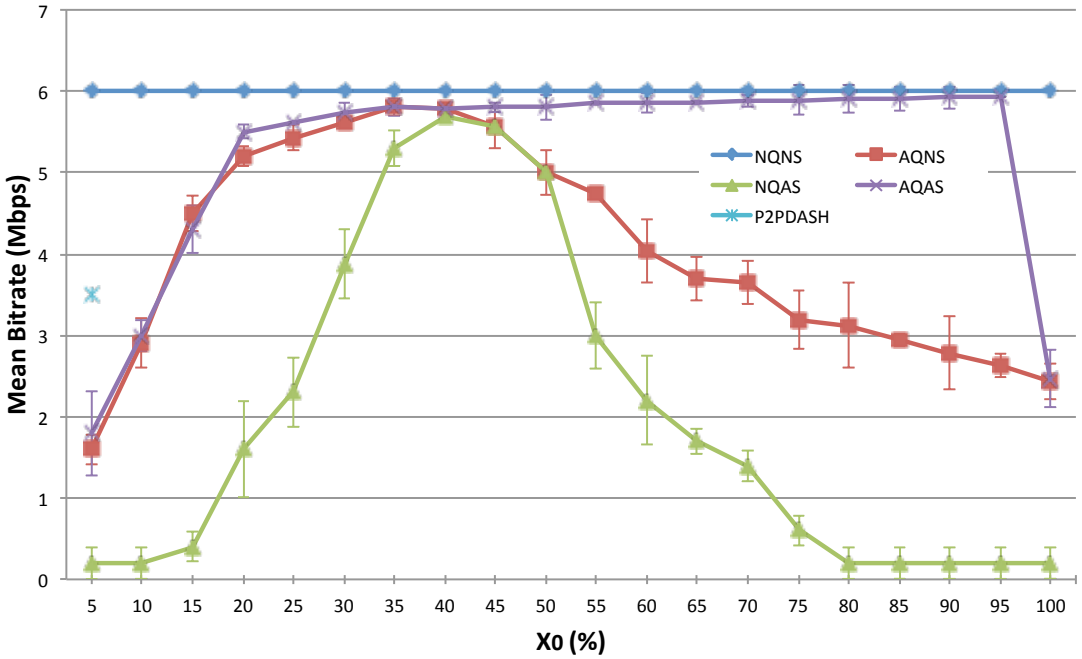

Figure 5.8 – Average mean bitrate before flash crowd

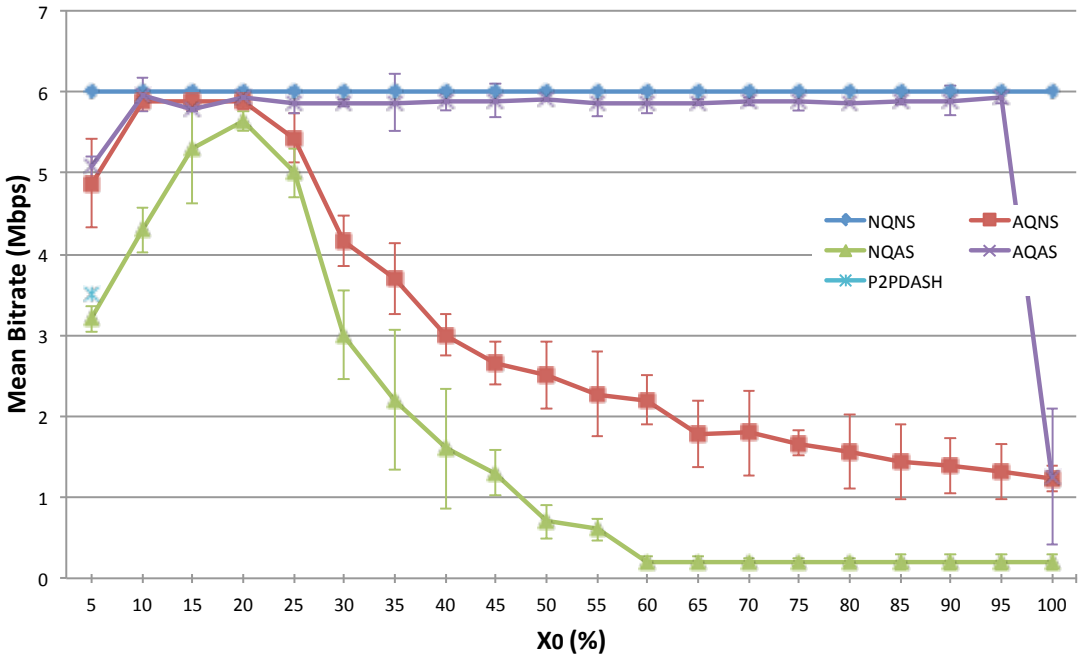

Figure 5.9 – Average mean bitrate after flash crowd

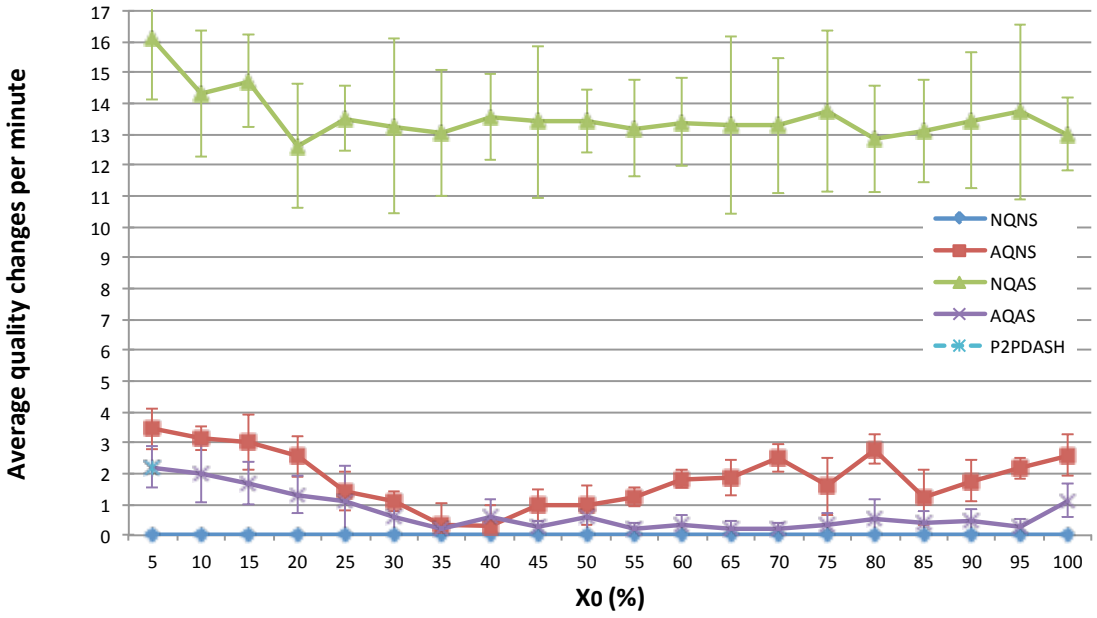

Figure 5.10 – Number of quality changes per minute before flash crowd

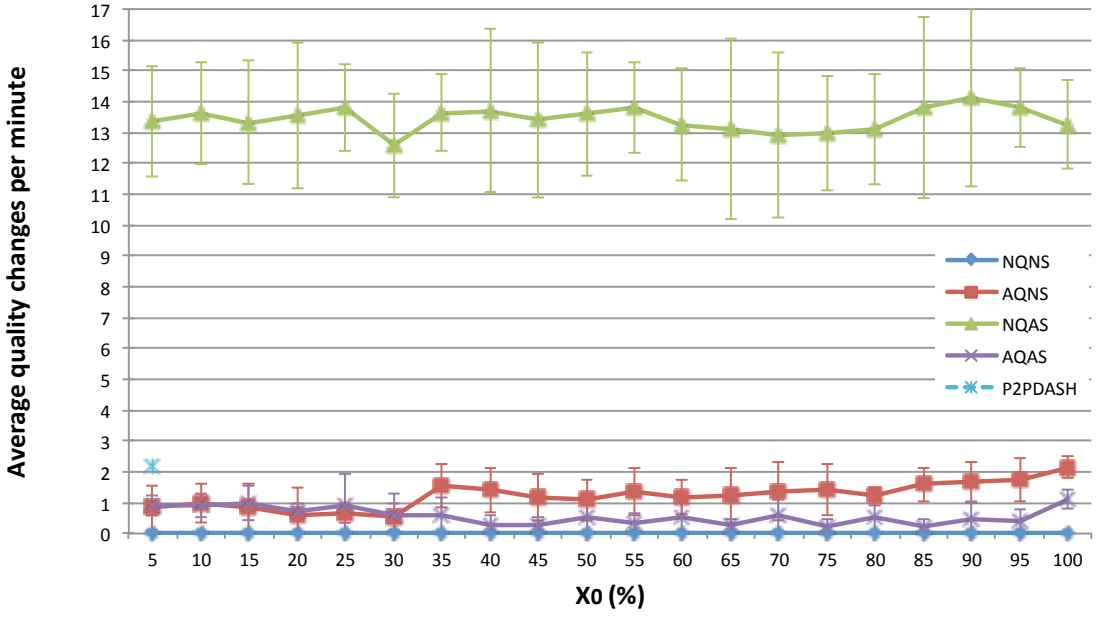

Figure 5.11 – Number of quality changes per minute after flash crowd

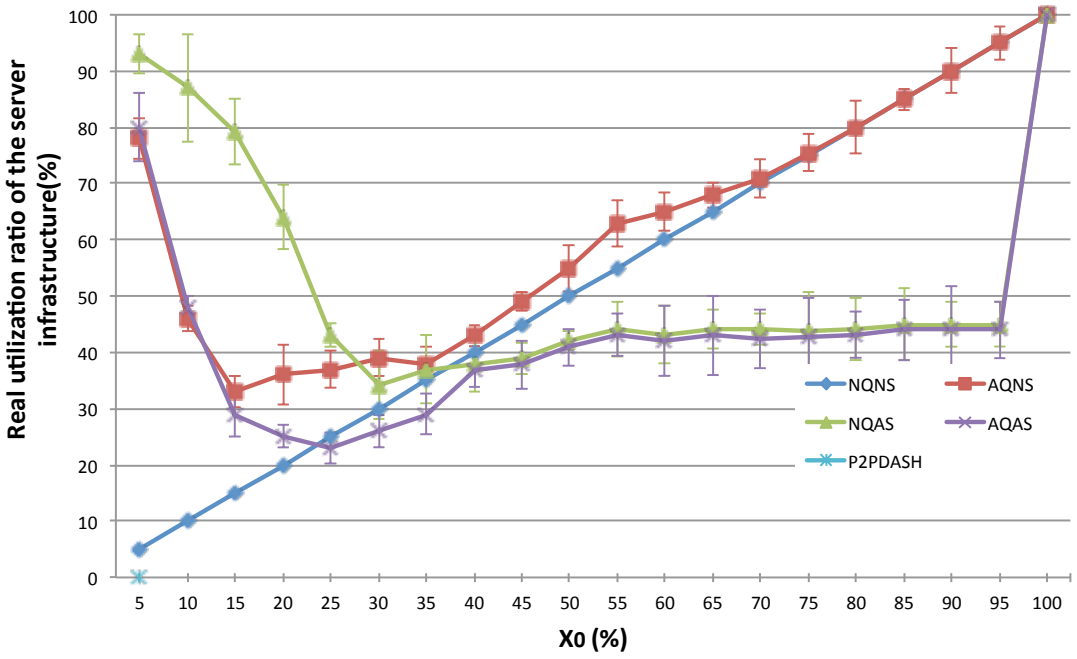

Figure 5.12 – Actual utilization of the server infrastructure before flash crowd

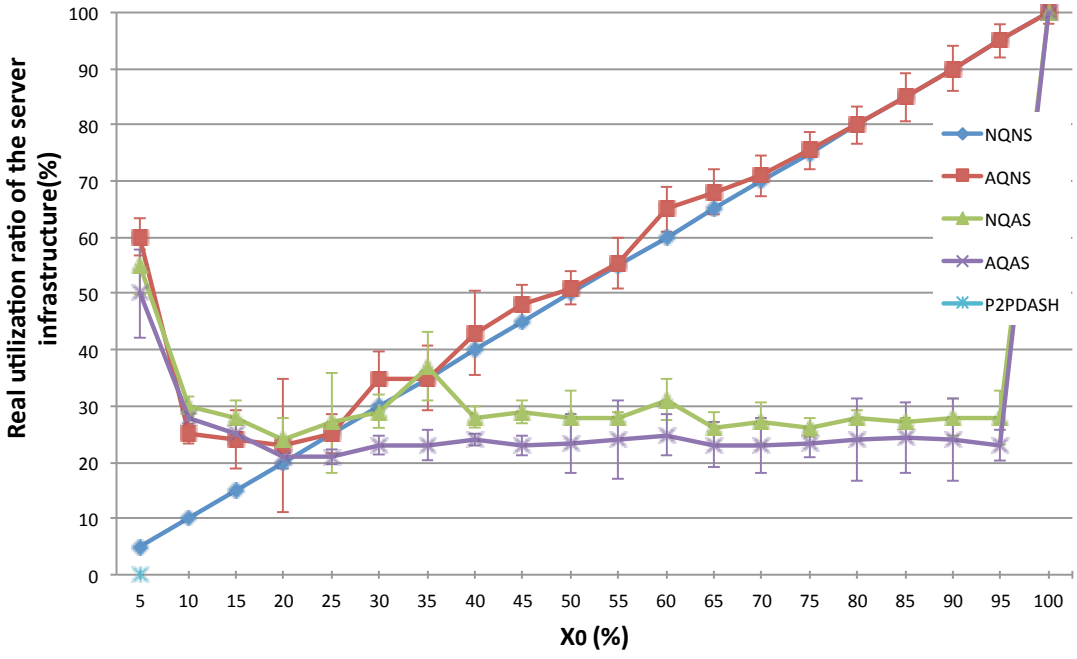

Figure 5.13 – Actual utilization of the server infrastructure after flash crowd

infrastructure becomes too high to sustain the important throughput requests from all concurrent peers. Consequently, the delivered throughput to the client is lower than the infrastructure capacity, resulting into many video freezes.

With NQAS, the value of X_0 is adjusted in time to better offload the server platform and find a good-enough trade-off between the global P2P system efficiency and the server utilization. On Figure 5.6, one can see the number of video stalls per minute that is overall lower than in the NQNS case. However, this number increases when X_0 takes on extreme values (lower than 20%). This is due to the same reasons as NQNS when X_0 is low. Contrarily to NQNS, NQAS is not subject to much video freezing events when the initial server participation to the streaming increases because NQAS peers directly reduce the server contribution as they obtain enough data from their neighbors. When X_0 is 100%, NQAS clients only rely on the server to obtain video segments, the number of rebuffering events drastically increases due to the lack of quality adaptation preventing video stalls.

When quality adaptation is enabled without scale adaptation, the AQNS peers automatically adjust the consumed and re-emitted quality for the good of the entire P2P system. As a result, no matter the initially value of X_0, the video stalls are significantly lower than in NQAS and NQNS. Indeed, video stalls occur between 0.01 and 0.1 per minute when X_0 takes on values higher than 30%. When X_0 falls below 30%, the rebufferings are a bit more frequent due to insufficient P2P data transfer and insufficient remaining time for the peers to fall back on the server infrastructure. As to AQAS that benefits from both adaptation types, the continuity of video playback is further enhanced compared to AQNS.

Regarding the P2P-DASH clients, they obtained a frequency of 2.5 video stalls per minutes, mostly due to the fact that they cannot fall back to the servers when the deadlines of video chunks are about to be missed.

Mean bitrate and number of quality changes

The P2P-DASH clients reach a 2.55 Mbps mean bitrate in average.

The NQAS peers suffer from the lack of quality adaptation and end-up requesting the content in the top quality before having to cancel some segment requests from their neighbors and re-assigning them to the servers. Most of the time, (when X_0 ranges from 5-30% and 55-100%), this is not even sufficient to reach the originally requested quality and the peers have to display the redundant bitrate, which drastically reduces the overal mean bitrates. Thus, NQAS peers only reach a mean bitrate higher than 4Mbps when X_0 is between 35% and 50%. In addition, these

mean bitrates come along with frequent video quality flickering due to the display of the redundant bitrate. NQNS always displayed the top content quality although with rebufferings for almost every new segment download.

When quality adaptation is enabled (AQNS and AQAS), the mean bitrate is improved and quality fluctuations and flickering drastically diminished compared to NQAS. The maximum mean bitrate is obtained when X_0 reaches 30% to 40% for AQNS and slowly declines with higher values of X_0, due to the enforced utilization of the fixed-size server infrastructure with limited throughput capacity. In contrast, the mean displayed bitrate of AQAS plateaus as soon as X_0 is greater than 30% because the AQAS peers locally optimize the usage of the P2P communications while adapting the consumed and re-emitted content quality. This results in tending toward a good functioning of all overlays.

Data consumption from the server infrastructure and content quality distribution

Although NQNS performed as expected because it always requests a fixed quantity of data from the server, and because it always displays the top content quality, the lack of adaptability of NQNS resulted in many video freezing events (almost one every segment).

By contrast, results show that AQNS can sustain a high QoE level and good quality distribution (shown in Figure 5.4) even when X_0 takes on low values between 25% and 40%. In addition, when X_0 reaches higher values and forces peers to consume more data from the server, the server infrastructure becomes saturated. Nevertheless, the PMS quality adaptation algorithm of AQNS decreases the target video quality of consuming peers in order to smoothly degrade QoE. However, the AQNS peers utilize a slightly greater amount of bandwidth from the server than originally set. This is explained by the resiliency mechanisms of PMS in falling back on servers when the P2P communications fail to provide sufficient throughput for the delivery of requested video chunks.

NQAS and AQAS adjust their usage of the server infrastructure and automatically converge to 40-45% of data consumption from the servers when the initial value of X_0 allowed it. For NQAS, when X_0 takes on values between 40% and 50%, an optimal tradoff between the system's utilization and the delivered QoE is achieved. However, when X_0 exceeds 50%, the usage of the server infrastructure reaches its maximum and the mean bitrate suddenly drops due to the lack of quality adaptation mechanisms that result in peers consuming the video at either the 150kbps or

6Mbps quality. Overall, we can find an optimal trade-off between the consumption of video data from the servers and the delivered QoE when X_0 falls between 40-50% for NQAS, between 30-40% for AQNS, and between 25-35% for AQAS.

Interestingly, when X_0 is lower than the 35% limit, all applications seem to reach their minimum of server utilization, but rapidly decrease the data delivered from the P2P communications when X_0 falls below 20-25%. More importantly, we can observe that PMS-AQAS avoids saturating the server infrastructure capacity by increasing its usage of the available throughput between the peers (based on the P2P efficiency and the local P2P system efficiency) and provides the highest QoE level among applications at the lowest server infrastructure utilization. Another important result in our evaluation exposes that when X_0 falls under 35%, the mean bitrate of AQAS and AQNS significantly drops due to too low P2P system efficiency caused by peer starvation. These results on QoE also demonstrate that even though the P2P-DASH streaming application mostly relies on the usage of P2P data exchanges, it obtains a better mean bitrate and less video rebuffering than NQAS when the available server resources are sparse. However, it still suffers from many video stalls in practice (2.5 per minute) compared to AQAS and AQNS.

5.4.5 Results after flash crowd

At the 15^{th} minute, the remaining 150 peers join the experiment. We record the peers' QoE metrics and data consumption from the servers from that moment in time until the end of the experiment, 15 minutes later. Globally, we can observe that the optimal trade-off between the utilization of the server infrastructure and the delivered QoE is found for lower values of X_0 compared to the evaluation results before the flash crowd. This is due to the delivery system being composed of more peers, which in turn increases the global demand of server-side resources and decreases the possible throughput delivered from the servers to each peer individually. Nevertheless, this is counteracted by newly arrived peers that contribute to the system by sharing with their neighbors the video chunks they have cached. Hence, the new peers can provide a great deal throughput for the P2P system that is judiciously utilized in the quality adaptation and scale adaptation algorithms.

Rebuffering per session and average start-up delay

As during the previous 15 minutes, the NQNS application performed the worse in terms of video stalls per minute. Additionally, the massive arrival of peers into the system offsets the number of rebufferings with regards to the value of X_0.

Before the flash crowd, when X_0 ranges in between 5% and 35%, data starvation was the main reason for the important number of video rebuffering. After the flash crowd, the newly arrived peers increased the capacity of the P2P system to transfer the previously cached chunk and reduced data starvation in the P2P overlay. Nevertheless, data starvation stil occurs but only when the peers are forced to utilize the servers at a lower rate (X_0 lower than 10%). Although the new peers could participate to decrease data starvation, as soon as the enforced server utilization exceeds 25%-35%, the server infrastructure becomes over-used and prevents the peers to obtain smooth uninterrupted video playback.

When scale adaptation is used, NQAS can lower the negative flash crowd impact when possible. This results into fewer video stalls, but cannot prevent data starvation when X_0 descends below 20%. AQNS and AQAS do not really suffer from the flash crowd as they rely on quality adaptation to aim at the good functioning of all overlays and prevent video stalls.

Interestingly, the P2P-DASH clients do not experience much variations in the video rebufferings, the value remaining steady at 2.3 video stalls per minute.

Mean bitrate and number of quality changes

There are no observed impacts on the visual quality displayed by the NQNS peers, caused by the absence of content quality adaptation. The AQNS reaches its maximum mean bitrate for enforced value of data consumption from the servers between 10 and 20%. Then, when X_0 increases, the mean bitrate rapidly drops due to an over-utilization of the servers, directly impacting the quality adaptation algorithm. Indeed, the AQNS peers cannot benefit from the large amount of network resources offered by the newly arrived peers. Contrarily, when X_0 is lowered down to 10%, data starvation augments and the mean bitrate decreases.

For the NQAS case, even though a high mean bitrate is obtained at an optimal value of X_0 at 20%, the video quality flickering effects amplify the poor perceived quality from the end-users. Furthermore, as soon as X_0 deviates from this optimum value, the mean bitrate suddenly drops due to the lack of quality adaptation and to the redundant GoPs being displayed, causing a great deal of quality fluctuations (13-14 per minute).

Finally, when both quality and scale adaptation are enabled, AQAS peers perform the best in terms of mean bitrate and quality changes. As soon as the data starvation zone is avoided ($X_0 < 10\%$) the mean bitrate is almost at its maximum, and remains steady regardless of the initially set server utilization.

As to the P2P-DASH clients, the massive arrival of new clients augmented the capacity of the P2P overlays. This resulted in a mean bitrate increase of 1.55 Mbps.

Data consumption from the server infrastructure

Interestingly, we can observe from NQAS and AQAS that both applications lessen their utilization of the server infrastructure and increase the P2P traffic. This translates into a shift in the optimal ratio between the QoE level and X_0 to 20% and 25% for NQAS. The two applications benefiting from the scale adaptation algorithm converge to a lower data consumption from the servers between 20 and 30% when allowed.

Finally, when the AQNS application uses servers only (i.e., $X_0 = 100\%$), it reflects the MS-Stream/MATHIAS use-case. In this case, the clients are using the available throughput to provide the best possible mean bitrate with almost almost no video rebuffering events.

In a nutshell, the PMS protocol, along with the proposed quality and scale adaptation algorithms, leverage the resources of consuming peers to provide the best possible QoE level without over-utilizing the server infrastructure made available.

5.5 Conclusion

In this chapter, we proposed a novel solution for video streaming over the Internet by relying on MS-Stream, MATHIAS and the P2P paradigm: PMS. The multiple-overlay architecture of the solution enables to clearly identify the challenges for content quality adaptation and self-scaling properties in the P2P/Multi-Server environment. We introduced a distributed and decentralized content quality adaptation algorithm aiming at providing the best possible user experience while ensuring the good functioning of all overlays. A local optimization method was proposed to have each peer individually adjusting its utilization of the server infrastructure. We evaluated our proposal in a national testbed hosted in several cities. Results showed the benefits of our solution in terms of QoE and scalability.

Chapter 6

Conclusion and perspectives

"The present is theirs; the future, for which I really worked, is mine"

— Nikola Tesla

End-users' QoE is a major element determining the success and adoption of current or future video streaming services. Video traffic over the Internet will experience a tremendous growth of 118%, and is expected to reach 81.8% of the total Internet traffic by 2020. Given the global trend of increasing video traffic and the issues pertaining to the underlying network architectures, Internet video streaming has gathered considerable attention from industry and academia. The last few years witnessed tremendous deployments of OTT video streaming systems. These deployments are based on a variety of architectures and include: cloud platforms and Content Distribution Networks based on HTTP Adaptive Streaming, P2P networks, and hybrid solutions combining P2P and CDN.

The contributions presented in this thesis can be classified into two main proposals: (1) providing the means to extend the capabilities of HAS solutions in using multiple servers simultaneously in order to increase the end-users' QoE and (2) enhancing the QoE and scalability potential of video streaming solutions by leveraging on content quality adaptation and self-scalability of hybrid P2P/Multi-Server streaming systems. In more details, we have made the following contributions:

- Introduction of an HAS-evolving streaming framework (*MS-Stream*) advocating for a client-centric utilization of multiple servers simultaneously. MS-Stream incorporates a client and a server solution. A MS-Stream client simultaneously requests several servers to deliver independently decodable sub-

segments composed in an online manner from the existing set of content qualities. Sub-segment composition requires a very light pre-processing operation from the servers before sub-segment delivery. When retrieved, the requested video sub-segments are merged to reconstruct and display the original requested content quality. In the event of sub-segment loss or late delivery, content playback continuity is not affected, only content quality is. MS-Stream was part of the contribution to two European projects DELTA [2015] (Eurostars) and DISEDAN [2016](CHIST-ERA).

- Based on the proposed framework, we exposed *MATHIAS*, a set of consumption algorithms embedded into the MS-Stream client to effectively utilize the resources made available to the clients, and to reach a target bitrate while providing as few video stalls as possible. MATHIAS incorporates four principal elements: adaptation of the number of used servers based on bottleneck detection; limiting and minimizing the bandwidth consumption overhead resulting from the data redundancy enabling independently decodable sub-segments; sub-segment scheduling to fairly use the available servers and communications channels; in-segment download adaptation to modify the ongoing downloads of sub-segments and prevent QoE degradation in the event of low performance in the network. MATHIAS enables a distributed and scalable solution for a fair usage of the servers to improve end-users' QoE. The European projects DELTA [2015] and DISEDAN [2016] allowed us to evaluate our solutions with servers in multiple locations in Europe.

- Following on MS-Stream and MATHIAS, we proposed to combine the studied approaches with the P2P paradigm and we presented a hybrid P2P/Multi-Server (*PMS*) solution for live streaming with scalability and quality adaptation capabilities. We detailed the streaming protocol of the PMS architecture. We put forward a distributed and decentralized quality adaptation algorithm relying upon local and global indicators of the functioning of the P2P data transfers.This new quality adaptation strategy aims at enhancing the end-users QoE while concurrently guaranteeing the successful functioning of the content delivery system. We also present a local optimization method to address the self-scalability properties of the PMS solutions where every peer locally minimizes its utilization of the fixed-size server infrastructure without sacrificing the achieved QoE gains.

The research contributions of this thesis have been presented in several interna-

tional conferences and journals. Additionally, we have developped a demonstrator of our work available online (http://msstream.net [MS-Stream, 2017]), which was awarded several prizes for its scientific and technical excellence in international conferences.

This thesis opens perspectives for ongoing and further work. As a first short-term research perspective, we identify one limitation of MS-Stream. In its current status, the MS-Stream solution requires light software add-ons to the HTTP servers to support the MS-Stream API, which is an obstacle for the adoption of an MS-Stream-like solution in existing CDN infrastructures that rely on static content caching. We have already made some progress to modify MS-Stream by relying on HTTP byte range requests for the client to select the GoPs at a given quality that should be delivered from the servers. The client would then have to find a way to retrieve information about the content itself to identify the boundaries of GoPs within video segments. Subsequently, the number of outgoing requests from the client could increase importantly, and we are currently investigating the impact of such approach on streaming stability and fairness, as well as on end-users' QoE.

Then, we identify the following perspectives for which we are planning a submission to the 2018 H2020 ICT-28 call for projects related to "Future Hyper-connected Sociality" into which the topic fits exactly. The latter call for project supports the new social media initiatives, and the transitions to P2P federated social networks based on smart decentralized architectures. Indeed, thanks to its codec agnosticism and DASH-compliance, our work exposes evolving solutions that can be applied to many scenarios, and can create new video streaming experiences and services. We foresee the advent of distributed social network platforms for User-Generated-Content (UGC), hosted at the end-users premises and relying on content delivery solutions similar to the ones presented in this thesis. However, relying on clients to utilize multiple distributed servers in such platform leads to security and privacy challenges. Indeed, by relying on clients to utilize multiple distributed servers, the knowledge of the distributed infrastructure is easily accessible. We will investigate the identified privacy, security and content consistency challenges by relying on the distributed nature of the UGC social network platform. We have started actions in developing a distributed social networks with user-generated multimedia content implementing MS-Stream/PMS for video delivery and we are currently investigating the privacy and security challenges related to this new social content consumption experiment.

With the rise of the popularity of omnidirectional, virtual reality and augmented

reality applications that require even greater amount of network bandwidth compared to traditional live and VoD streaming, we envision the benefits in terms of end-users' QoE and resource usage flexibiliy in using adaptive multiple-source streaming. We have already made some efforts to explore the field of multi-source multi-path streaming for live event streaming as well as live-conferencing in these novel applications. Moreover, in the context of distributed media social network advocating for user-generated content, new challenges rise related to content preparation and production (especially in terms of onmidirectional, virtual reality and augmented reality streaming), end-user business models for sharing their resources to the distributed media social network platform (including crowd-sourcing models for identification and rewarding of resource sharing and user-generated content). Recently, we also have witnessed the emergence of Internet-of-Things (IoT) applications where resources are sparse, highly volatile and distributed. The streaming approach presented in this thesis could easily make use of IoT devices/platforms for the purpose of enhancing QoE. We have started to explore the field of IoT for video streaming and "Social Network of Objects" for novel multimedia consumption applications in distributed media social networks.

Among other perspectives, we are also exploring context-aware adaptation scenarios to enable more interactive and intelligent media services assisted by multiple-source delivery, with advanced ways of content adaptation. Namely, the H2020 ICT-7 Internet of Radio-Light (IoRL) project proposed the Internet of Radio Light for tackling the issues of congestion, interference, security and safety concerns, restricted propagation, poor in-door location accuracy of wireless networks in buildings. The IoRL project aims at developing a safer, more secure, customizable and intelligent building network that reliably delivers increased throughput from access points located within buildings. Efforts are currently being put to utilize the contextual information of such scenario to enhance the delivery of multimedia content over the Radio-Light technology with multiple sources.

The identified research and innovation perspectives have also led to recruit two new Ph.D candidates to tackle these promising topics.

Appendix A

Publications

A.1 Published papers in international peer-reviewed conferences

J. Bruneau-Queyreix, M. Lacaud, D. Negru, J. Batalla, and E. Borcoci, *"QoE Enhancement Through Cost-Effective Adaptation Decision Process for Multiple-Server Streaming over HTTP,"* in IEEE International Conference on Multimedia and Expo (ICME), 2017.

J. Bruneau-Queyreix, M. Lacaud, D. Negru, J. Batalla, and E. Borcoci, *"MS-Stream: A Multiple-Source Adaptive Streaming Solution Enhancing Consumers Perceived Quality,"* in IEEE Consumer Communications and Networking Conference (CCNC), 2017.

J. Bruneau-Queyreix, M. Lacaud, P. Anapliotis D. Negru, *"On providing Multiple-Server Support to Dynamic Adaptive Streaming Applications for Enhanced QoE,"* in IEEE International Conference on Telecommunications and Multimedia (TEMU), 2016.

E. Borcoci, T. Ambarus, **J. Bruneau-Queyreix**, D. Negru, J. Batalla, *"Optimization of Multi-server Video Content Streaming in 5G Environment,"* in IEEE International Conference on Evolving Internet (INTERNET), 2016.

J. Bruneau-Queyreix, D. Negru, J. Batalla and E. Borcoci, *"Multiple Description - DASH: Pragmatic video streaming maximizing End-Users' Quality of*

Experience," in IEEE International Conference on Communication (ICC), 2016.

S. G. Obreja, R. Iorga, E. Borcoci, C. Cernat, M. Vochin, J. Batalla, **J. Bruneau-Queyreix**, D. Negru, *"Over the Top Content Streaming Adaptive System- Implementation and Validation,"* The Ninth International Conference on Communication Theory, Reliability, and Quality of Service (CTRQ), 2016.

J. Bruneau-Queyreix, D. Negru and J. Batalla, *"Home-Box based collaborative caching strategy: An asset for Content Delivery Networks,"* in IEEE International Conference on Telecommunications and Multimedia (TEMU), 2014.

A.2 Published articles in international peer-reviewed journals

J. Bruneau-Queyreix, M. Lacaud, D. Negru, J. Batalla, and E. Borcoci, *"Adding a New Dimension to HTTP Adaptive Streaming through Multiple-Source Capabilities,"* IEEE MultiMedia Magazine (in press), 2017.

J. Batalla, P. Krawiec, C. Mavromoustakis, G. Mastorakis, N. Chilamkurti, D. Negru, **J. Bruneau-Queyreix** and E. Borcoci, *"Efficient Media Streaming with Collaborative Terminals for the Smart City Environment,"* IEEE Communications Magazine, vol. 55, no. 1, pp. 98-104, 2017.

J. Batalla, P. Krawiec, **J. Bruneau-Queyreix**, D Négru, E. Borcoci, A. Beben and P. Wisniewski, *"On providing Cloud-awareness to client's DASH application by using DASH over HTTP 2.0,"* Journal of Telecommunications and Information Technology, no 1/2016.

A.3 Published chapters in international peer-reviewed books

P. Krawiec, J. Batalla, **J. Bruneau-Queyreix**, D. Negru, G. Mastorakis, and C. Mavromoustakis, *"Collaborative Media Delivery in 5G Mobile Cloud Networks,"* in Advances in Mobile Cloud Computing and Big Data in the 5G Era, Springer Vol.22, pp. 341-360, 2017.

E. Markakis, D. Negru, **J. Bruneau-Queyreix**, E. Pallis, G. Mastorakis, C. X. Mavromoustakis, *"A P2P Home-Box Overlay for Efficient Content Distribution,"* in Emerging Innovations in Wireless Networks and Broadband Technologies, IGI, 2016.

A.4 Published demonstrations and posters in international peer-reviewed conferences

J. Bruneau-Queyreix, M. Lacaud, D. Negru, *"A Hybrid P2P/Multi-Server Quality Adaptive Live- Streaming Solution for High End-User's QoE,"* in IEEE International Conference on Multimedia and Expo (ICME), DASH-IF Grand Challenge, 2017.

J. Bruneau-Queyreix, M. Lacaud, D. Negru, *"A Multiple-Source Adaptive Streaming Solution Enhancing Consumers Perceived Quality,"* in IEEE Consumer Communications and Networking Conference (CCNC) – Demonstation track- 2017.

J. Bruneau-Queyreix, D. Negru, J. Batalla and E. Borcoci, *"Home-Boxes: Context-aware distributed middleware assisting content delivery solutions,"* ACM/IFIP/USENIX Middleware. Poster session. Bordeaux (France), 2014.

A.5 Awards

ICME 2017 Grand Challenge Winner: J. Bruneau-Queyreix, M. Lacaud, D. Negru, *"A Hybrid P2P/Multi-Server Quality Adaptive Live- Streaming Solution for High End-User's QoE,"* in IEEE International Conference on Multimedia and Expo (ICME), DASH-IF Grand Challenge, 2017.

CCNC 2017 Best demonstration award: J. Bruneau-Queyreix, M. Lacaud, D. Negru, *"A Multiple-Source Adaptive Streaming Solution Enhancing Consumers Perceived Quality,"* in IEEE Consumer Communications and Networking Conference (CCNC) – Demonstation track- 2017.

TEMU 2014 Best student paper award: J. Bruneau-Queyreix, D.

Negru and J. Batalla, *"Home-Box based collaborative caching strategy: An asset for Content Delivery Networks,"* in IEEE International Conference on Telecommunications and Multimedia (TEMU), 2014.

A.6 Reports

J. Mongay Batalla, P. Krawiec, Wojciech Burakowski, **J. Bruneau-Queyreix**, C. Mavromoustakis, G. Mastorakis, Y. Kryftis, S. Battista, K. Dudek, *"DELTA - D03: Demonstrator definition and specification,"* 2015.

C. Mavromoustakis, G. Mastorakis, P. Krawiec, J. Mongay Batalla, M. Gajewski, W. Latoszek, V. Sobiecki, J. Bielecki, S. Battista, **J. Bruneau-Queyreix**, K. Christoforou, A. Kyprianou, *"DELTA - D04: MDM development and implementation,"* 2015.

P. Krawiec, J. Mongay Batalla, M. Gajewski, W. Latoszek, V. Sobiecki, Jarosaw Bielecki, S. Battista, C. X. Mavromoustakis, G. Mastorakis, Y. Kryftisv, **J. Bruneau-Queyreix**, K. Christoforou (CABLENET), A. Kyprianou (CABLENET), *"DELTA - D05: MAS module development and implementation,"* 2015.

P. Krawiec, J. Mongay Batalla, M. Gajewski, W. Latoszek, V. Sobiecki, J. Bielecki, S. Battista, C. X. Mavromoustakis, G. Mastorakis, Yiannos Kryftisv, **J. Bruneau-Queyreix**, K. Christoforou, A. Kyprianou, *"DELTA - D06: MQM module development and implementation,"* 2015.

J. Bruneau-Queyreix, D. Négru, *"DELTA - D07: EHG development and implementation,"* 2015.

C. X. Mavromoustakis, G. Mastorakis, P. Krawiec, J. Mongay Batalla, M. Gajewski, W. Latoszek, V. Sobiecki, J. Bielecki, S. Battista, **J. Bruneau-Queyreix**, K. Christoforou, A. Kyprianou *"DELTA - D08: Media Services Manager,"* 2015.

P. Krawiec, J. Mongay Batalla, M. Gajewski, K. Christoforou, **J. Bruneau-**

Queyreix, *"DELTA - D09: Demonstrator Integration,"* 2015.

P. Krawiec, J. Mongay Batalla, M. Gajewski, Yiannos Kryftis, Constantinos Mavromoustakis, G. Mastorakis, **J. Bruneau-Queyreix**, *"DELTA - D10: DELTA trials, experimentation and evaluation,"* 2015.

E. Borcoci, S. Obreja, R. Badea, O. Catrina, C. Cernat, J.M. Batalla, A. Beben, **J. Bruneau Queyreix**, D. Negru, *"DISEDAN - D2.1 System requirements and comparative analysis of existing solutions for media content server selection and media adaptation,"*, 2014.

E. Borcoci, O. Catrina, M. Constantinescu, R. Badea, A. Constantinescu, J.M. Batalla, A. Beben, P. Krawiec, **J. Bruneau Queyreix**, D. Negru, *"DISEDAN - D2.2 Specification of the multi-criteria decision process algorithms and mechanisms for media content server selection,"*, 2015.

J. Mongay Batalla, A. Beben, P. Krawiec, P. Wisniewski, **J. Bruneau-Queyreix**, D. Negru, E. Borcoci, R. Badea, *"DISEDAN - D2.3 Specification of the Dual adaptation mechanism,"*, 2014.

O. Catrina, E. Borcoci, R. Badea, C. Cernat, R. Iorga, J.M. Batalla, A. Beben, P. Krawiec, **J. Bruneau Queyreix**, D. Negru, *"DISEDAN - D2.2 Media content server selection and dual adaptation algorithms validation in large-scale context,"*, 2015.

J. Bruneau Queyreix, D. Negru, J. Mongay Batalla, P. Krawiec, A. Beben, E. Borcoci, O. Catrina, M. Constantinescu, R. Badea, A. Constantinescu, *"DISEDAN - D4.1 Final Integration of DISEDAN System,"*, 2015.

J. Bruneau-Queyreix, D. Negru, J. Mongay Batalla, A. Beben, P. Krawiec, P. Wisniewski, S. Obreja, R. Iorga, E. Borcoci, C. Cernat, M. Vochin, *"DISEDAN - D4.2 Final Validation of the DISEDAN integrated Pilot,"*, 2015.

Appendix B

Résumé en Français

Les solutions de streaming vidéo adaptatives basées sur l'utilisation du protocol HTTP ont été largement plébiscitées dans les mondes de l'industrie et de la recherche, notamment pour les possibilités d'améliorations de qualité d'experience qu'elles offrent ainsi que pour leurs facilités de déploiement liées au protocol HTTP. Le standard MPEG-DASH (Dynamic Adaptive Streaming over HTTP ISO/IEC MPEG [2014]; Sodagar [2011] est rapidement devenu l'une des approches la plus utilisée dans le monde du streaming vidéo sur Internet. Les solutions basés sur DASH permettent de stocker au niveau d'un server HTTP plusieurs versions d'un même contenu vidéo encodées sur de multiple qualités. Chaque version, aussi appelée représentation, est ensuite découpée en segment de quelques secondes (Figure B.1). De cette façon, le client peut recupérer sequentiellement des segments de vidéo tout en adaptant la qualité demandée et en evitant les problemes de transmission liés à une surcharge du réseau.

B.1 MS-Stream: Vers une évolution pragmatique des solutions de streaming adaptatives centrées client et basées sur HTTP pour l'utilisation du plusieurs serveurs en simultané

Pour autant, bien que les solutions telles que DASH permettent d'augmenter la qualité d'experience (QoE) utilisateurs en diminuant la qualité de la vidéo transmise sur les réseaux pour minimiser les interruptions vidéo liées au temps de chargement, la qualité intrinsèque de la vidéo est limitée par les capacités physiques du chemin

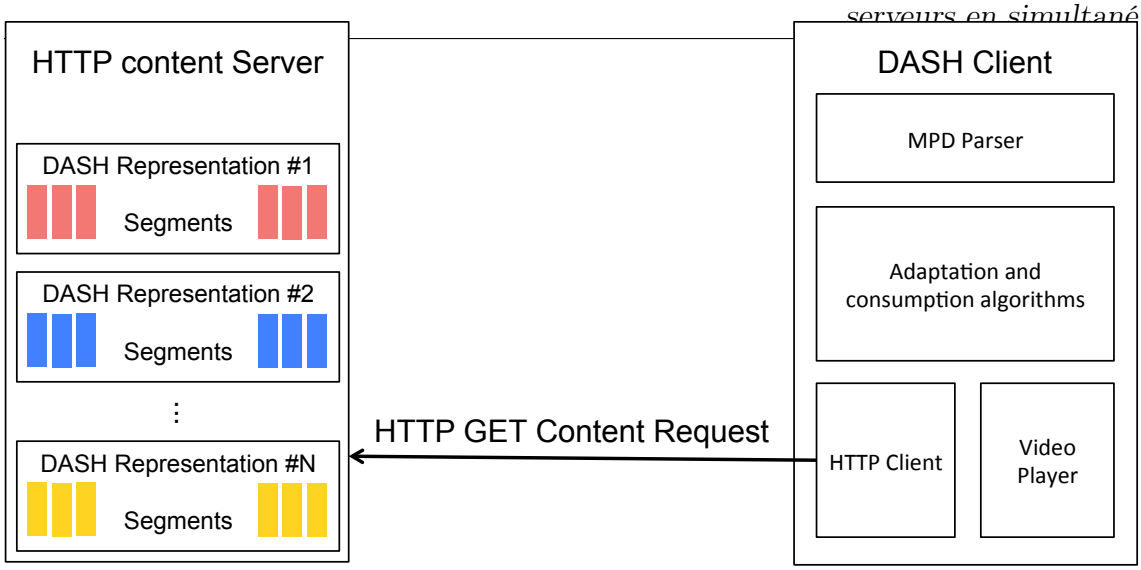

Figure B.1 – Architecture du modèle client-server de DASH

entre le serveur utilisé et le client. Dans l'objectif d'augmenter la qualité d'experience utilisateurs et de diminuer les couts de déploiements des services de streaming, les travaux de cette thèse de doctorat proposent de faire évoluer de façon pragmatique les solutions de streaming adaptatives actuelles vers l'utilisation en simultané de plusieurs sources (serveurs ou pairs).

La première contribution de cette thèse présente MS-Stream, un framework évolutif de streaming adaptatif basé sur HTTP et utilisant plusieurs serveurs simultanément. MS-Stream offre la possibilité d'exploiter la bande passante disponible dans les infrastructures distribuées et les réseaux hétérogènes. La Figure B.2 met en evidence l'évolutivité de notre solution quand à l'architecture technique proposée.

Le scenario envisagé est decrit ci-dessous:

1. Le client MS-Stream emet de façon simultané des requetes aux serveurs MS-Stream afin de créer et de délivrer des *sous-segments* via l'API HTTP exposé au niveau des serveurs. Le client explicite alors, en paramètre des requetes envoyées, la façon dont les sous-segments doivent etre generés.

2. La création des sous-segments par les serveurs suit un schema de codage à description multiple (MDC) [Kazemi *et al.*, 2013]. Pour créer un sous-segment, un serveur va donc alterner de façon successive la qualité des Group of Pictures (GoPs) qui composent les segments vidéo. Le sous-segment resultant est alors une séquence lisible de GoPs à une qualité souhaité par le client et à une qualité extremement faible afin de minimiser les couts de transmission sur le

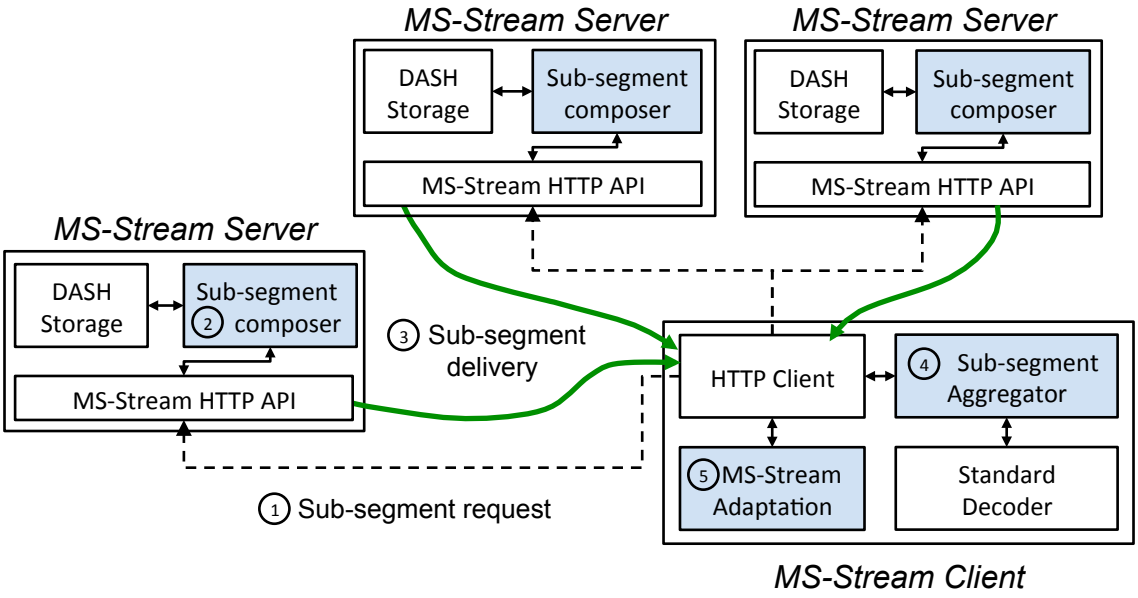

Figure B.2 – Architecture client-serveur MS-Stream

réseaux.

3. Les sous-segments sont transmis sur le réseau et le client MS-Stream assure la synchronization des données en fonction des variations de débit venant de chacun des serveurs. En cas de retard dans la livraison des sous-segments, le client MS-Stream peut prendre la décision d'annuler le téléchargement des sous-segments pouvant causer des interruptions vidéo. Ce faisant, la qualité de la vidéo peut s'en retrouver diminuer, mais la continuité du visionnage de la vidéo n'est pas impactée et la QoE s'en retrouve améliorée.

4. Une fois les sous-segments reçus, le client MS-Stream va pouvoir, grâce au *Sub-segment aggregator*, reconstruire un segment vidéo avec la meilleure qualité visuelle possible à partir des GoPs des sous-segments receptionnés. La vidéo est ensuite décodée grâce à un *Standard video decoder*;

5. Enfin, dans le module *MS-Stream Adaptation*, des techniques d'adaptation peuvent etre proposées pour utiliser au mieux les resources réseau maintenant disponibles grâce à l'utilisation de plusieurs sources en simultané.

B.2 MATHIAS: Algorithmes de streaming adaptatifs multi-sources

La deuxieme contribution de cette thèse est MATHIAS, un groupe d'algorithmes d'adaptation centrés client, implémentés dans MS-Stream, qui a pour vocation d'optimiser l'utilisation des ressources réseau hétérogènes mises à disposition du client pour obtenir une qualité vidéo cible. Le fonctionnement des heuristiques de MATHIAS est décrit dans la Figure B.3.

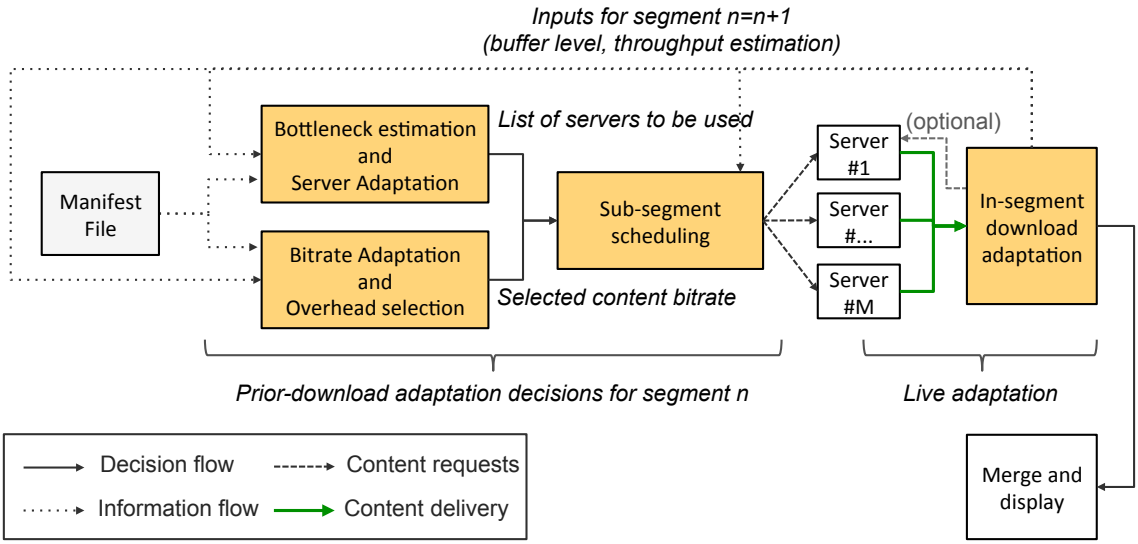

Figure B.3 – Adaptation et consommation de contenu dans MS-Stream avec les heuristiques MATHIAS. Les modules *Bottleneck Estimation* et *Server Adaptation* ne sont utilisés qu'une fois tout les deux telechargement de segment vidéo.

Le rôle de MATHIAS est permet à chaque client de controller (1) le nombre de serveur utilisé en simultané grâce à une methode d'estimation de la source de congestion dans le réseau (coté client, ou coté serveur), (2) d'adapter la consommation supplémentaire de bande passante réseau découlant de la transmission de GoPs en qualité très faible qui ne sont pas forcément affichés sur le terminal de l'utilisateur, (3) de faire face à l'hétérogeneité des resources disponibles pour assigner la livraison des GoPs en qualité haute aux serveurs les plus performant, et (4) de réagir aux fluctuations soudaines et non-anticipées des capacités des serveurs tout en donnant à l'utilisateur une experience de streaming ininterrompu.

B.3 PMS: Une solution de streaming hybride Pair-à-Pair/Multi-Serveur avec adaptation de la qualité et de l'echelle du système

Pour finir, nous allons plus loin dans les capacités de mise à l'echelle et de qualité d'experience de MS-Stream et MATHIAS en tirant profit des ressources physiques des consommateurs. Nous proposons la troisième contribution de cette thèse; une solution hybride pair-à-pair/multi-server de streaming adaptatif: PMS.

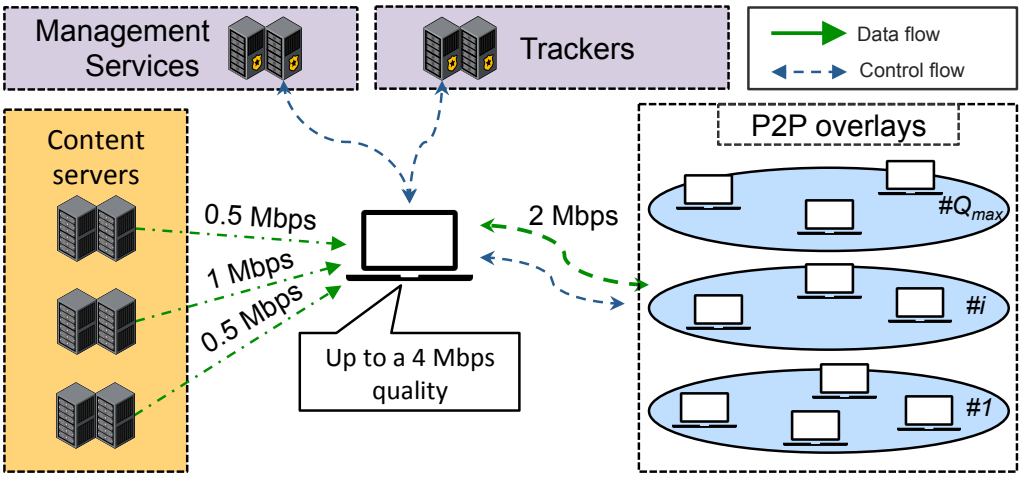

Figure B.4 – PMS high level solution overview

Dans PMS (FigureB.4), nous proposons une architecture multi-overlay dans laquelle chaque overlay est identifiée par la qualité de la vidéo consommée et ré-émise par les pairs qui la compose. De ce fait, un pair ne peut faire partie que d'une seule overlay à la fois. Chaque pair peut récupérer la vidéo à la fois depuis ces voisins, et à la fois depuis les serveurs. Grâce aux metriques systémiques remontées par les pairs aux trackers, ces derniers peuvent calculer des indiquateurs globaux sur le bon fonctionnement de chacune des overlays et de la livraison de chacune des qualités.

Au sein de PMS, nous proposons deux logiques distribuées d'adaptation de la qualité vidéo et de l'echelle du système pour permettre à chaque client de tendre vers une utilisation optimale de l'infrastructure de streaming tout en consommant la plus haute qualité video possible. La Figure B.5 met en perspective l'algorithme precedemment proposé (MATHIAS) pour l'utilisation des serveurs, tout en l'étendant afin de répondre aux problématiques d'adaptation de la qualité et de l'echelle du

système. Le rôle de l'algorithme d'adaptation de la qualité et de permettre à chaque pair de visualiser la meilleure qualité possible en controllant la transition du pair d'une overlay à une autre, tout en essayant de garantir le bon fonctionnement des transmissions de données opérées dans l'overlay. Pour ce faire, le pair se base à la fois sur les indicateurs globaux calculés par les trackers, mais aussi sur des indicateurs locaux de la session de streaming dans l'objectif d'avoir une vision plus précise du fonctionnement du système quand à sa capacité à fournir un débit réseau demandé. Finalement, l'algorithme d'adaptation de l'echelle du système va permettre à chaque pair d'adapter le pourcentage X de GoPs demandés aux serveurs dans l'objectif de diminuer le plus possible la contribution des serveurs à la transmissions de données tout en ne perdant pas les gains de qualité et de QoE réalisés jusqu'à présent. De cette manière, chaque pair va pouvoir locallement optimiser le débit demandé aux serveurs et aux pairs voisins.

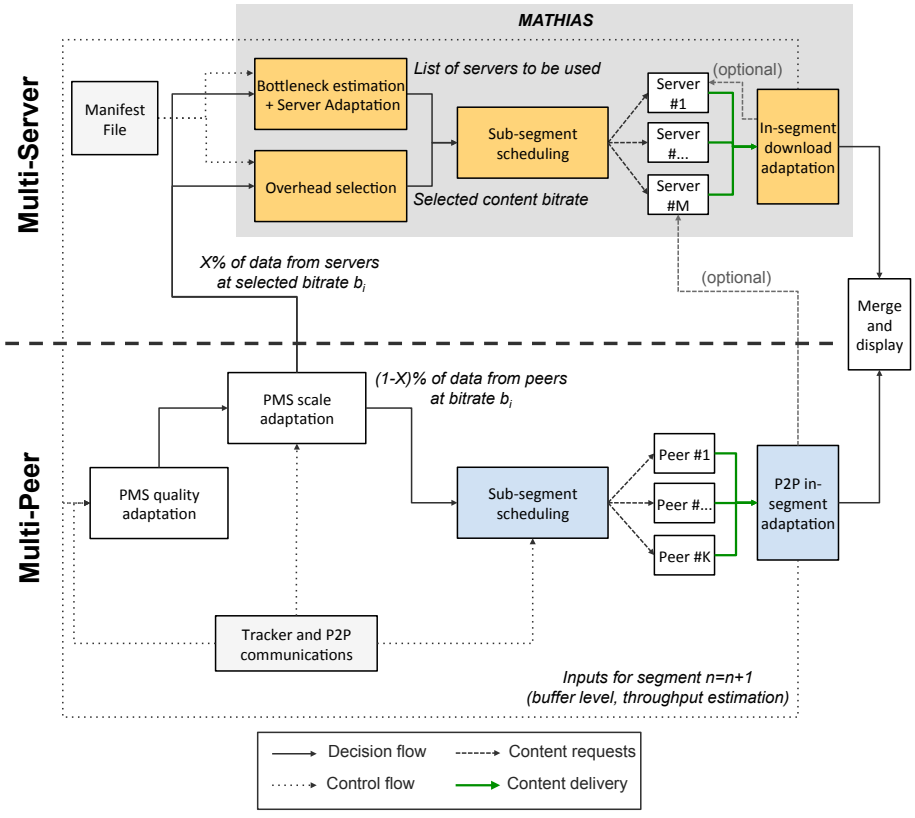

Figure B.5 – PMS consumption decisioning for quality and scale adaptation

La méthodologie d'évaluation lors de ces travaux à consister à valider nos hypothèse de recherche en environnements controllés dans un premier temps avant de passer à des évaluations larges echelles avec un banc de test à echelle national.

Table B.1 – Application de streaming evalué

Protocol	Methode de consommation et d'adaptation
PMS-NQNS	Non-Quality-adaptive, and Non-Scalability-adaptive: version basique du client PMS sans adaptation de qualité ni d'adaptation de mise à l'echelle
PMS-AQNS	Adaptive-Quality and Non-Scalability-adaptive: version améliorée avec adaptation de la qualité uniquement
PMS-NQAS	Non-Quality-adaptive and Scalability-Adaptive: version améliorée avec adaptation de l'echelle uniquement
PMS-AQAS	Adaptive-Quality Scalability-Adaptive: AQNS + NQAS
P2P-DASH [Merani and Natali, 2016]	P2P-DASH: ré-implementation d'un papier de recherche récent sur la combinaison de P2P et de DASH qui propose uniquement un algorithme d'adaptation de la qualité en se basant principalement sur la transmission de données entre pairs (l'utilisation globale du serveur ne dépasse pas 4 fois le bitrate moyen de la vidéo consommée par les pairs)

En ce qui concerne les evaluations larges echelles, quinze serveurs ayant une capacité de débit en upload de 25 Mbps ont été utilisés et étaient répartis entre Paris Roubaix, Bordeaux et Versailles. 300 client-pairs ont été déployés dans ces villes afin de tester nos propositions. Nous avons évalué les metriques de QoE suivantes: Bitrate vidéo moyen (mean bitrate, Figure B.6), nombre d'interruption vidéo (video reuffering events, Figure B.7), nombre de changement de qualité (quality changes, Figure B.8). Afin d'évaluer l'utilisation du système faite par les pairs, nous affichons également le pourentage moyen de GoPs que chauqe pairs demande aux serveurs sur la durée de leur session de streaming (Figure B.9).

Nous avons implémenté les cinq applications de streaming listées dans la Table B.1, ce qui nous permet d'isoler chacune de nos contributions et d'observer l'impact de chacune d'entre elles sur la QoE des utilisateurs et sur la scalabilité du système. Dans chaque expérience, une application de streaming est selectionné pour tous les pairs, et une valeur initiale du pourcentage de GoPs demandé par chaque pair aux serveurs est choisie. Les résultats nous permettent dans un premier temps de montrer les avantages en termes de bitrate vidéo moyen et d'interruptions vidéo de notre approche avec PMS-AQAS vis-à-vis de l'approche de l'état de l'art P2P-DASH. En revanche, l'approche de l'état de l'art permet d'obtenir une meilleure mise à l'echelle du système étant donnée qu'un quantité fixe et extremement faible de données est délivrée depuis les serveurs aux pairs.

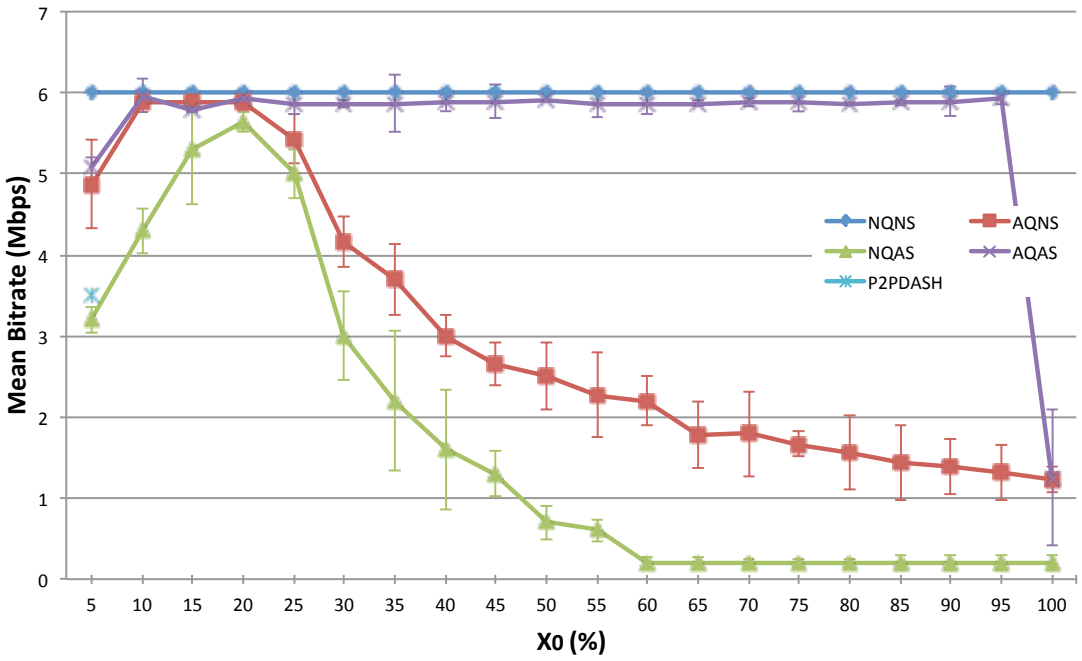

Figure B.6 – Bitrate vidéo moyen par minute

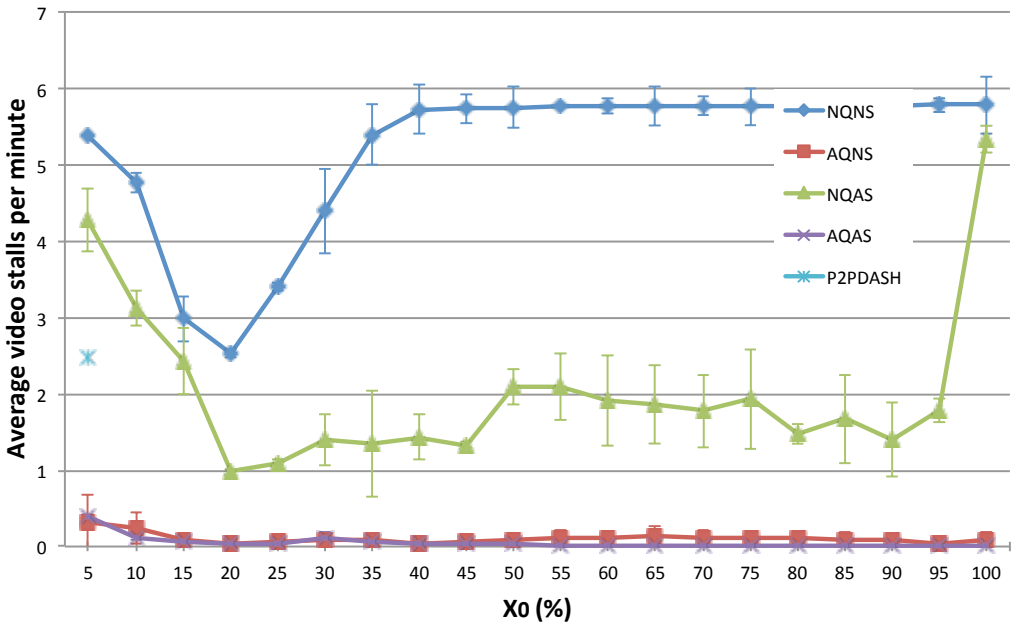

Figure B.7 – Nombre d'interruptions vidéo par minute

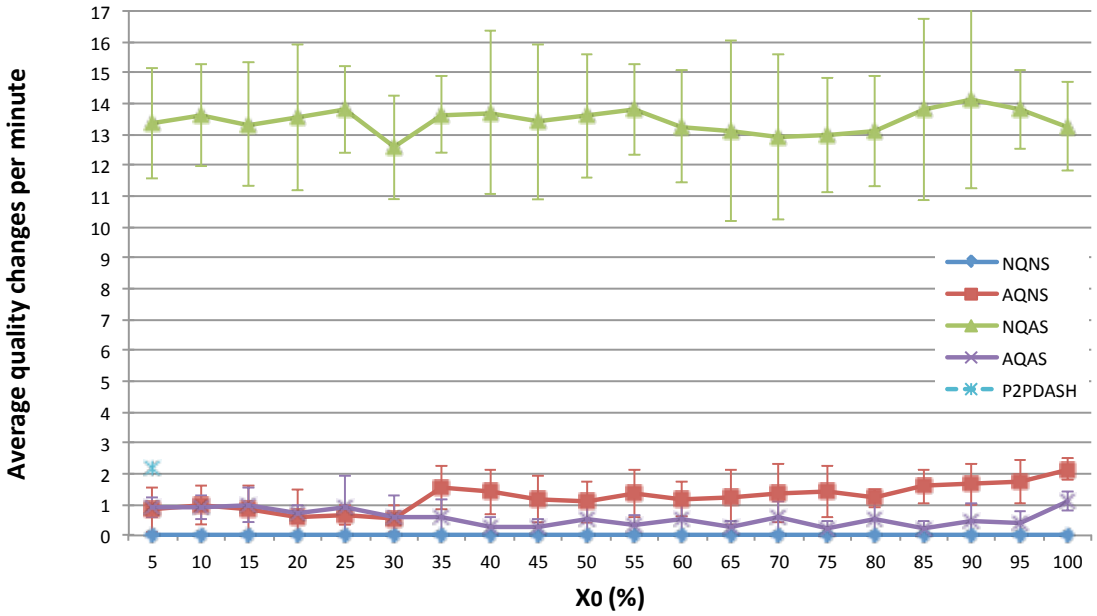

Figure B.8 – Nombre de changements de qualité par minute

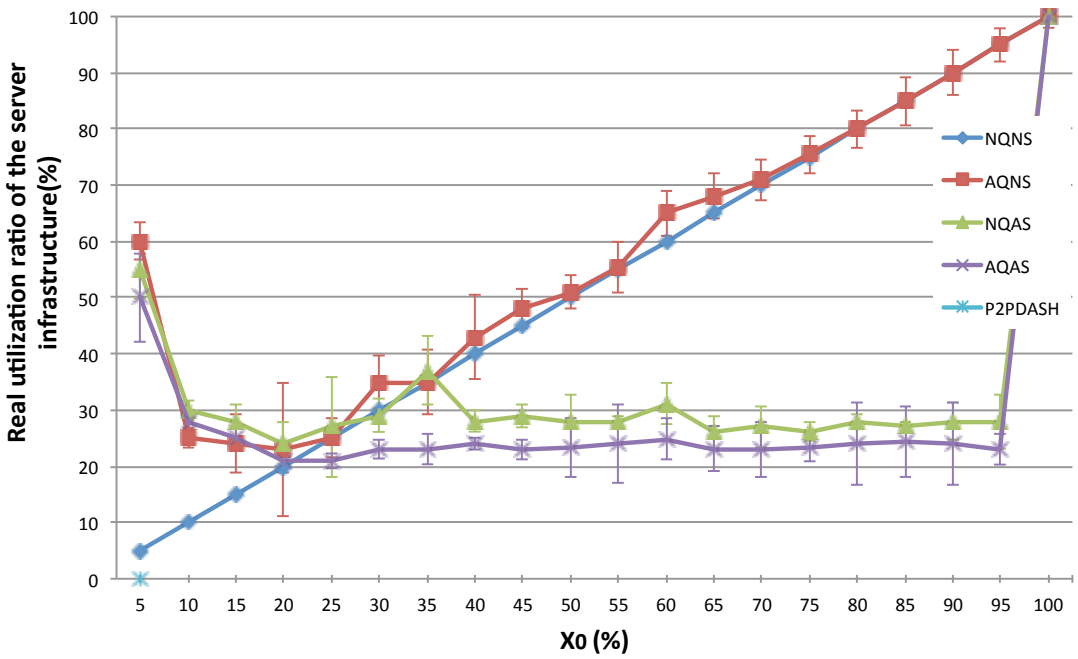

Figure B.9 – Pourcentage moyen de GoPs demandé aux serveurs

Bibliography

3GPP, 2012. Transparent end-to-end Packet-switched Streaming Service (PSS); Progressive Download and Dynamic Adaptive Streaming over HTTP (3GP-DASH). URL https://www.qtc.jp/3GPP/Specs/26247-b10.pdf 17

Abboud, O., Zinner, T., Pussep, K., Al-Sabea, S. and Steinmetz, R., 2011. On the Impact of Quality Adaptation in SVC-based P2P Video-on-demand Systems. *Proceedings of the Second Annual ACM Conference on Multimedia Systems - MMSys*. doi:10.1145/1943552.1943582. 22, 33

Abdelzaher, T., Diao, Y., Hellerstein, J. L., Lu, C. and Zhu, X., 2008. Introduction to Control Theory And Its Application to Computing Systems. *Spinger Performance Modeling and Engineering*, page 185–215. doi:10.1007/978-0-387-79361-0_7. 22, 28

Adhikari, V. K., Guo, Y., Hao, F., Hilt, V. and Zhang, Z. L., 2012a. A tale of three CDNs: An active measurement study of Hulu and its CDNs. *2012 Proceedings IEEE INFOCOM Workshops*. doi:10.1109/infcomw.2012.6193524. 50

Adhikari, V. K., Jain, S., Chen, Y. and Zhang, Z. L., 2012b. Vivisecting YouTube: An active measurement study. *2012 Proceedings IEEE INFOCOM*. doi:10.1109/infcom.2012.6195644. 50

Adhikari, V. K., Jain, S. and Zhang, Z. L., 2010. YouTube traffic dynamics and its interplay with a tier-1 ISP. *Annual conference on Internet measurement - IMC 10*. doi:10.1145/1879141.1879197. 50

Adhikari, V. K. *et al.*, 2012c. Unreeling netflix: Understanding and improving multi-CDN delivery. *IEEE INFOCOM*. 2, 4, 48, 50, 64

Adobe, 2010. Dynamic streaming in Flash Media Server 3.5 – Part 1: Overview of the new capabilities | Adobe Developer Connection. http://www.adobe.com/devnet/adobe-media-server/articles/dynstream_advanced_pt1.html. 3, 17

Akamai, . Akamai Edge Flash Demo. http://wwwns.akamai.com/hdnetwork/demo/flash/default.html. Online; Accessed August 2017. 3

Akhshabi, S., Anantakrishnan, L., Begen, A. C. and Dovrolis, C., 2012a. What happens when HTTP adaptive streaming players compete for bandwidth? *Proceedings of the 22nd international workshop on Network and Operating System Support for Digital Audio and Video - NOSSDAV 12*. doi:10.1145/2229087.2229092. 16, 25, 26, 41, 42, 43

Akhshabi, S., Anantakrishnan, L., Dovrolis, C. and Begen, A. C., 2013. Server-based traffic shaping for stabilizing oscillating adaptive streaming players. *ACM Workshop on Network and Operating Systems Support for Digital Audio and Video - NOSSDAV*. doi:10.1145/2460782.2460786. 35

Akhshabi, S., Begen, A. and Dovrolis, C., 2011. Experimental evaluation of rate-adaptation algorithms in adaptive streaming over HTTP. *Proceedings of the 2nd ACM conference on Multimedia systems - MMSys '11*. 23, 24

Akhshabi, S., Narayanaswamy, S., Begen, A. C. and Dovrolis, C., 2012b. An experimental evaluation of rate-adaptive video players over HTTP. *Signal Processing: Image Communication*, 27(4):271–287. doi:10.1016/j.image.2011.10.003. 22, 24, 27, 28, 41, 42

Alcock, S. and Nelson, R., 2011. Application flow control in YouTube video streams. *ACM SIGCOMM Computer Communication Review*, 41(2):24. doi:10.1145/1971162.1971166. 35, 36

Andelin, T., Chetty, V., Harbaugh, D., Warnick, S. and Zappala, D., 2012. Quality selection for Dynamic Adaptive Streaming over HTTP with Scalable Video Coding. *ACM Conference on Multimedia Systems - MMSys*. doi:10.1145/2155555.2155580. 22, 33, 34

Anjum, Nasreen, Karamshuk, Dmytro, Shikh-Bahaei, Mohammad and Sastry, Nishanth, 2017. Survey on peer-assisted content delivery networks. *Computer Networks*, 116:79–95. doi:10.1016/j.comnet.2017.02.008. 4

Apostolopoulos, J.G. and Trott, M.D., 2004. Path diversity for enhanced media streaming. *IEEE Communications Magazine*, 42(8):80–87. doi:10.1109/mcom.2004.1321395. 73

Applegate, D., Archer, A., Gopalakrishnan, V., Lee, S. and Ramakrishnan, K. K., 2010. Optimal content placement for a large-scale VoD system. *International COnference on - Co-NEXT 10*. doi:10.1145/1921168.1921174. 49

Baccelli, F., Mathieu, F., Norros, I. and Varloot, R., 2013. Can P2P networks be super-scalable? *2013 Proceedings IEEE INFOCOM*. doi:10.1109/infcom.2013.6566973. 56

Balachandran, A., Sekar, V., Akella, A. and Seshan, S., 2013. Analyzing the potential benefits of CDN augmentation strategies for internet video workloads. *2013 conference on Internet measurement conference - IMC 13*. doi:10.1145/2504730.2504743. 49, 50, 54

Basso, S., Servetti, A., Masala, E. and Martin, J. C., 2014. Measuring DASH streaming performance from the end users perspective using neubot. *ACM Multimedia Systems Conference on - MMSys*. doi:10.1145/2557642.2563671. 22, 31, 34

Begen, A., Streeter, K., Bouazizi, I. and Denoual, F., 2016a. MPEG DASH Requirements for a webpush Protocol.
URL https://www.ietf.org/proceedings/91/slides/slides-91-webpush-3.pdf xiii, 37, 38

Begen, A., Streeter, K., Bouazizi, I. and Denoual, F., 2016b. MPEG DASH Requirements for a webpush Protocol.
URL https://tools.ietf.org/html/draft-begen-webpush-dash-reqs-00 35, 37

Belshe, M., Peon, R. and Thomson, M., 2015. Hypertext Transfer Protocol Version 2 (HTTP/2).
URL https://tools.ietf.org/html/rfc7540 39

Bernet, Y., Ford, P., Yavatkar, R., Baker, F., Zhang, L., Speer, M., Braden, R., Davie, B., Wroclawski, J. and Felstaine, E., 2000. A Framework For Integrated Services Operation Over Diffserv Networks.
URL https://tools.ietf.org/html/rfc2998 10

Blender, Foundation, 2017. Big Buck Bunny Video. https://peach.blender.org/. Online; Accessed August 2017. 79

Bobarshad, H., Schaar, M. V., Aghvami, A. H., Dilmaghani, R. S. and Shikh-Bahaei, M. R., 2012. Analytical Modeling for Delay-Sensitive Video Over WLAN. *IEEE Transactions on Multimedia*, 14(2):401–414. doi:10.1109/tmm.2011.2173477. 57

Bobarshad, H., Schaar, M. V. and Shikh-Bahaei, M. R., 2010. A Low-Complexity Analytical Modeling for Cross-Layer Adaptive Error Protection in Video Over WLAN. *IEEE Transactions on Multimedia*, 12(5):427–438. doi:10.1109/tmm.2010.2050734. 57

Bouten, N., Latre, S., Famaey, J., Leekwijck, W. V. and Turck, F. D., 2014. In-Network Quality Optimization for Adaptive Video Streaming Services. *IEEE Transactions on Multimedia*, 16(8):2281–2293. doi:10.1109/tmm.2014.2362856. 29

Bouten, Niels, Famaey, Jeroen, Latré, Steven, Huysegems, Rafael, De Vleeschauwer, Bart, Van Leekwijck, Werner and De Turck, Filip, 2013. QoE Optimization Through In-network Quality Adaptation for HTTP Adaptive Streaming. In *Proceedings of the 8th International Conference on Network and Service Management*, CNSM '12, pages 336–342. International Federation for Information Processing.
URL http://dl.acm.org/citation.cfm?id=2499406.2499460 29

Braden, R., Zhang, L., Berson, S., Herzog, S. and Jamin, S., 1997. Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification.
URL https://tools.ietf.org/html/rfc2205 10

Cable Television Laboratories, Inc, 2016. IP Multicast Adaptive Bit Rate).
URL https://community.cablelabs.com/wiki/plugins/servlet/cablelabs/alfresco/download?id=3edb1609-17ff-4844-87ed-124314a73e7c 13

Camarillo, G., 2009. Peer-to-Peer (P2P) Architecture: Definition, Taxonomies, Examples, and Applicability.
URL https://tools.ietf.org/html/rfc5694 51

Capovilla, N., Eberhard, M., Mignanti, S., Petrocco, R. and Vehkaper, J., 2010. An Architecture for Distributing Scalable Content over Peer-to-Peer Networks. In *2010 Second International Conferences on Advances in Multimedia*, pages 1–6. doi:10.1109/MMEDIA.2010.17. 52

Cardwell, N., 2016. BBR: Congestion-Based Congestion Control.
URL https://queue.acm.org/detail.cfm?id=3022184 35, 40

Carlucci, G., Cicco, L. De and Mascolo, S., 2015. HTTP over UDP: An Experimental Investigation of QUIC. *Proceedings of the 30th Annual ACM Symposium on Applied Computing - SAC 15*. doi:10.1145/2695664.2695706. 40

Castro, M., Druschel, P., Kermarrec, A., Nandi, A., Rowstron, A. and Singh, A., 2003. SplitStream. *ACM symposium on Operating systems principles - SOSP 03*. doi:10.1145/945472.945474. 52, 60

Chellouche, S.A., Negru, D., Chen, Y. and Sidibe, M., 2012. Home-box-assisted content delivery network for internet video-on-demand services. *2012 IEEE Symposium on Computers and Communications (ISCC)*. 53

Chen, L., Zhou, Y., Jing, M. and Ma, T. T. B., 2015. Thunder crystal: a novel crowdsourcing-based content distribution platform. *ACM Workshop on Network and*

Operating Systems Support for Digital Audio and Video - NOSSDAV 15. doi:10.1145/2736084.2736085. 56, 60

Chen, Y.C., Lim, Y., Gibbens, R. J., Nahum, E. M., Khalili, R. and Towsley, D., 2013. A measurement-based study of MultiPath TCP performance over wireless networks. *Conference on Internet measurement conference - IMC 13*. doi:10.1145/2504730.2504751. 41, 45

Cherif, W., Fablet, Y., Nassor, E., Taquet, J. and Fujimori, Y., 2015. DASH fast start using HTTP/2. *ACM Workshop on Network and Operating Systems Support for Digital Audio and Video - NOSSDAV*. doi:10.1145/2736084.2736088. 35, 39

Chien, Y.L., Lin, K. C. and Chen, M., 2015. Machine learning based rate adaptation with elastic feature selection for HTTP-based streaming. *IEEE International Conference on Multimedia and Expo (ICME)*. doi:10.1109/icme.2015.7177418. 22, 31

Chiu, D. and Jain, R., 1989. Analysis of the increase and decrease algorithms for congestion avoidance in computer networks. *Computer Networks and ISDN Systems*, 17(1):1–14. doi:10.1016/0169-7552(89)90019-6. 26

Chromium, 2010. SPDY: An experimental protocol for a faster web. URL https://www.chromium.org/spdy 39

Chromium, 2017. QUIC, a multiplexed stream transport over UDP. URL https://www.chromium.org/quic 39

Cicco, L. De, Caldaralo, V., Palmisano, V. and Mascolo, S., 2013. ELASTIC: A Client-Side Controller for Dynamic Adaptive Streaming over HTTP (DASH). *International Packet Video Workshop*. doi:10.1109/pv.2013.6691442. 22, 30

Cicco, L. De and Mascolo, S., 2010. An Experimental Investigation of the Akamai Adaptive Video Streaming. *HCI in Work and Learning, Life and Leisure Lecture Notes in Computer Science*, page 447–464. doi:10.1007/978-3-642-16607-5_31. 28

Cisco, 2016. Cisco Visual Networking Index. http://www.cisco.com/c/dam/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.pdf. Online; Accessed August 2017. 1, 50, 63

Claeys, M., Latre, S., Famaey, J. and Turck, F D., 2014a. Design and Evaluation of a Self-Learning HTTP Adaptive Video Streaming Client. *IEEE Communications Letters*, 18(4):716–719. doi:10.1109/lcomm.2014.020414.132649. 22, 31, 32

Claeys, M., Latre, S., Famaey, J. and Turck, F D., 2014b. Design of a q-learning-based client quality selection algorithm for http adaptive video streaming. *IEEE Communications Letters*, 18(4):716–719. doi:10.1109/lcomm.2014.020414.132649. 22, 31, 32

CloudStream, 2017. CloudStream. http://cloudstreamglobal.com. Online; Accessed August 2017. 2

Cofano, G., Cicco, L. De and Mascolo, S., 2014. A Control Architecture for Massive Adaptive Video Streaming Delivery. *Workshop on Design, Quality and Deployment of Adaptive Video Streaming - VideoNext*. doi:10.1145/2676652.2676655. 22, 23, 29

Cofano, G., Cicco, L. De, Zinner, T., Nguyen-Ngoc, A., Tran-Gia, P. and Mascolo, S., 2016. Design and experimental evaluation of network-assisted strategies for HTTP adaptive streaming. *Proceedings of the 7th International Conference on Multimedia Systems - MMSys 16*. doi:10.1145/2910017.2910597. 35, 38

Corbillon, X. *et al.*, 2016. Cross-layer scheduler for video streaming over MPTCP. *ACM International Conference on Multimedia Systems (MMSys)*. doi:10.1145/2910017.2910594. 41, 44, 45

Cortes, C. and Vapnik, V., 1995. Support-vector networks. *Machine Learning*, 20(3):273–297. doi:10.1007/bf00994018. 25

Cranley, N., Perry, P. and Murphy, L., 2006. User perception of adapting video quality. *International Journal of Human-Computer Studies*, 64(8):637–647. doi:10.1016/j.ijhcs.2005.12.002. 26

DASH-IF, 2016. DASH-IF Position Paper: Server and Network Assisted DASH (SAND). URL http://dashif.org/wp-content/uploads/2017/01/SAND-Whitepaper-Dec13-final.pdf 37

DASH-IF, 2016. DASH Industry Forum - Software . http://dashif.org/software/. Online; Accessed August 2017. 18

DASH-IF, 2017a. DASH-IF - DASH Industry Forum. https://dashif.org. Online; Accessed August 2017. 18, 79

DASH-IF, 2017b. Guidelines - Completed DASH-IF Interoperability Documents. http://dashif.org/guidelines/. Online; Accessed August 2017. 18

DASH-IF, 2017c. Industry-Forum/dash.js Adaptive Video Player. https://github.com/dash-industry-forum/dash.js. Online; Accessed August 2017. 18, 79

DASH-IF, 2017d. Test Assets Database. http://testassets.dashif.org/. Online; Accessed August 2017. 18

DASH-IF, DASH-Industry Forum, 2014. Guidelines for implementation: DASH-AVC/264 Test cases and Vectors. http://dashif.org/wp-content/uploads/2015/04/DASH-AVC-264-Test-Vectors-v09-CommunityReview.pdf. Online; Accessed August 2017. 18, 30, 82

De Cicco, L., Mascolo, S. and Palmisano, V., 2011. Feedback control for adaptive live video streaming. In *ACM conference on Multimedia systems - MMSys*. 22, 28, 35, 36

Deering, S. E. and Cheriton, D. R., 1990. Multicast routing in datagram internetworks and extended LANs. *ACM Transactions on Computer Systems*, 8(2):85–110. doi:10.1145/78952.78953. 11

DELTA, 2015. Network-Aware Delivery Clouds for User Centric Media Events - DELTA European EUROSTAR Project.
URL http://www.mosys.unic.ac.cy/delta/ 6, 7, 115, 148

Deng, H. and Xu, J., 2013. CorePeer: A P2P Mechanism for Hybrid CDN-P2P Architecture. *Web-Age Information Management Lecture Notes in Computer Science*, page 278–286. doi:10.1007/978-3-642-39527-7_28. 51

Deng, S., Netravali, R., Sivaraman, A and Balakrishnan, H., 2014. WiFi, LTE, or Both?: Measuring Multi-Homed Wireless Internet Performance. *Conference on Internet Measurement Conference - IMC 14*. doi:10.1145/2663716.2663727. 41, 45

Diallo, M. Tourad, Moustafa, H., Afifi, H. and Marechal, N., 2013. Adaptation of audiovisual contents and their delivery means. *Communications of the ACM*, 56(11):86–93. doi:10.1145/2500726. 14, 18

Diot, C., Levine, B.n., Lyles, B., Kassem, H. and Balensiefen, D., 2000. Deployment issues for the IP multicast service and architecture. *IEEE Network*, 14(1):78–88. doi:10.1109/65.819174. 11

DISEDAN, 2016. DISEDAN - Service and User-Based Distributed Selection of Content Streaming Source and Dual Adaptation".
URL http://www.chistera.eu/projects/disedan 6, 7, 115, 148

DVB, 2016. Digital Video Broadcasting (DVB);MPEG-DASH Profile for Transport of ISO BMFF Based DVB Services over IP Based Networks.
URL https://www.dvb.org/resources/public/standards/a168_dvb-dash.pdf 18

Eberhard, M., Szkaliczki, T., Hellwagner, H., Szobonya, L. and Timmerer, C., 2010. Knapsack problem-based piece-picking algorithms for layered content in peer-to-peer networks. *Proceedings of the 2010 ACM workshop on Advanced video streaming techniques for peer-to-peer networks and social networking - AVSTP2P 10*. doi:10.1145/1877891.1877908. 52

Essaili, A. El, Schroeder, D., Staehle, D., Shehada, M., Kellerer, W. and Steinbach, E., 2013. Quality-of-experience driven adaptive HTTP media delivery. *2013 IEEE International Conference on Communications (ICC)*. doi:10.1109/icc.2013.6654905. 20

Evensen, K., Kaspar, D., Griwodz, C., Halvorsen, P., Hansen, A. F. and Engelstad, P., 2012. Using bandwidth aggregation to improve the performance of quality-adaptive streaming. *Signal Processing: Image Communication*, 27(4):312–328. doi:10.1016/j.image.2011.10.007. 41, 42, 47

Evensen, K., Kupka, T., Kaspar, D., Halvorsen, P. and Griwodz, C., 2010. Quality-adaptive scheduling for live streaming over multiple access networks. *International workshop on Network and operating systems support for digital audio and video - NOSSDAV*. doi:10.1145/1806565.1806573. 41, 46

Evensen, K., Petlund, A., Riiser, H., Vigmostad, P., Kaspar, D., Griwodz, C. and Halvorsen, P., 2011. Demo: Quality-Adaptive Video Streaming With Dynamic Bandwidth Aggregation on Roaming, Multi-Homed Clients. *International conference on Mobile systems, applications, and services - MobiSys 11*. doi:10.1145/1999995.2000032. 41, 46

Fairhurst, G., Trammell, B. and Kaijhlewind, M., 2017. Services Provided by IETF Transport Protocols and Congestion Control Mechanisms.
URL https://tools.ietf.org/html/rfc8095 41, 45

Famaey, J., Latré, S., Bouten, N., Van de Meerssche, W., De Vleeschauwer, B., Van Leekwijck, W. and De Turck, F., 2013. On the merits of SVC-based HTTP Adaptive Streaming. *2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*, pages 419–426. 22, 33, 34

Ford, A., Raicu, C., Handley, L. and Bonaventure, O., 2012. TCP Extensions for Multipath Operation with Multiple Addresses.
URL https://tools.ietf.org/html/rfc6824 44

Frank, B., Poese, I., Lin, Y., Smaragdakis, G., Feldmann, A., Maggs, B., Rake, J., Uhlig, S. and Weber, R., 2013. Pushing CDN-ISP collaboration to the limit. *ACM*

SIGCOMM Computer Communication Review, 43(3):34. doi:10.1145/2500098.2500103. 50

Friedman, T., Caceres, R. and Clark, A., 2003. RTP Control Protocol Extended Reports (RTCP XR).
URL https://tools.ietf.org/html/rfc3611 11

Fuente, Y. S., Schierl, T., Hellge, C., Wiegand, T., Hong, D., Vleeschauwer, D. D., Leekwijck, W. V. and Louédec, Y. L., 2011. iDASH. *ACM conference on Multimedia systems - MMSys*. doi:10.1145/1943552.1943586. 22, 32, 34

Gao, G., Wen, Y., Zhang, W. and Hu, H., 2015. Cost-efficient and QoS-aware content management in media cloud: Implementation and evaluation. *IEEE International Conference on Communications (ICC)*. doi:10.1109/icc.2015.7249422. 49

Garcia, M.-N, Simone, F. De, Tavakoli, S., Staelens, N., Egger, S., Brunnstrom, K. and Raake, A., 2014. Quality of experience and HTTP adaptive streaming: A review of subjective studies. *International Workshop on Quality of Multimedia Experience (QoMEX)*. doi:10.1109/qomex.2014.6982310. 14, 43

Gautam, N., Petander, H. and Noel, J., 2013. A comparison of the cost and energy efficiency of prefetching and streaming of mobile video. *Workshop on Mobile Video - MoVid*. doi:10.1145/2457413.2457416. 41, 42

Georgopoulos, P., Elkhatib, Y., Broadbent, M., Mu, M. and Race, N., 2013. Towards network-wide QoE fairness using openflow-assisted adaptive video streaming. *ACM SIGCOMM workshop on Future human-centric multimedia networking - FhMN*. doi:10.1145/2491172.2491181. 26

Gouache, S., Bichot, G., Bsila, A. and Howson, C., 2011. Distributed and adaptive HTTP streaming. *IEEE International Conference on Multimedia and Expo*. doi:10.1109/icme.2011.6012028. 22, 24, 41, 48

Grafl, M., Timmerer, C., Hellwagner, H., Cherif, W. and Ksentini, A., 2013. Evaluation of hybrid Scalable Video Coding for HTTP-based adaptive media streaming with high-definition content. *International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM)*. doi:10.1109/wowmom.2013.6583506. 22, 33

Gramatikov, S. and Jaureguizar, F., 2015. Modelling and analysis of non-cooperative peer-assisted VoD streaming in managed networks. *Multimedia Tools and Applications*, 75(8):4321–4348. doi:10.1007/s11042-015-2477-9. 56, 60

Grizzly, 2017. Grizzly Project. https://javaee.github.io/grizzly/. [Online]. 90

Ha, D., Silverton, T. and Fourmeaux, O., 2011. A novel Hybrid CDN-P2P mechanism For effective real-time media streaming.
URL https://www-npa.lip6.fr/~fourmaux/Stages/HA.ACM_Rapport.pdf 54, 57, 59, 60

Halepkidis, E., Pentikousis, K., Denazis, S., Salim, J. H., Meyer, D. and Koufopavlou, O., 2015. Software-Defined Networking (SDN): Layers and Architecture Terminology.
URL https://tools.ietf.org/html/rfc7426 35, 38

Hamilton, R., Iyengar, J., Swett, I. and Wilk, A., 2016. QUIC: A UDP-Based Secure and Reliable Transport for HTTP/2.
URL https://tools.ietf.org/html/draft-hamilton-early-deployment-quic-00 39

Han, B., Qian, F., Ji, L. and Gopalakrishnan, V., 2016. Mp-Dash. *International on Conference on emerging Networking EXperiments and Technologies - CoNEXT*. doi: 10.1145/2999572.2999606. 41, 45

Handley, M. and Jacobson, V., 1998. SDP: Session Description Protocol.
URL https://tools.ietf.org/html/rfc2327 11

Havey, D., Chertov, R. and Almeroth, K., 2012. Receiver driven rate adaptation for wireless multimedia applications. *Conference on Multimedia Systems - MMSys*. doi: 10.1145/2155555.2155582. 35

Hesmans, B., Duchene, F., Paasch, C., Detal, G. and Bonaventure, O., 2013. Are TCP extensions middlebox-proof? *Workshop on Hot topics in middleboxes and network function virtualization - HotMiddlebox 13*. doi:10.1145/2535828.2535830. 41, 45

Honda, M., Nishida, Y., Raiciu, C., Greenhalgh, A., Handley, M/ and Tokuda, H., 2011. Is it still possible to extend TCP? *ACM SIGCOMM conference on Internet measurement conference - IMC*. doi:10.1145/2068816.2068834. 41, 45

Hooft, J. V.D., Petrangeli, S., Claeys, M., Famaey, J. and Turck, F. De, 2015. A learning-based algorithm for improved bandwidth-awareness of adaptive streaming clients. *IFIP/IEEE International Symposium on Integrated Network Management (IM)*. doi:10.1109/inm.2015.7140285. 22, 32

Hossfeld, T., Egger, S., Schatz, R., Fiedler, M., Masuch, K. and Lorentzen, C., 2012. Initial delay vs. interruptions: Between the devil and the deep blue sea. In *Quality of Multimedia Experience (QoMEX), 2012 Fourth International Workshop on*, pages 1–6. IEEE. 19, 20

Hossfeld, T., Schatz, R., Biersack, E. and Plissonneau, L., 2013. Internet Video Delivery in Youtube: From Traffic Measurements to Quality of Experience. In *Data Traffic Monitoring and Analysis*, pages 264—-301. Springer. 2

Hossfeld, T., Seufert, M., Hirth, M., Zinner, T., Tran-Gia, P. and Schatz, R., 2011. Quantification of YouTube QoE via crowdsourcing. In *Multimedia (ISM), 2011 IEEE International Symposium on*, pages 494–499. IEEE. 19

Hossfeld, T., Seufert, M., Sieber, C. and Zinner, T., 2014. Assessing effect sizes of influence factors towards a QoE model for HTTP adaptive streaming. *2014 Sixth International Workshop on Quality of Multimedia Experience (QoMEX)*. doi:10.1109/qomex.2014.6982305. 20

Hossfeld, T., Strohmeier, D., Raake, A. and Schatz, R., 2013. Pippi Longstocking calculus for temporal stimuli pattern on YouTube QoE: 1+ 1= 3 and 1· 4≠ 4· 1. In *Proceedings of the 5th Workshop on Mobile Video*, pages 37–42. ACM. 20

Houdaille, R. and Gouache, S., 2012. Shaping HTTP adaptive streams for a better user experience. *Multimedia Systems Conference -MMSys-*. doi:10.1145/2155555.2155557. 22

Hu, C., Chen, M., Xing, C. and Xu, B., 2012. EUE principle of resource scheduling for live streaming systems underlying CDN-P2P hybrid architecture. *Peer-to-Peer Networking and Applications*, 5(4):312–322. doi:10.1007/s12083-012-0140-z. 54, 57, 60

Huang, C., Li, J. and Ross, K. W., 2007. Can internet video-on-demand be profitable? *ACM SIGCOMM Computer Communication Review*, 37(4):133. doi:10.1145/1282427.1282396. 50, 54, 56

Huang, C., Wang, A., Li, J. and Ross, K. W, 2008. Understanding hybrid CDN-P2P: why limelight needs its own Red Swoosh. *Proceedings of the 18th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, pages 75–80. 53

Huang, T.-Y., Handigol, N., Heller, B., Mckeown, N. and Johari, R., 2012. Confused, timid, and unstable: picking a video streaming rate is hard. *ACM conference on Internet measurement conference - IMC*. doi:10.1145/2398776.2398800. 22, 23, 26, 28

Huang, T-Y., Johari, R., McKeown, N., Trunnell, M. and Watson, M., 2014. A Buffer-based Approach to Rate Adaptation: Evidence from a Large Video Streaming Service. In *ACM Conference on SIGCOMM*, SIGCOMM '14, pages 187–198. ACM, New York, NY, USA. ISBN 978-1-4503-2836-4. doi:10.1145/2619239.2626296. 22, 28

Huang, T.Y., Johari, R. and Mckeown, N., 2013. Downtown abbey without the hiccups. *ACM SIGCOMM workshop on Future human-centric multimedia networking*. doi:10.1145/2491172.2491179. 21, 22, 28, 41, 42

Huang, Y., 2008. Challenges, design and analysis of a large-scale P2P-VoD system. In *ACM SIGCOMM*, pages 375–388. 2, 51, 52

Huysegems, R., Hooft, J. V., Bostoen, T., Alface, P. R., Petrangeli, S., Wauters, T. and Turck, F. D., 2015. HTTP/2-Based Methods to Improve the Live Experience of Adaptive Streaming. *ACM international conference on Multimedia - MM*. doi:10.1145/2733373.2806264. 35, 39

Huysegems, R., Vleeschauwer, B. D., Wu, T. and Leekwijck, W. V., 2012. SVC-Based HTTP Adaptive Streaming. *Bell Labs Technical Journal*, 16(4):25–41. doi:10.1002/bltj.20532. 32, 34

ISO/IEC, 2012. Information technology – Coding of audio-visual objects – Part 12: ISO base media file format.
URL https://www.iso.org/standard/61988.html 15

ISO/IEC, 2017a. ISO/IEC 13818-1:2015 - Information technology – Generic coding of moving pictures and associated audio information – Part 1: Systems.
URL https://www.iso.org/standard/67331.html 15

ISO/IEC, 2017b. ISO/IEC 14496-12:2015 - Information technology – Coding of audio-visual objects – Part 12: ISO base media file format.
URL https://www.iso.org/standard/68960.html 15

ISO/IEC, 2017c. ISO/IEC 23009-5:2017 - Information technology – Dynamic adaptive streaming over HTTP (DASH) – Part 5: Server and network assisted DASH (SAND).
URL https://www.iso.org/standard/69079.html 37

ISO/IEC MPEG, 2014. ISO/IEC 23009-1:2014 Dynamic adaptive streaming over HTTP. https://www.iso.org/standard/65274.html. Online; Accessed August 2017. 3, 14, 18, 73, 157

Jain, M. and Dovrolis, C., 2004. Ten fallacies and pitfalls on end-to-end available bandwidth estimation. *ACM SIGCOMM conference on Internet measurement - IMC*. doi:10.1145/1028788.1028825. 23, 25

Jersey, 2017. Jersey: RESTful Web Services in Java. https://jersey.github.io. [Online]. 90

Jiang, J., Sekar, V., Milner, H., Shepherd, D., Stoica, I. and Zhang, H., 2016. CFA: A Practical Prediction System for Video QoE Optimization. *Proceedings of the 13th Usenix Conference on Networked Systems Design and Implementation - NSDI*, pages 137–150.
URL http://dl.acm.org/citation.cfm?id=2930611.2930621 22, 27, 41

Jiang, J., Sekar, V. and Zhang, H., 2012. Improving fairness, efficiency, and stability in HTTP-based adaptive video streaming with FESTIVE. *International conference on Emerging networking experiments and technologies - CoNEXT*. doi:10.1145/2413176.2413189. 20

Jiang, J., Sekar, V. and Zhang, H., 2014. Improving Fairness, Efficiency, and Stability in HTTP-Based Adaptive Video Streaming With Festive. *IEEE/ACM Transactions on Networking*, 22(1):326–340. doi:10.1109/tnet.2013.2291681. 22, 23, 24, 27, 41, 42, 43

Jin, Y., Wen, Y. and Guan, K., 2016. Toward Cost-Efficient Content Placement in Media Cloud: Modeling and Analysis. *IEEE Transactions on Multimedia*, 18(5):807–819. doi:10.1109/tmm.2016.2537199. 49

Johansen, D., Endestad, T., Riiser, H., Griwidz, C., Halvorsen, P., Johansen, H., Aarflot, T., Hurley, J., Kvalnes, A., Gurrin, C. and et al., 2009. Davvi: A prototype for the next generation multimedia entertainment platform. *ACM international conference on Multimedia - MM*. doi:10.1145/1631272.1631482. 41, 42, 46

Joseph, V. and Veciana, G. De, 2014. NOVA: QoE-driven optimization of DASH-based video delivery in networks. *IEEE INFOCOM*. doi:10.1109/infocom.2014.6847927. 29

Jurca, D., Chakareski, J., Wagner, J. and Frossard, P., 2007. Enabling adaptive video streaming in P2P systems [Peer-to-Peer Multimedia Streaming]. *IEEE Communications Magazine*, 45(6):108–114. doi:10.1109/mcom.2007.374427. 52, 60

Jurca, D. and Frossard, P., 2007. Video Packet Selection and Scheduling for Multipath Streaming. *IEEE Transactions on Multimedia*, 9(3):629–641. doi:10.1109/tmm.2006.888017. 41, 45

Kalva, H., Adzic, V. and Furht, B., 2012. Comparing MPEG AVC and SVC for adaptive HTTP streaming. *IEEE International Conference on Consumer Electronics (ICCE)*. doi:10.1109/icce.2012.6161787. 34

Karamshuk, D., Sastry, N., Secker, A. and Chandaria, J., 2015. ISP-friendly peer-assisted on-demand streaming of long duration content in BBC iPlayer. *2015 IEEE Conference on Computer Communications (INFOCOM)*. doi:10.1109/infocom.2015.7218393. 50, 54

Kaspar, D., Evensen, K., Engelstad, P. and Hansen, A. F., 2010a. Using HTTP Pipelining to Improve Progressive Download over Multiple Heterogeneous Interfaces. *International Conference on Communications*. doi:10.1109/icc.2010.5502574. 41, 46

Kaspar, D., Evensen, K., Engelstad, P., Hansen, A. F., Halvorsen, P. l and Griwodz, C., 2010b. Enhancing Video-on-Demand Playout over Multiple Heterogeneous Access Networks. *Consumer Communications and Networking Conference*. doi:10.1109/ccnc.2010.5421846. 41, 46

Kaune, S., Rumin, R. C., Tyson, G., Mauthe, A., Guerrero, C. and Steinmetz, R., 2010. Unraveling BitTorrents File Unavailability: Measurements and Analysis. *2010 IEEE Tenth International Conference on Peer-to-Peer Computing (P2P)*. doi:10.1109/p2p.2010.5569991. 53

Kazemi, M. *et al.*, 2013. A review of multiple description coding techniques for error-resilient video delivery. *Springer Multimedia Systems*, 20(3):283–309. 65, 67, 73, 158

Khan Pathan, A. and Buyya, R., 2010. A Taxonomy and Survey of CDNs. *Content Delivery Networks Lecture Notes Electrical Engineering*. 64

Kim, T. N., Jeon, S. and Kim, Y., 2011. A CDN-P2P hybrid architecture with content/location awareness for live streaming service networks. *2011 IEEE 15th International Symposium on Consumer Electronics (ISCE)*. doi:10.1109/isce.2011.5973865. 54, 55, 58, 60

Kim, Y., Kim, Y., Yoon, H. and Yeom, I., 2015. Peer-assisted multimedia delivery using periodic multicast. *Information Sciences*, 298:425–446. doi:10.1016/j.ins.2014.11.033. 53

Kleinrouweler, J. W., Cabrero, S. and Cesar, P., 2016. Delivering Stable High-quality Video: An SDN Architecture with DASH Assisting Network Elements. *International Conference on Multimedia Systems - MMSys*. doi:10.1145/2910017.2910599. 35, 38

Kua, J., Armitage, G. and Branch, P., 2017. A Survey of Rate Adaptation Techniques for Dynamic Adaptive Streaming Over HTTP. *IEEE Communications Surveys & Tutorials*, 19(3):1842–1866. doi:10.1109/comst.2017.2685630. 14

Kuhn, N. *et al.*, 2014. DAPS: Intelligent delay-aware packet scheduling for multipath transport. *IEEE International Conference on Communications (ICC)*. doi:10.1109/icc.2014.6883488. 41, 44

Kupka, T., Halvorsen, P. and Griwodz, C., 2011. An evaluation of live adaptive HTTP segment streaming request strategies. *Conference on Local Computer Networks - LCN*. doi:10.1109/lcn.2011.6115524. 43

Kupka, T., Halvorsen, P. and Griwodz, C., 2012. Performance of on-off traffic stemming from live adaptive segmented HTTP video streaming. *IEEE Conference on Local Computer Networks*. doi:10.1109/lcn.2012.6423654. 41, 42

Kurosaka, T. and Bandai, M., 2015. MPTCP with multiple ACKs for heterogeneous communication links. *IEEE Consumer Communications and Networking Conference (CCNC)*. doi:10.1109/ccnc.2015.7158048. 41, 44

Kuschnig, R., Kofler, I. and Hellwagner, H., 2010. Improving Internet Video Streaming Performance by Parallel TCP-Based Request-Response Streams. *IEEE Consumer Communications and Networking Conference*. doi:10.1109/ccnc.2010.5421815. 41, 47

Lederer, S., Müller, C. and Timmerer, C., 2012. Towards peer-assisted dynamic adaptive streaming over HTTP. *Packet Video Workshop (PV), 2012 19th International*, pages 161–166. 59, 60

Lei, J., Shi, L. and Fu, X., 2009. An experimental analysis of Joost peer-to-peer VoD service. *Peer-to-Peer Networking and Applications*, 3(4):351–362. doi:10.1007/s12083-009-0063-5. 52

Li, J., Cui, Y. and Chang, B., 2007. PeerStreaming: design and implementation of an on-demand distributed streaming system with digital rights management capabilities. *Multimedia Systems*, 13(3):173–190. doi:10.1007/s00530-006-0073-6. 52, 60

Li, X., Dong, M., Ma, Z. and Fernandes, F. C.a., 2012a. GreenTube: Power optimization for mobile video streaming via dynamic cache managemen. *ACM international conference on Multimedia - MM*. doi:10.1145/2393347.2393390. 41, 42

Li, Z., Shen, H., Wang, H., Liu, G. and Li, J., 2012b. SocialTube: P2P-assisted video sharing in online social networks. *2012 Proceedings IEEE INFOCOM*. doi:10.1109/infcom.2012.6195721. 54

Li, Z., Shen, H., Wang, H., Liu, G. and Li, J., 2012c. SocialTube: P2P-assisted video sharing in online social networks. *2012 Proceedings IEEE INFOCOM*. doi:10.1109/infcom.2012.6195721. 54

Li, Z., Zhu, X., Gahm, J., Pan, R., Hu, H., Begen, A. C. and Oran, D., 2014. Probe and Adapt: Rate Adaptation for HTTP Video Streaming At Scale. *IEEE Journal on Selected*

Areas in Communications (JSAC), 32(4):719–733. doi:10.1109/jsac.2014.140405. 22, 25, 29, 30

Limelight, 2017. Limelight Networks: Content Delivery Network (CDN) Services. URL https://www.limelight.com/delivery/ 50

Lin, J. and Hwang, W., 2011. Efficient Scalable Video Coding Based on Matching Pursuits. *Effective Video Coding for Multimedia Applications*. doi:10.5772/14924. 32

Liu, C., Bouazizi, I. and Gabbouj, M., 2011a. Parallel Adaptive HTTP Media Streaming. *International Conference on Computer Communications and Networks (ICCCN)*. doi:10.1109/icccn.2011.6005910. 48

Liu, C., Bouazizi, I. and Gabbouj, M., 2011b. Rate adaptation for adaptive HTTP streaming. *ACM conference on Multimedia systems - MMSys*. doi:10.1145/1943552.1943575. 22, 27, 41, 48

Liu, C., Bouazizi, I., Hannuksela, M. M. and Gabbouj, M., 2012a. Rate adaptation for dynamic adaptive streaming over HTTP in content distribution network. *Signal Processing: Image Communication*, 27(4):288–311. doi:10.1016/j.image.2011.10.001. 40, 41, 48

Liu, H. H., Wang, Y., Yang, Y. Richard, Wang, H. and Tian, C., 2012b. Optimizing cost and performance for content multihoming. *ACM SIGCOMM Computer Communication Review*, 42(4):371. doi:10.1145/2377677.2377753. 17

Liu, X., Dobrian, F., Milner, H., Jiang, J., Sekar, V., Stoica, I. and Zhang, H., 2012c. A case for a coordinated internet video control plane. *ACM SIGCOMM Computer Communication Review*, 42(4):359. doi:10.1145/2377677.2377752. 17, 23, 50

Liu, Y., Guo, Y. and Liang, C., 2008. A survey on peer-to-peer video streaming systems. *Peer-to-Peer Networking and Applications*, 1(1):18–28. doi:10.1007/s12083-007-0006-y. 52, 57

Liu, Z., Shen, Y., Ross, K. W, Panwar, S. S and Wang, Y., 2009. LayerP2P: using layered video chunks in P2P live streaming. *IEEE Transactions on Multimedia*, 11(7):1340–1352. 52, 60

Lu, M-T, Wu, J-C, Peng, K-J, Huang, P., Yao, J. and Chen, H., 2007. Design and Evaluation of a P2P IPTV System for Heterogeneous Networks. *IEEE Transactions on Multimedia*, 9(8):1568–1579. doi:10.1109/tmm.2007.907456. 67

Lu, Z., Wang, Y., Y., Y. Richard *et al.*, 2012. An Analysis and Comparison of CDN-P2P-hybrid Content Delivery System and Model. *JCM*, 7(3):232–245. 53

Lu, Z. H., Gao, X. H., Huang, S. J. and Huang, Y., 2011. Scalable and Reliable Live Streaming Service through Coordinating CDN and P2P. *International Conference on Parallel and Distributed Systems*. doi:10.1109/icpads.2011.113. 54, 57, 59, 60

Lua, E. Keong, Crowcroft, J., Pias, M., Sharma, R. and Lim, S., 2005. A survey and comparison of peer-to-peer overlay network schemes. *IEEE Communications Surveys & Tutorials*, 7(2):72–93. doi:10.1109/comst.2005.1610546. 52

Ma, M., Wang, Z., Su, K. and Sun, L., 2016. Understanding the Power of Smartrouter-Based Peer CDN for Video Streaming. *2016 25th International Conference on Computer Communication and Networks (ICCCN)*. doi:10.1109/icccn.2016.7568594. 56, 60

Major, R. and Hurst, M., 2014. Patent US8868772 - Apparatus, system, and method for adaptive-rate shifting of streaming content.
URL https://www.google.com/patents/US8868772 17

Mansy, A., Ammar, M., Chandrashekar, J. and Sheth, A., 2013. Characterizing Client Behavior of Commercial Mobile Video Streaming Services. *Workshop on Mobile Video Delivery - MoViD*. doi:10.1145/2594449.2579469. 35, 37

Markakis, E., Negru, D., Bruneau-Queyreix, J., Pallis, E., Mastorakis, G. and Mavromoustakis, C. X., 2017. A P2P Home-Box Overlay for Efficient Content Distribution. *Emerging Innovations in Wireless Networks and Broadband Technologies Advances in Wireless Technologies and Telecommunication*, page 199–220. doi:10.4018/978-1-4666-9941-0.ch009. 57, 60

Medjiah, S., Ahmed, T. and Boutaba, R., 2014. Avoiding quality bottlenecks in P2P adaptive streaming. *IEEE Journal on Selected Areas in Communications*, 32(4):734–745. 52, 60

Menasche, D. S., Rocha, A. A.a., Li, B., Towsley, D. and Venkataramani, A., 2009. Content availability and bundling in swarming systems. *Proceedings of the 5th international conference on Emerging networking experiments and technologies - CoNEXT 09*. doi:10.1145/1658939.1658954. 53

Menkovski, V. and Liotta, A., 2013. Intelligent control for adaptive video streaming. *IEEE International Conference on Consumer Electronics (ICCE)*. doi:10.1109/icce.2013.6486825. 22, 31

Merani, M. and Natali, L., 2016. Adaptive streaming in P2P live video systems: A distributed rate control approach. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 12(3):46. 59, 60, 134, 135, 163

Microsoft, 2017. Smooth streaming. http://www.microsoft.com/silverlight/smoothstreaming/. [Online]. 3, 17, 24, 41, 42

Miller, K., Bethanabhotla, D., Caire, G. and Wolisz, A., 2015. A Control-Theoretic Approach to Adaptive Video Streaming in Dense Wireless Networks. *IEEE Transactions on Multimedia*, 17(8):1309–1322. doi:10.1109/tmm.2015.2441002. 22, 29

Miller, K., Quacchio, E., Gennari, G. and Wolisz, A., 2012. Adaptation algorithm for adaptive streaming over HTTP. *International Packet Video Workshop (PV)*. doi:10.1109/pv.2012.6229732. 22, 23, 28

Mirza, M., Sommers, J., Barford, P. and Zhu, X., 2007. A machine learning approach to TCP throughput prediction. *ACM SIGMETRICS Performance Evaluation Review*, 35(1):97. doi:10.1145/1269899.1254894. 25

Mok, R. K. P., Luo, X., Chan, E. W. W. and Chang, R. K. C., 2012. QDASH: a QoE-aware DASH system. *Multimedia Systems Conference - MMSys*. 3, 20, 22, 26, 27, 35, 36

Moon, Y., Kim, J-N. and Youn, C-H., 2013. Churn-aware optimal layer scheduling scheme for scalable video distribution in super-peer overlay networks. *The Journal of Supercomputing*, 66(2):700–720. 52, 60

MS-Stream, 2017. MS-Stream Demonstration.
URL http://msstream.net 6, 149

Mu, M., Simpson, S., Farshad, A., Ni, Q. and Race, N., 2015. User-level fairness delivered: Network resource allocation for adaptive video streaming. *International Symposium on Quality of Service (IWQoS)*. doi:10.1109/iwqos.2015.7404718. 26

Mueller, C., Lederer, S. and Timmerer, C., 2012. A proxy effect analyis and fair adatpation algorithm for multiple competing Dynamic Adaptive Streaming over HTTP clients. *Visual Communications and Image Processing*. doi:10.1109/vcip.2012.6410799. 35, 36

Mueller, C., Lederer, S., Timmerer, C. and Hellwagner, H., 2013. Dynamic Adaptive Streaming over HTTP/2.0. *IEEE International Conference on Multimedia and Expo (ICME)*. doi:10.1109/icme.2013.6607498. 35, 39

Muller, C., Lederer, S. and Timmerer, C., 2012a. An evaluation of dynamic adaptive streaming over HTTP in vehicular environments. *Workshop on Mobile Video - MoVid 12*. doi:10.1145/2151677.2151686. 22, 33

Muller, C., Renzi, D., Lederer, S., Battista, S. and Timmerer, C., 2012b. Using Scalable Video Coding for Dynamic Adaptive Streaming over HTTP in mobile environments. *2012 Proceedings of the 20th European Signal Processing Conference (EUSIPCO)*, pages 2208–2212. 22, 33

Müller, C. and Timmerer, C., 2011. A VLC media player plugin enabling dynamic adaptive streaming over HTTP. *ACM international conference on Multimedia - MM*. doi:10.1145/2072298.2072429. 39

Netflix, 2017. Netflix. https://www.netflix.com. Online; Accessed August 2017. 2

Nguyen, T. and Cheung, S. S., 2005. Multimedia streaming using multiple TCP connections. *IEEE International Performance, Computing, and Communications Conference*. doi:10.1109/pccc.2005.1460558. 41, 47

Ni, P., Eg, R., Eichhorn, A., Griwodz, C. and Halvorsen, Paal, 2011. Flicker effects in adaptive video streaming to handheld devices. In *Proceedings of the 19th ACM International Conference on Multimedia*, pages 463–472. 20

Nicolas, Y., Wolff, D., Rossi, D. and Finamore, A., 2013. I Tube, YouTube, P2PTube: Assessing ISP benefits of peer-assisted caching of YouTube content. *IEEE P2P 2013 Proceedings*. doi:10.1109/p2p.2013.6688724. 56, 60

OIPF, 2014. Open IPTV Forum (OIPF) Release 2 Specification Volume 2a - HTTP Adaptive Streaming.
URL http://www.oipf.tv/web-spec/volume2a.html 17, 18

Oyman, O. and Singh, S., 2012. Quality of experience for HTTP adaptive streaming services. *IEEE Communications Magazine*, 50(4):20–27. doi:10.1109/mcom.2012.6178830. 20

Pantos, R., 2015. HTTP live streaming.
URL https://tools.ietf.org/html/draft-pantos-http-live-streaming-07 3, 17

Passarella, A., 2012. A survey on content-centric technologies for the current Internet: CDN and P2P solutions. *Computer Communications*, 35(1):1–32. doi:10.1016/j.comcom.2011.10.005. 57

Pathan, M. and Buyya, R., 2008. Performance models for peering Content Delivery Networks. *IEEE International Conference on Networks*. doi:10.1109/icon.2008.4772632. 49

Periscope, 2017. Periscope. https://www.pscp.tv. [Online]. 2

Peterson, R. and Sirer, E., 2009. Antfarm: Efficient content distribution with managed swarms. *NSDI*. 53

Popa, L., Ghodsi, A. and Stoica, I., 2010. HTTP as the narrow waist of the future internet. *ACM SIGCOMM Workshop on Hot Topics in Networks - Hotnets*. doi:10.1145/1868447.1868453. 17

PPLive, . PPLive. http://www.pptv.com/. Online; Accessed August 2017. 2

Priyadarshini, M. S. and Rekh, A S., 2016. Streaming Video with MultiPath TCP in Wireless Networks. *IEEE Transactions on Mobile Computing,*, 15(4):2345–2361. doi:https://doi.org/10.1109/TMC.2015.2497238. 46

Prosad, R., Davrolis, C., Murray, M. and Claffy, K.c., 2003. Bandwidth estimation: metrics, measurement techniques, and tools. *IEEE Network*, 17(6):27–35. doi:10.1109/mnet.2003.1248658. 23, 24

Pu, W., Zou, Z. and Chen, C. W., 2011. Dynamic Adaptive Streaming over HTTP from Multiple Content Distribution Servers. *IEEE Global Telecommunications Conference (GLOBECOM)*. 41, 48

Pu, W., Zou, Z. and Chen, C. W., 2012. Video adaptation proxy for wireless Dynamic Adaptive Streaming over HTTP. *International Packet Video Workshop (PV)*. doi:10.1109/pv.2012.6229745. 35, 37

Qiu, X., Liu, H., Li, D., Zhang, S., Ghosal, D. and Mukherjee, B., 2010. Optimizing HTTP-based Adaptive Video Streaming for wireless access networks. *IEEE International Conference on Broadband Network and Multimedia Technology (IC-BNMT)*. doi:10.1109/icbnmt.2010.5705208. 22, 24, 29

Qui, D., 2013. Modeling and performance analysis of BitTorrent-like peer-to-peer networks. *ACM SIGCOMM Computer Communication Review*.
URL http://dl.acm.org/citation.cfm?doid=1030194.1015508 53

Quinn, B. and Almeroth, K., 2001. IP Multicast Applications: Challenges and Solutions. *RFC 3170*. doi:10.17487/rfc3170. 11

Raiciu, C. *et al.*, 2012. How Hard Can It Be? Designing and Implementing a Deployable MPTCP. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. ISBN 978-931971-92-8. 41, 44

Ramamurthi, V. and Oyman, O., 2013. Link aware HTTP Adaptive Streaming for enhanced quality of experience. *IEEE Global Communications Conference (GLOBECOM)*. doi:10.1109/glocom.2013.6831314. 22, 25

Ramamurthi, V., Oyman, O. and Foerster, J., 2015. Using link awareness for HTTP Adaptive Streaming over changing wireless conditions. *International Conference on Computing, Networking and Communications (ICNC)*. doi:10.1109/iccnc.2015.7069436. 22, 25

Rao, A., Legout, A., Lim, Y., Towsley, D., Barakat, C. and Dabbous, W., 2011. Network characteristics of video streaming traffic. *COnference on emerging Networking EXperiments and Technologies on - CoNEXT 11*. doi:10.1145/2079296.2079321. 26, 41, 42

Ratnasamy, S., Francis, P., Handley, M., Karp, R. and Schenker, S., 2001. A scalable content-addressable network. *ACM SIGCOMM Computer Communication Review*, 31(4):161–172. doi:10.1145/964723.383072. 58, 60

Rejaie, R. and Kangasharju, J., 2001. Mocha. *International workshop on Network and operating systems support for digital audio and video - NOSSDAV*. doi:10.1145/378344.378345. 37

Roverso, R., Naiem, A., Reda, M., El-Beltagy, M., El-Ansary, S., Franzen, N. and Haridi, S., 2011. On the feasibility of centrally-coordinated Peer-to-Peer live streaming. *IEEE Consumer Communications and Networking Conference (CCNC)*. doi:10.1109/ccnc.2011.5766328. 53

Ruckert, J., Knierim, T. and Hausheer, D., 2014. Clubbing with the peers: A measurement study of BitTorrent live. *IEEE International Conference on Peer-to-Peer Computing*. doi:10.1109/p2p.2014.6934295. 50

Sahasrabuddhe, L. H. and Mukherjee, B., 2000. Multicast routing algorithms and protocols: a tutorial. *IEEE Network*, 14(1):90–102. doi:10.1109/65.819175. 11

Sandvine, 2015a. Global Internet Phenomena, Asia Pacific and Europe. https://www.sandvine.com/downloads/general/global-internet-phenomena/2015/global-internet-phenomena-report-apac-and-europe.pdf. [Online]. 1

Sandvine, 2015b. Global Internet Phenomena, Latin America and North America. https://www.sandvine.com/downloads/general/global-internet-phenomena/2015/global-internet-phenomena-report-latin-america-and-north-america.pdf. [Online]. 1

Sani, Y., Mauthe, A. and Edwards, C., 2017. Adaptive Bitrate Selection: A Survey. *IEEE Communications Surveys & Tutorials*, page 1–1. doi:10.1109/comst.2017.2725241. 14

Satoda, K., Yoshida, H., Ito, H. and Ozawa, K., 2012. Adaptive video pacing method based on the prediction of stochastic TCP throughput. *2012 IEEE Global Communications Conference (GLOBECOM)*. doi:10.1109/glocom.2012.6503400. 35, 36

Scellato, S., Mascolo, C., Musolesi, M. and Crowcroft, J., 2011. Track globally, deliver locally. *International conference on World wide web - WWW 11*. doi:10.1145/1963405.1963471. 49

Schulzrinne, H., Casner, S., Frederick, R. and Jacobson, V., 2003. RTP: A Transport Protocol for Real-Time Applications.
URL https://tools.ietf.org/html/rfc3550 10

Schulzrinne, H., Rao, A. and Lanphier, R., 1998. Real Time Streaming Protocol (RTSP).
URL https://tools.ietf.org/html/rfc2326 11

Schwarz, H., Marpe, D. and Wiegand, T., 2007. Overview of the Scalable Video Coding Extension of the H.264/AVC Standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(9):1103–1120. doi:10.1109/tcsvt.2007.905532. 10, 12, 32, 33, 34

Schwarz, H. and Wien, M., 2008. The Scalable Video Coding Extension of the H.264/AVC Standard [Standards in a Nutshell]. *IEEE Signal Processing Magazine*, 25(2):135–141. doi:10.1109/msp.2007.914712. 33

Seufert, M., Egger, S., Slanina, M., Zinner, T., Hobfeld, T. and Tran-Gia, P., 2015a. A Survey on Quality of Experience of HTTP Adaptive Streaming. *IEEE Communications Surveys & Tutorials*, 17(1):469–492. doi:10.1109/comst.2014.2360940. 20, 25

Seufert, M., Egger, S., Slanina, M., Zinner, T., Hobfeld, T. and Tran-Gia, P., 2015b. A survey on quality of experience of HTTP adaptive streaming. *IEEE Communications Surveys & Tutorials*, 17(1):469–492. 20

Sieber, C., Hossfeld, T., Zinner, T., Tran-Gia, P. and Timmerer, C., 2013. Implementation and user-centric comparison of a novel adaptation logic for DASH with SVC. *2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*, pages 1318–1323. 22, 33

Siekkinen, M., Hoque, M. A., Nurminen, J. K. and Aalto, M., 2013. Streaming over 3G and LTE: How to Save Smartphone Energy in Radio Access Network-friendly Way. *Workshop on Mobile Video - MoVid*. doi:10.1145/2457413.2457417. 35

Singh, A. *et al.*, 2012. Performance comparison of scheduling algorithms for multipath transfer. *IEEE Global Communications Conference (GLOBECOM)*. doi:10.1109/glocom.2012.6503517. 41, 45

Sobhani, A., Yassine, A. and Shirmohammadi, S., 2015. A fuzzy-based rate adaptation controller for DASH. *ACM Workshop on Network and Operating Systems Support for Digital Audio and Video - NOSSDAV*. doi:10.1145/2736084.2736090. 22, 31

Sodagar, I., 2011. The MPEG-DASH standard for multimedia streaming over the Internet. *IEEE MultiMedia*, 18(4):62–67. 3, 14, 157

Spiteri, K., Urgaonkar, R. and Sitaraman, R. K., 2016. BOLA: Near-optimal bitrate adaptation for online videos. *IEEE International Conference on Computer Communications - INFOCOM 2016*. doi:10.1109/infocom.2016.7524428. 22, 30

Stockhammer, T., 2011. Dynamic adaptive streaming over HTTP. *ACM conference on Multimedia systems - MMSys*. doi:10.1145/1943552.1943572. 18

Streamroot, 2017. Vivendi selects Streamroot for high quality OTT video delivery. https://www.streamroot.io/vivendi-selects-streamroot-high-quality-ott-video-delivery/. Online; Accessed August 2017. 2

Stutzbach, D. and Rejaie, R., 2006. Understanding churn in peer-to-peer networks. *ACM SIGCOMM on Internet measurement*. doi:10.1145/1177080.1177105. 51

Sun, Y., Yin, X., Jiang, J., Sekar, V., Lin, F., Wang, N., Liu, T. and Sinopoli, B., 2016. CS2P: Improving Video Bitrate Selection and Adaptation with Data-Driven Throughput Prediction. *ACM SIGCOMM Conference*. doi:10.1145/2934872.2934898. 22, 27

Tappayuthpijarn, K., Stockhammer, T. and Steinbach, E., 2011. HTTP-based scalable video streaming over mobile networks. *IEEE International Conference on Image Processing*. doi:10.1109/icip.2011.6116069. 22, 33

Thang, T., Ho, Q., Kang, J. and Pham, A., 2012a. Adaptive streaming of audiovisual content using MPEG DASH. *IEEE Transactions on Consumer Electronics*, 58(1):78–85. doi:10.1109/tce.2012.6170058. 22, 23, 24, 29

Thang, T. C., Le, H. T., Pham, A. T. and Bo., Y. M., 2014. An Evaluation of Bitrate Adaptation Methods for HTTP Live Streaming. In *IEEE Journal on Selected Areas in Communications*, tome 32, pages 693–705. 22, 31

Thang, T.-C., Pham, A. T., Nguyen, H. X., Cuong, P. L. and Kang, J. W., 2012b. Video streaming over HTTP with dynamic resource prediction. *International Conference on Communications and Electronics (ICCE)*. doi:10.1109/cce.2012.6315884. 22, 24

Tian, G. and Liu, Y., 2012. Towards Agile and Smooth Video Adaptation in Dynamic HTTP streaming. *Proceedings of the 8th international conference on Emerging networking experiments and technologies - CoNEXT*. 16, 22, 25, 29

Tian, G. and Liu, Y., 2013. On Adaptive HTTP Streaming to Mobile Devices. *International Packet Video Workshop*. doi:10.1109/pv.2013.6691450. 22, 24

Tian, G. and Liu, Y., 2016. Towards Agile and Smooth Video Adaptation in HTTP Adaptive Streaming. *IEEE/ACM Transactions on Networking*, 24(4). 16, 41, 48

Tian, G., Xu, Y., Liu, Y. and Ross, K., 2013. Mechanism design for dynamic P2P streaming. *2013 IEEE Thirteenth International Conference on Peer-to-Peer Computing (P2P)*, pages 1–10. 59, 60

Tillo, T., Baccaglini, E. and Olmo, G., 2010. Multiple Descriptions Based on Multirate Coding for JPEG 2000 and H.264/AVC. *IEEE Transactions on Image Processing*, 19(7):1756–1767. doi:10.1109/tip.2010.2045683. 67

Timmerer, C., 2016a. Advanced Transport Options for DASH: QUIC and HTTP/2 (Part II).
URL https://bitmovin.com/advanced-transport-options-dash-quic-http2-part-ii/
35, 40

Timmerer, C., 2016b. Transport options for MPEG-DASH over HTTP/2 and QUIC.
URL https://bitmovin.com/advanced-transport-options-dash-quic-http2/ 35, 40

Timmerer, C. and Begen, A.C., 2014. Over-the-Top Content Delivery: State of the Art and Challenges Ahead. In *ACM International Conference on Multimedia (Multimedia)*, pages 1231–1232. Orlando, USA. 2

Twitch, 2017. Twitch. https://www.twitch.tv. [Online]. 2

Unanue, I., Urteaga, I., Husemann, R., Del, J., Roesler, V., Rodriguez, A. and Sanchez, P., 2011. A Tutorial on H.264/SVC Scalable Video Coding and its Tradeoff between Quality, Coding Efficiency and Performance. *Recent Advances on Video Coding*. doi:10.5772/19227. 33

Varma, S., 2015. Flow Control for Video Applications. *Internet Congestion Control*, page 173–203. doi:10.1016/b978-0-12-803583-2.00006-2. 10, 11, 12

Vergados, D. J., Michalas, A., Sgora, A. and Vergados, D. D., 2014. A control-based algorithm for rate adaption in MPEG-DASH. *nternational Conference on Information, Intelligence, Systems and Applications*. doi:10.1109/iisa.2014.6878834. 22, 31

Villa, B. J., Heegaard, P. E. and Instefjord, A., 2012. Improving Fairness for Adaptive HTTP Video Streaming. *Springer, Information and Communication Technologies in Computer Science*, page 183–193. doi:10.1007/978-3-642-32808-4_17. 41, 42

Vriendt, J. De, Vleeschauwer, D. De and Robinson, D. C., 2014. QoE model for video delivered over an LTE network using HTTP adaptive streaming. *Bell Labs Technical Journal*, 18(4):45–62. doi:10.1002/bltj.21645. 20

Wang, B., Wei, W., Guo, Z. and Towsley, D., 2009. Multipath live streaming via TCP. *ACM Transactions on Multimedia Computing, Communications, and Applications*, 5(3):1–23. doi:10.1145/1556134.1556142. 41, 44, 45

Wei, S. and Swaminathan, V., 2013. Low Latency Live Video Streaming over HTTP 2.0. *Network and Operating System Support on Digital Audio and Video Workshop - NOSSDAV*. doi:10.1145/2597176.2578277. 35, 39

Wei, S. and Swaminathan, V., 2014. Cost effective video streaming using server push over HTTP 2.0. *International Workshop on Multimedia Signal Processing (MMSP)*. doi:10.1109/mmsp.2014.6958796. 39

Wendell, P. and Freedman, M. J., 2011. Going viral: flash crowds in an open CDN. *ACM SIGCOMM conference on Internet measurement conference - IMC 11*. doi:10.1145/2068816.2068867. 50

Wischik, D., Handley, M. and Braun, M. B., 2008. The resource pooling principle. *ACM SIGCOMM Computer Communication Review*, 38(5):47. doi:10.1145/1452335.1452342. 44

Wu, D., Liang, C., Liu, Y. and Ross, K., 2009. View-upload decoupling: A redesign of multi-channel P2P video systems. *IEEE INFOCOM*, pages 2726–2730. 126

Xiang, S., Cai, L. and Pan, J., 2012. Adaptive scalable video streaming in wireless networks. *Multimedia Systems Conference on - MMSys*. doi:10.1145/2155555.2155583. 22, 33

Xiao, X., Shi, Y., Gao, Y. and Zhang, Q., 2009. Layerp2p: A new data scheduling approach for layered streaming in heterogeneous networks. *IEEE INFOCOM*, pages 603–611. 52, 60

Xie, S., Li, B., Keung, G. and Zhang, X., 2007. Coolstreaming: Design, Theory, and Practice. *IEEE Transactions on Multimedia*, 9(8):1661–1671. doi:10.1109/tmm.2007.907469. 52, 60

Xiong, P., Shen, J., Wang, Q., Jayasinghe, D., Li, J. and Pu, C., 2012. NBS: A Network-Bandwidth-Aware Streaming Version Switcher for Mobile Streaming Applications under Fuzzy Logic Control. *IEEE International Conference on Mobile Services*. doi:10.1109/mobserv.2012.10. 22, 30, 31

Xu, D., Kulkarni, S., Rosenberg, C. and Chai, H., 2006a. Analysis of a CDN–P2P hybrid architecture for cost-effective streaming media distribution. *Multimedia Systems*, 11(4):383–399. 53

Xu, D., Kulkarni, S., Rosenberg, C. and Chai, H., 2006b. Analysis of a CDN–P2P hybrid architecture for cost-effective streaming media distribution. *Multimedia Systems*, 11(4):383–399. doi:10.1007/s00530-006-0015-3. 54, 57, 58, 59, 60

Xu, Y., Zhou, Y. and Chiu, D., 2014. Analytical QoE Models for Bit-Rate Switching in Dynamic Adaptive Streaming Systems. *IEEE Transactions on Mobile Computing*, 13(12):2734–2748. doi:10.1109/tmc.2014.2307323. 22

Xunlei Kankan, 2017. Xunlei Kankan. http://www.kankan.com. [Online]. 2

Yin, H., Liu, X., Zhan, T., Sekar, V., Qiu, F., Lin, C., Zhang, H. and Li, B., 2009. Design and Deployment of a Hybrid CDN-P2P System for Live Video Streaming: Experiences with LiveSky. In *ACM International Conference on Multimedia*, MM '09, pages 25–34. ACM, New York, NY, USA. 2, 53

Yin, H., Liu, X., Zhan, T., Sekar, V., Qiu, F., Lin, C., Zhang, H. and Li, B., 2010. Livesky: Enhancing cdn with p2p. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 6(3):16. 2, 53, 54

Yin, X., Jindal, A., Sekar, V. and Sinopoli, B., 2015. A Control-Theoretic Approach for Dynamic Adaptive Video Streaming over HTTP. *ACM SIGCOMM Computer Communication Review*, 45(5):325–338. doi:10.1145/2829988.2787486. 22, 29

Yin, X., Sekar, V. and Sinopoli, B., 2014. Toward a Principled Framework to Design Dynamic Adaptive Streaming Algorithms over HTTP. *ACM Workshop on Hot Topics in Networks - HotNets*. doi:10.1145/2670518.2673877. 22, 29

Yitong, L., Yun, S., Yinian, M., Jing, L., Qi, L. and Dacheng, Y., 2013a. A study on quality of experience for adaptive streaming service. In *IEEE International Conference on Communications Workshops (ICC)*, pages 682–686. IEEE. 20, 127

Yitong, L., Yun, S., Yinian, M., Jing, L., Qi, L. and Dacheng, Y., 2013b. A study on Quality of Experience for adaptive streaming service. *2013 IEEE International Conference on Communications Workshops (ICC)*. doi:10.1109/iccw.2013.6649320. 20

Youtube, 2017. Youtube. https://www.youtube.come. Online; Accessed August 2017. 2

Zambelli, A., 2009. Microsoft Smooth Streaming. http://www.iis.net/learn/media/on-demand-smooth-streaming/smooth-streaming-technical-overview. Online; accessed August 2017. 3

Zattoo, . Zattoo. https://www.zattoo.com/. Online; Accessed August 2017. 2

Zhang, G., Liu, W., Hei, X. and Cheng, W., 2015a. Unreeling Xunlei Kankan: understanding hybrid CDN-P2P video-on-demand streaming. *IEEE Transactions on Multimedia*, 17(2):229–242. 2, 53, 54, 55

Zhang, S. *et al.*, 2015b. Presto: Towards fair and efficient HTTP adaptive streaming from multiple servers. *IEEE International Conference on Communications (ICC)*. 41, 48

Zhang, X., Liu, J., Li, B. and Yum, Y., 2005. CoolStreaming/DONet: A data-driven overlay network for peer-to-peer live media streaming. *IEEE INFOCOM*. 52, 60

Zhao, M., Aditya, P., Chen, A., Lin, Y., Haeberlen, A., Druschel, P., Maggs, B., Wishon, B. and Ponec, M., 2013. Peer-assisted content distribution in Akamai netsession. *2013 conference on Internet measurement conference - IMC 13*. doi:10.1145/2504730.2504752. 53, 54, 55

Zhou, C., Lin, C., Zhang, X. and Guo, Z., 2013a. Buffer-based smooth rate adaptation for dynamic HTTP streaming. *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference*. doi:10.1109/apsipa.2013.6694183. 22, 29

Zhou, C., Lin, C., Zhang, X. and Guo, Z., 2014. A Control-Theoretic Approach to Rate Adaption for DASH Over Multiple Content Distribution Servers. *IEEE Transactions on Circuits and Systems for Video Technology*, pages 681–694. 41, 48

Zhou, C., Zhang, X. and Guo, Z., 2013b. A control theory based rate adaption scheme for dash over multiple servers. *Visual Communications and Image Processing (VCIP)*. doi:10.1109/vcip.2013.6706335. 22, 29

Zink, M., Suh, K., Gu, Y. and Kurose, J., 2009. Characteristics of YouTube network traffic at a campus network – Measurements, models, and implications. *Computer Networks*, 53(4):501–514. doi:10.1016/j.comnet.2008.09.022. 56