Universidade de Aveiro Departamento de Eletrónica, Telecomunicações e Informática
2019

André Filipe Pinheiro Dias

QoE over-the-top multimédia em redes sem fios

QoE over-the-top multimedia over wireless networks

Universidade de Aveiro Departamento de Eletrónica, Telecomunicações e Informática
2019

André Filipe Pinheiro Dias

QoE over-the-top multimédia em redes sem fios

QoE over-the-top multimedia over wireless networks

"*O desafio da modernidade vence-se desenvolvendo soluções criativas, dando a resposta a problemas antigos.* "

— Carlos Dias

Universidade de Aveiro
2019

Departamento de Eletrónica,
Telecomunicações e Informática

**André Filipe
Pinheiro Dias**

QoE over-the-top multimédia em redes sem fios

QoE over-the-top multimedia over wireless networks

Tese apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Computadores e Telemática, realizada sob a orientação científica da Doutora Susana Isabel Barreto de Miranda Sargento, Professora associada c/ agregação do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro, e do Doutor André Braga Reis, Investigador de Pós-Doutoramento do Instituto de Telecomunicações de Aveiro .

agradecimentos / acknowledgements

Aproveito a oportunidade para agradecer à minha família, Pai, Mãe, Irmão e Afilhada, pelo prazer e pelas oportunidades que me proporcionaram de estudar na universidade e curso que gosto e escolhi, assim como todo o apoio durante todo este percurso.
A todos os meus amigos pelos momentos de felicidade que tivemos nesta academia e iremos ter no futuro.
A todos os colegas do grupo NAP, Tiago Oliveira, Rui Dias, José Silva e Pedro Martins pelo conhecimento e motivação transmitido durante este periodo de investigação e dissertação.
À Professora Doutora Susana Sargento, Doutor André Reis, Doutor Lucas Guardalben, Doutor João Nogueira e Professor Adriano Fiorese por todo o apoio, conhecimento, orientação e dicas para a vida que me deram.
Agradeço o suporte financeiro no âmbito do Projeto POCI-01-0247-FEDER-017738 UltraTV – Ecossistema de aplicações para TV UltraHD.
Muito obrigado!...

Palavras Chave

Over-The-Top, Multimédia, Redes de Entrega de Conteúdos, Caching, Qualidade de Experiência, Machine Learning.

Resumo

Um dos objetivos de um operador é melhorar a qualidade de experiência do cliente em redes onde existem conteúdos Over-the-top (OTT) a serem entregues. O aparecimento de serviços como o YouTube, Netflix ou Twitch, onde no primeiro caso são carregadas mais de 300 horas de vídeo por minuto na plataforma, vem trazer problemas às redes de dados geridas que já existiam, assim como desafios para os resolver. O tráfego de vídeo corresponde a 75% de todos os dados transmitidos na Internet. Assim, não só a Internet se tornou o meio de transmissão de vídeo 'de facto', como o tráfego de dados em geral continua a crescer exponencialmente, proveniente do desejo de consumir mais conteúdos. Esta tese apresenta duas propostas de modelos e arquitetura que pretendem melhorar a qualidade de experiência do utilizador, ao prever a quantidade de vídeo em avanço passível de ser pré-carregado, de forma a optimizar a eficiência de entrega das redes onde a qualidade de serviço não é possível de ser garantida. O pré-carregamento dos conteúdos é feito no servidor de cache mais próximo do cliente. Para tal, é utilizado um processo analítico hierárquico (AHP), onde através de um método subjetivo de comparação de atributos, e da aplicação de uma função de valores ponderados nas medições das métricas de qualidade de serviço, é obtida a quantidade a pré-carregar. Além deste método, é também proposta uma abordagem com técnicas de inteligência artificial. Através de redes neurais, há uma tentativa de auto-aprendizagem do comportamento das redes OTT com mais de 14.000 horas de consumo de vídeo sobre diferentes condições de qualidade, para se tentar estimar a experiência sentida e maximizar a mesma, sem degradação da entrega de serviço normal. No final, ambos os métodos propostos são avaliados num cenário de utilizadores num comboio a alta velocidade.

Keywords

Over-The-Top, Multimedia, Content Delivery Networks, Caching, Quality of Experience, Machine Learning.

Abstract

One of the goals of an operator is to improve the Quality of Experience (QoE) of a client in networks where Over-the-top (OTT) content is being delivered. The appearance of services like YouTube, Netflix or Twitch, where in the first case it contains more than 300 hours of video per minute in the platform, brings issues to the managed data networks that already exist, as well as challenges to fix them. Video traffic corresponds to 75% of the whole transmitted data on the Internet. This way, not only the Internet did become the 'de facto' video transmission path, but also the general data traffic continues to exponentially increase, due to the desire to consume more content. This thesis presents two model proposals and architecture that aim to improve the users' quality of experience, by predicting the amount of video in advance liable of being prefetched, as a way to optimize the delivery efficiency where the quality of service cannot be guaranteed. The prefetch is done in the clients' closest cache server. For that, an Analytic Hierarchy Process (AHP) is used, where through a subjective method of attribute comparison, and from the application of a weighted function on the measured quality of service metrics, the amount of prefetch is achieved. Besides this method, artificial intelligence techniques are also taken into account. With neural networks, there is an attempt of self-learning with the behavior of OTT networks with more than 14.000 hours of video consumption under different quality conditions, to try to estimate the experience felt and maximize it, without the normal service delivery degradation. At last, both methods are evaluated and a proof of concept is made with users in a high speed train.

Contents

List of Figures

List of Tables

Glossary

AHP	Analytic Hierarchy Process
AMOS	Average Mean Opinion Score
API	Application Programming Interface
ATM	Asynchronous Transfer Mode
CDN	Content Delivery Network
CLI	Command Line Interface
CPU	Central Processing Unit
DASH	Dynamic Adaptive Streaming over HTTP
DSL	Digital Subscriber Line
FPS	Frames per Second
FTTH	Fiber to the Home
HAS	HTTP Adaptive Streaming
HDS	HTTP Dynamic Streaming
HTTP	Hypertext Transfer Protocol
ICMP	Internet Control Message Protocol
IIR	Infinite Impulse Response
IIS	Internet Information Services
IP	Internet Protocol
ISP	Internet Service Provider
ITU-R	International Telecommunication Union - Radiocommunication Sector
ITU	International Telecommunication Union
LTE	Long-Term Evolution
LTS	Long Term Support
MOS	Mean Opinion Score
MP4	Moving Pictures Experts Group-4 Part 14
MPEG	Moving Picture Experts Group
MPLS	Multi-Protocol Label Switching
MTU	Maximum Transmission Unit
NAAD	Network-Awareness Adaptation & Distribution
NFS	Network File System
NTP	Network Time Protocol
OQD	Optimized Quality of DASH
OTA	Over-the-air
OTT	Over-the-top
PON	Passive Optical Network
PPPoE	Point-to-Point Protocol over Ethernet)
QoE	Quality of Experience
QoS	Quality of Service
RAM	Radom Access Memory
RTP	Real-time Transport Protocol
RTSP	Real Time Streaming Protocol
SDN	Software Defined Network
SSH	Secure Shell
TCP	Transmission Control Protocol
TV	Television
UDP	User Datagram Protocol
USB	Universal Serial Bus
VCR	Video Cassette Recorder
VM	Virtual Machine
VoD	Video on Demand
XML	Extensible Markup Language

CHAPTER 1

Introduction

One of the objectives of an operator is to improve the networks where Over-the-top (OTT) content is being delivered to the consumer. It is well known that an end user will have a different QoE based on where the content is being watched (Phone, Laptop, Desktop, Set-top Box, TV), and what kind of network is in use (either DSL, 3G, 4G, Fiber, etcetera). Since this content might be delivered over all these different types of networks, each of them with different characteristics, adequate methods and approaches are needed to improve the user's QoE.

Ideally, the user would like to obtain the best QoE possible, but on the other hand, it is difficult for the provider to guarantee the QoE due to several reasons. These reasons can range from high latency on the user's end, low bandwidth due to a cellular connection, variable jitter and packet loss on congested wireless environments, or even high splitting ratios leading to a mix of the previously mentioned parameters. The result is a variable QoE which is very hard to guarantee. Therefore, regarding this objective some questions arise:

- Can the user's QoE be improved by using Quality of Service (QoS) network metrics as a mean to estimate the amount of prefetching?
- Can a content prefetching approach based on those QoS network metrics improve QoE?

Substantial work has already been done showing how a network architecture based on tier caching layers and methods might provide the final mobile user with an improved QoE, by using prefetching techniques that aim to reduce the content delivery response time by bringing said content as close as possible to the consumer before it is explicitly requested [1].

This Thesis proposes an approach to fine tune a prefetching mechanism, based on the network conditions reported as a networking awareness model, supporting a proposed approach for forecasting the number of video chunks to prefetch in advance, by using a weighted average, where the weights are calculated using the judgment matrix technique found on the multi-criteria Analytic Hierarchy Process (AHP) decision method. Moreover, a machine learning approach using neural networks will be considered to learn the network conditions and the required prefetching to improve the QoE.

1.1 Motivation

Given the importance of the users' opinion score, it is imperative to serve the best possible QoE to them. As such, the system must be designed to accommodate the best general quality to the biggest amount of users as possible, thus improving the overall QoE.

One of the most important metrics that influence the Mean Opinion Score (MOS) is the amount and duration of buffering that occurs during a Live or Video on Demand (VoD) transmission. In order to reduce the buffering events, adaptive streaming became one of the leading video transmission methods. As the name itself explains, the streams are adapted along the transmission to accommodate network variations that disrupt a seamless stream.

Typically, in adaptive streaming the quality (bit rate) is decreased when, for some reason, the network quality is decreased. This quality drop tends to be due to high delay, or due to lower available bandwidth. In fiber networks providers are selling bundles with substantial bandwidth, and fiber optics give the lowest possible delay in networks nowadays. However, it is still a reality that most home users do not have those kinds of connections, and nowadays there is a growing interest in mobile devices to consume multimedia content.

However, DSL, 3G/LTE, WiFi, Satellite, and nearly every other technology currently used, have significant and known issues [2] that converge in varying and low results of QoE, which presents both a challenge and a margin for improvement.

1.2 Objectives and Contributions

It is, thus, of the utmost importance to find better and smarter solutions that take into account the currently implemented architectures and technologies in the real world. These solutions should aim to improve the overall QoE, even if this leads to more network overhead. Improving the QoE will directly lead to customer satisfaction, and indirectly to a growth in the company's installed user base.

With this in mind, the primary goal of this thesis is to find solutions that aim at improving the user's QoE while still minimizing the network overhead. This improvement should extend to every network technology, meaning wired or wireless, with high or low bandwidth, latency and others.

Some other specific goals of this thesis are to:

- Analyze how variable networking conditions might affect a client's experience;
- Propose two distinct optimization methods with the potential to minimize how networking variations influence the client;
- Propose and implement an architecture that is capable of supporting tests with both methods without direct human interference and able to output comparable results;
- Emulate a real-world scenario and apply both methods in order to verify their applicability in actual real conditions.

Internal research project reports for the P2020 UltraTV project (Altice Labs and NAP/IT) on QoE Evaluation, Adaptative distributed mechanisms and Content delivery mechanisms have been delivered.

1.3 Outline

The first chapter of the thesis presents an introduction to the importance of QoE in today's multimedia context, motivation and goals.

The second chapter presents the State of the art, by showing the current developments concerning OTT multimedia networks, multimedia streaming technologies, prefetching and QoE.

The third chapter describes the proposed optimization models, as well as the architecture that aims not only to mirror an OTT distribution system, but also to have a testbed that allows addressing comparisons under different environments.

The fourth chapter describes all the modules included in the architecture, their implementation and the communication process between parts of the system.

The fifth chapter displays the results and comparison of the proposed models, and how they would be applicable in a real-world scenario.

The sixth chapter provides the conclusions of this dissertation and future research guidelines.

CHAPTER 2

State of the Art

This chapter presents related work in the area of this Dissertation.

2.1 OTT Multimedia Networks

Global Internet Protocol (IP) traffic is increasing exponentially. Not only each client is relying on ever-increasing bandwidth, but having higher quality on the consumed content has a massive impact on the network. With the rise of new players and modern technologies, Over-the-Top services became not only a selling point, but a challenge for operators to face on a global scale. Besides, this OTT content is delivered within the Internet Service Providers (ISPs)' network for free, by a third-party, which sells a service directly to the customer. Maintaining these services is becoming harder while still keeping a high Quality of Experience (QoE) for the overall users, given that operators have no direct control over the data.

Nowadays, the largest slice of global IP network traffic belongs to video, which amounts to 75% (data from 2017). This results in nearly 93000 Petabytes of monthly video data. These statistics are expected to increase in the next few years [3]. This means not only that Internet has became one of the standard ways to transport video, but also that it became part of the daily life. Furthermore, with the rise of online services like Youtube and Netflix, the desire for users' customized content lead to tailor-made Over-the-Top (OTT) services, again increasing the need for more data. Given the exponentially increase in bandwidth generated from this kind of traffic, both the networks and the Service Providers have to deal with issues that arise from it. Consequently, there is the need to guarantee the best possible overall users' Quality of Experience to maximize profits and keep clients in the business.

2.2 OTT Multimedia Services in Telecommunication Operators

In the beginning, operators had pay TV, which is a business model that allows watching television content via a subscription. This model was supported by managed networks where operators could directly control its quality and content. Unlike Over-the-air television signals, where content can be watched for free, pay TV needs proprietary equipment and is typically delivered via satellite, coaxial cable and more recently through fiber via an overlay technique. The RF Video Overlay consists of injecting the Television (TV) signal coming from the Digital TV Headend on a Passive Optical Network (PON), which has enough bandwidth to support hundreds of TV channels under a single fiber wavelength (1550 nm) together in the same fiber as the downstream (1490 nm) and upstream (1310 nm) traffic [4].

The appearance of OTT services took all the control from the operators, and with 300 hours of new video being uploaded every minute and 5.000 million videos watched every day on YouTube itself, these services came to offer a user-centered content which was not available before. This offer may allow clients to watch TV beyond the television at home, and expand the multimedia into smartphones, computers or even gaming consoles and become a strong competition against common operator's services. Not only it becomes a possibility, but the users are changing the old way of watching television [5]. With this community interest and a broad variety of content, operators are shifting and implementing their own OTT services.

The different ways of watching television content are Linear TV, Time-shift TV, and Video-on-Demand [6].

Linear TV is the regular TV broadcast. It was for decades the dominant method of watching TV. This kind of broadcast follow a pre-defined program line-up and can be serviced as an Over-the-air (OTA) or Pay-TV service which was the traditional way to watch TV for decades.

Time-shift is detached from Linear TV because the content does not follow a time-specific frame to be watched. Normal Linear TV content is recorded, and through time-shifting, it can be seen later. The amount of time that each time-shifting functionality is available, and, which and how much channels are supported, changes depending on the operator choice.

Pause TV is a functionality of time-shift, which allows for a client to pause the current broadcast, and continue at a later time from the same moment or back to live content.

Start-over TV the program can be restarted during its transmission or even after it finishes. Personal Video Recorder is only used with the direct user interaction, meaning that a user either manually starts a record at a specific time, or schedules a later time. Similar to Video Cassette Recorder (VCR), with the addition of supporting non-linear access and larger capacities.

Catch-up TV relies on an automated process that records all Live content into VoD, allowing for the visualization of specific programs at any time. Some operators offer up to one month of Catch-up TV.

VoD concerns services where users pay for specific content access and can be divided in:

Transaction VoD which a user purchases the access to specific content for a pre-determined amount of time, for example, 24 or 48 hours. During that period, the content can be accessed as many times as desired.

Electronic Sell Through VoD is a type of VoD that sells access to the content on a platform. The access is unrestricted, and only visible through the mentioned platform. Subscription VoD sells access into a platform instead of a single content. In this platform, every content from the catalog can be accessed without restrictions.

2.3 Multimedia Streaming Technologies

Streaming is defined as the process of transferring data via a channel to its destination with real-time characteristics, where it is decoded and consumed via a user/device in real time, i.e., as the data is being delivered on the fly [7].

With the appearance of Real Time Streaming Protocol (RTSP) [8], streaming became more popular. By then, common internet connections were Dial-ups, which had 56Kbps of bandwidth, were expensive, and not enough to support the transmission of multimedia. The next step in access networks evolution was the appearance of DSL, which brought broadband internet speeds, capable of sustaining higher throughput that together with better video compression algorithms, the streaming of audio and video content is now feasible.

The relevance of video streaming services kept growing in the last decade, and TV streaming services became so widespread that most telecommunication operators have TV streaming offers of their own, both IPTV and OTT [6].

In push-based protocols, a client and server must establish a connection, which the server keeps a session and streams the packets to the client until the connection is either stopped or interrupted by the client. In pull-based streaming, the server remains idle, waiting for client requests. The most common protocol for pull-based streaming is Hypertext Transfer Protocol (HTTP) [9].

2.3.1 Traditional Streaming

In [10], it is shown that RTSP is defined as a stateful protocol which means that, from the first time a client connects to the streaming server until disconnection, the server keeps track of a client's state. The client communicates its state to the server by issuing PLAY, PAUSE or TEARDOWN commands in order to control the stream.

Once a session between the client and the server has been established, the server begins sending down the media as a steady stream of small packets (the format of these packets is known as Real-time Transport Protocol (RTP)). The size of a typical RTP packet is 1452 bytes, which means that, in a video stream encoded at 1 Mbps, each packet only carries roughly about 11 msec of video. This packet size as Maximum Transmission Unit (MTU) is quite efficient under DSL, because of the Point-to-Point Protocol over Ethernet) (PPPoE) protocol and the legacy Asynchronous Transfer Mode (ATM) core frame sizes as explained in [11]. In RTSP the packets can be transmitted either over User Datagram Protocol (UDP) or Transmission Control Protocol (TCP) transports – the latter is preferred in cases where firewalls or proxies block UDP packets, but can also lead to increased latency (TCP packets get re-sent until received).

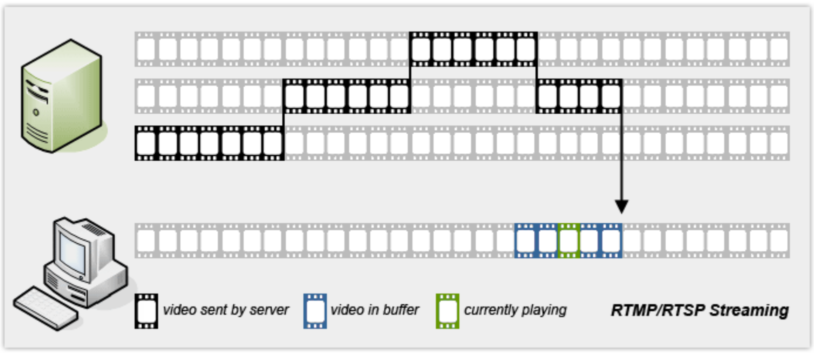


Figure 2.1: Traditional Streaming (RTSP) [12]

However, RTSP presents some constraints. Considering a large platform that needs to deliver streaming content to millions of users, the system would need to handle the same amount of sessions. Besides, each session would lead to multiple port management which increases the complexity of the system. Furthermore, each client would need to make sure his router or gateway would have NAT Transversal rules for RTSP ports and protocol. Due to the issues presented, only some particular use-cases adopted this technology [6].

A comparable solution, but in this case stateless, is the HTTP protocol. If a client requests data, the server will deliver, however, it will be an independent and one-time session. In order to use this protocol for streaming, Windows Media Services use a modified version of HTTP, known as MS-WMSP, which adds to the regular HTTP the possibility of keeping track of the states.

2.3.2 Progressive Download

Progressive download is another kind of pull-based content delivery. The name has its origins in the way the content is downloaded while it can be playing at the same time. Usually, browsers have a cache that provides temporary storage for some seconds ahead of the video playing time. If the server supports HTTP 1.1, this delivery method also allows seeking, which consists of requesting positions of the multimedia file, that were not yet downloaded [13].

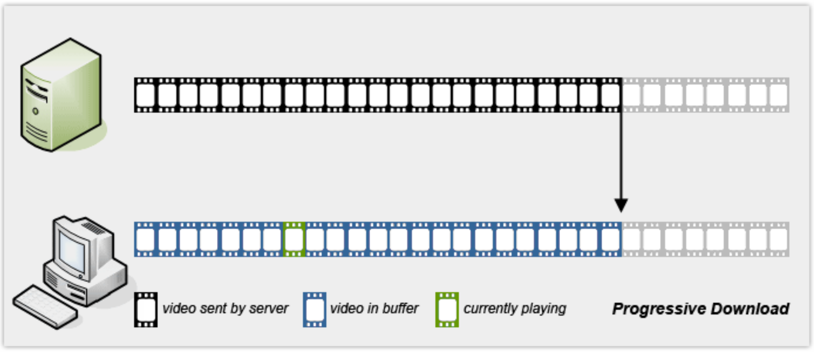


Figure 2.2: Progressive Streaming [12]

Since this technology supports HTTP, some advantages are that most firewalls already allow this content to be transverse, and the content can be the target of caching through proxy or Content Delivery Networks (CDNs) with the addition of not needing to keep a session per client. These benefits are some of the reasons why it has been commonly adopted and is now in use by some of the biggest players, including YouTube, Vimeo, and MySpace.

There are a few downsides, which include not supporting Live streaming since, for the progressive download to happen, the content would automatically stop being live. The streaming cannot be adjusted based on QoS metrics while RTSP could. There could be initial and seeking delay because a few seconds of video must be buffered first. While RTSP may support a packet loss without stopping, this technology must receive all data before advancing. And at last, every time a video is not fully watched there is always wasted bandwidth [6].

2.3.3 Adaptive Streaming

Adaptive streaming emerged as a way to unite the advantages of older technologies. One of those advantages is supporting adaptation similarly to the RTSP, where through the feedback QoS metrics sent through RTCP it is possible to adapt.

A scalable encoding process can achieve better results depending on the feedback send and extra data received, for example, as the SVC class provides.

Segmented HTTP-based delivery

To achieve an easy and scalable delivery process it can use the HTTP protocol, so, it leverages web servers, caches, and CDN distribution.

Another advantage is the support for high scalability since it is at its base a file transfer, but instead of one big progressive download, it relies on very small progressive downloads (chunks). The player has the responsibility of merging all those chunks, which are videos with few seconds, and make them appear to the user as if it would be a continuous stream.

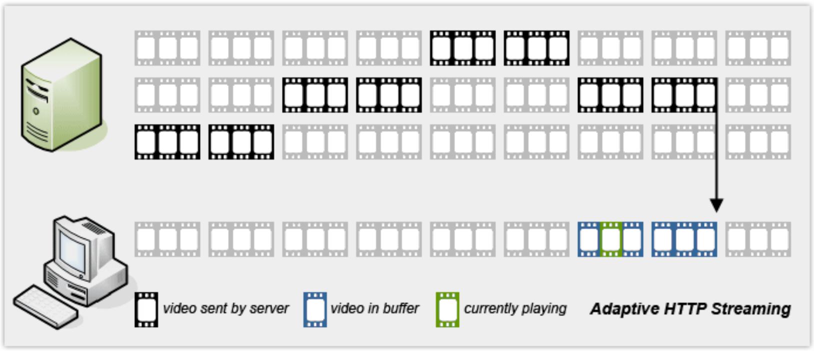


Figure 2.3: Adaptive Streaming [12]

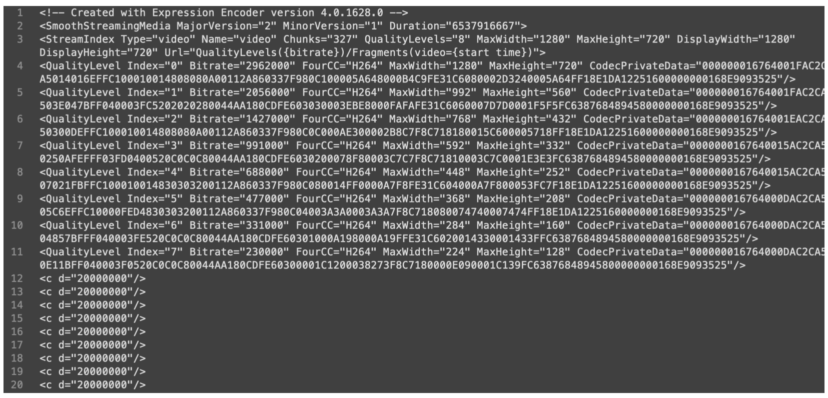
The chunks are usually 2-4 seconds long, and the content is encoded with different qualities to allow quality (bitrate requested) switching, thus achieving bandwidth efficiency. Since the player merges those small chunks to create a big continuous stream at the client side, it means that each sequential chunk can have a different quality, which gives freedom to the client to use different methods to try and improve the overall quality through that request process.

Even though it has eliminated many downsides from the previous streaming types, like lack of live streaming and adaptation support, it still possesses some disadvantages, like the lack of standardization. Since different players are developing and improving this technology, technologies like Apple HTTP Live Streaming (HLS) [14], Microsoft Smooth Streaming [15], and Adobe HTTP Dynamic Streaming (HDS) [16] have different support, players and requirements.

Smooth Streaming

Microsoft smooth streaming is an adaptive segmented http-based media delivery developed by Microsoft. The technology has thee elements, the encoder Microsoft expression encoder; Microsoft's Internet Information Services (IIS) Media Services extension, which is an extension for IIS that provides the streaming service to the clients and the smooth streaming client, which is a player based on Silverlight capable of playing the smooth streaming content.

The encoder provides methods to transform other video formats into smooth streaming capable ones. The way it stores files is according to a single contiguous file for each encoded bitrate. It uses MP4 as its file format and video container.



```xml
<!-- Created with Expression Encoder version 4.0.1628.0 -->
<SmoothStreamingMedia MajorVersion="2" MinorVersion="1" Duration="6537916667">
<StreamIndex Type="video" Name="video" Chunks="327" QualityLevels="8" MaxWidth="1280" MaxHeight="720" DisplayWidth="1280"
DisplayHeight="720" Url="QualityLevels({bitrate})/Fragments(video={start time})">
<QualityLevel Index="0" Bitrate="2962000" FourCC="H264" MaxWidth="1280" MaxHeight="720" CodecPrivateData="000000016764001FAC2C
A5014016EFFC100010014808080A00112A860337F980C100005A648000B4C9FE31C6080002D3240005A64FF18E1DA12251600000000168E9093525"/>
<QualityLevel Index="1" Bitrate="2056000" FourCC="H264" MaxWidth="992" MaxHeight="560" CodecPrivateData="000000016764001FAC2CA
503E047BFF040003FC5202020280044AA180CDFE603030003EBE8000FAFAFE31C6060007D7D0001F5F5FC63876848945800000000168E9093525"/>
<QualityLevel Index="2" Bitrate="1427000" FourCC="H264" MaxWidth="768" MaxHeight="432" CodecPrivateData="000000016764001EAC2CA
50300DEFFC100010014808080A00112A860337F980C0C000AE300002B8C7F8C718180015C600005718FF18E1DA12251600000000168E9093525"/>
<QualityLevel Index="3" Bitrate="991000" FourCC="H264" MaxWidth="592" MaxHeight="332" CodecPrivateData="0000000167640015AC2CA5
0250AFEFFF03FD0400520C0C0C80044AA180CDFE6030200078F80003C7C7F8C71810003C7C0001E3E3FC63876848945800000000168E9093525"/>
<QualityLevel Index="4" Bitrate="688000" FourCC="H264" MaxWidth="448" MaxHeight="252" CodecPrivateData="0000000167640015AC2CA5
07021FBFFC10001001483030320011 2A860337F980C080014FF0000A7F8FE31C604000A7F800053FC7F18E1DA12251600000000168E9093525"/>
<QualityLevel Index="5" Bitrate="477000" FourCC="H264" MaxWidth="368" MaxHeight="208" CodecPrivateData="000000016764000DAC2CA5
05C6EFFC10000FED4830303200112A860337F980C04003A3A0003A3A7F8C718080074740007474FF18E1DA12251600000000168E9093525"/>
<QualityLevel Index="6" Bitrate="331000" FourCC="H264" MaxWidth="284" MaxHeight="160" CodecPrivateData="000000016764000DAC2CA5
04857BFFF040003FE520C0C0C80044AA180CDFE60301000A198000A19FFE31C6020014330001433FFC63876848945800000000168E9093525"/>
<QualityLevel Index="7" Bitrate="230000" FourCC="H264" MaxWidth="224" MaxHeight="128" CodecPrivateData="000000016764000DAC2CA5
0E11BFF040003F0520C0C0C80044AA180CDFE60300001C1200038273F8C7180000E090001C139FC63876848945800000000168E9093525"/>
<c d="20000000"/>
<c d="20000000"/>
<c d="20000000"/>
<c d="20000000"/>
<c d="20000000"/>
<c d="20000000"/>
<c d="20000000"/>
<c d="20000000"/>
<c d="20000000"/>
```

Figure 2.4: Smooth Streaming client Manifest

The encoder also generates two manifest files, which are two Extensible Markup Language (XML) style files. This file provides useful information to the client, such as the encoded content type, that can be text, audio or video, the bit rate, resolution, codec type and its location. The second one, depicted in figure 2.4 which is the client manifest provides information about the content, such as, the available resolutions, the duration and the timestamps of each fragmented chunk.

By accessing the manifest, the client is now able to choose the preferred method to and dynamically decide which quality to request, giving freedom to the creation and use of heuristics modules that determine when and how to switch bitrates.

2.4 Content Delivery Networks and Quality of Service

Content delivery networks are a way to distribute content as an optimized way to every client, allowing services to scale better, be faster and provide higher availability by sharing resources, and at the same time reducing origin Load [17], [18]. The goal of this kind of delivery is increasing the quality of experience. With the massification of Over-the-top (OTT) content, the use of HTTP protocol to deliver streaming and the file-object storage of multimedia, CDNs now also support OTT multimedia.

One of the goals of the CDNs is bringing the content close to the user, so the delivery is quicker and nefarious network effects are subtler. In [19] several approaches have been studied and show that proxy caching is an effective means to reduce access latencies as well as resource consumption for networked applications, and show proof that good results can be achieved due to the unique features of media objects like huge size and high bandwidth demand.

These caching solutions aim to have an intermediary between the client and the origin server, not only to lower the usage of the upper tier servers but because of faster delivery times. The downside is that these servers typically have a smaller storage than the origins. So, some form of replacement strategy must exist to choose what to cache, when to cache, and how long.

According to [20] there are stategies based on:

- Access Time Intervals
- Frequency
- Object Size
- Objective Function

The way they measure efficiency is through cache hit or miss. Whenever a HIT occurs, it counts as a positive effect and a MISS as a negative. In [21] it seems that a relatively small number of proxy-caches with a reasonable amount of storing capacity are able to ensure a high byte hit ratio. This is desirable especially in situations when the external bandwidth is a scarce or expensive resource. For example if a multimedia origin server is on another continent or country, this can be a good approach to improve the service and experience for users abroad. These are some of the use cases targeted by the CDNs from YouTube and Netflix.

From [22] some example of applications of CDN technologies may be found all over the Internet:

- Academic Institutions : Codeen, Coral, Flash Crowds Alleviation Network (FCAN);
- Network Operators : AT&T, Telefonica, Telus, British Telecom;
- Social Networks : Facebook, MySpace, Twitter;
- Online Retailers : Amazon, eBay, NewEgg;
- Media Providers : YouTube, Netflix, Hulu, iTunes;
- Service Providers : Akamai, CloudFlare, Amazon CloudFront.

Popular solutions for proxy-caching modules and systems include Apache Traffic Server (ATS) [23], Microsoft's IIS with Application Request Routing (ARR) [24], Nginx [25], Varnish [26], and Squid [27]. They may be combined to optimize the cache performance of a delivery system.

Nginx was chosen to be used in this thesis since it is commonly used as a production proxy-caching. Also, it show good performance ranking compared to other solutions [1], [28].

2.5 Prefetching

Content prefetching brings contents close to end users before their explicit requests to reduce the content retrieval time [29]. Based on this principle [30] shows that it is possible to achieve energy and traffic savings by having prefetched content.

The challenge is to correctly predict the future consumption of clients and populate the caches with the content before the consumer requests it. According to [31], prefetching strategies are diverse, and no single strategy has yet been proposed which provides optimal performance, since there will always be a compromise between the hit ratio and bandwidth [32]. Intuitively, to increase the hit ratio, it is necessary to prefetch those objects that are accessed most frequently but, to minimize the bandwidth consumption, it is necessary to select those objects with longer update intervals [33]. To be effective, prefetching must be implemented in such a way that prefetches are timely, useful and introduce little overhead [32].

Much older than OTT content and even web consumption, similar prediction needs are found in the CPUs themselves. Several layers of cache are implemented to maximize hit ratios and decrease latencies by using higher cache tiers only when inevitable. By using a similar strategy, when a consumer requests the OTT content, in this case, a smooth streaming video, to the proxy-cache via HTTP, the prefetching mechanism calculates the next chunks and predicts their qualities, which customers will later request them. Once the prefetching predicts the future chunks, it requests those chunks to the proxy via HTTP, which will eventually be requested by the clients. Thus, the prefetching pre-populates the cache.

2.6 Quality of Experience

The typical network performance analysis is covered by QoS. However, nowadays given the OTT content and the highly user-centered content, QoE is taking an important part in the network analysis. It is crucial to understand which factors can be controlled and which cannot in order to maximize the users' QoE. To do so, there must be a way to measure QoE, since something that cannot be measured cannot be optimized. QoE measurement, or assessment, is divided into two main categories, depending on how it is performed: subjective and objective.

The correlation between subjective quality assessment (i.e. Mean Opinion Score, MOS evaluation), video performance metrics and Quality of Service (QoS) parameters has been studied in [34], including the analysis over both subjective and objective QoS factors that affect QoE [35], as well as tests to see how QoS parameters impacted the final QoE. It is then very visible how small QoS variations might have a big impact on the user's experience. It is also based on this correlation principle that the proposed architecture targets cache optimization for QoE improvement. First, the solution needs a way for QoE estimation. The work done has been based on the following research [36], which proposes a model for determining the QoE estimate of a playback session of HTTP adaptive video streaming, encompassing its complete range of characteristics. It also states that, to reach the envisioned solution, several key-metrics are extracted throughout the playback session, and then analyzed by an analytical method to predict the consumers' QoE. That work has been validated by conducting a survey. Given the close MOS estimation to the aforementioned model, it will be used in this solution to show how MOS reacts to different tests under different scenarios. It is important to observe how this model reacts to the QoS parameters variation.

2.6.1 Subjective Assessment

Subjective QoE assessments involve surveys to people that have been exposed to the service or application whose QoE is being assessed. These surveys rely on users' opinions to rate the service performing under different conditions. The rating system may be based on qualitative or quantitative evaluations. Also, this evaluation technique typically requires users to give their opinion, which will be highly based on personal life experiences giving a different perspective by each different user, its mood, or perhaps the system responsiveness.

Qualitative evaluations tend to focus on comparative evaluations between different experiments, such as indicating that the first experience was more pleasant than the second one. This type of evaluation may require a huge post processing of the data, perhaps with advanced statistical tools, to be able to be used in computer simulations or operations, as it is not formatted for a computer to understand. As such, it does not appear to be the best option to be used in a dynamic QoE environment that needs to adapt itself to multiple constraints. An example would be a user answering an open-ended question comparing the assessed service or application.

Some of the examples would be:

- What do you think was worse/better in the second compared video?
- Which video you consider better, the first or the second?
- How the user experience improved/decreased?
- What is your opinion on the application/system?
- What do you think could be improved?

As for quantitative assessments, users are asked to use a number to grade their experience according to pre-established scales; thus, being objective. This is more widely used as it facilitates data processing. In the context of video reproduction, ITU-R recommends the usage of BT.500–13 [37] for video quality assessment. Given the pre-defined scales, a computer can use them as values which can then be used to adjust the so called dynamic adaptive QoE environment as needed. As such, this appears to be most suited for the intended goal under the subjective type.

Some examples would be:

- On a scale of 1 to 5, how would you rate the first/second video?
- Using the same scale, how would you rate the system?
- Give a score to the experience.
- Score the video quality.

2.6.2 Objective Assessment

In contrast to the previously described subjective assessment methods, which are laborious, time-consuming and expensive, the objective approach is an alternative that builds on the technical characteristics usually provided by QoS parameters to model and extrapolate a perceptual quality indicator. Because there is usually no way of ensuring that, by itself, the model is accurate, objective assessment algorithms are usually trained and modeled by resorting to known QoE estimations from subjective models.

There are some objective metrics that may be used, such as video frame rate, resolution, and compression level, which can be monitored using specific tools [38], and then correlated with the users' perceived quality. These correlations enable the creation of QoE models based on technical parameters.

Some examples would be:

- Amount of buffering (milliseconds);
- Average throughput (Mbps);

- Number of chunks requested on each quality level;
- Number of times the quality increased/decreased;
- Total consumed time vs requested time;
- Initial playout delay;
- Seeking delay.

According to [39], there has been some development in the Content Inspection/Artifact-based Measurement. These two are related to the artifact analysis of the video and operate over the decoded content which is a way to measure QoE. This is a quite exact way to measure one of the parameters for QoE. This method appears to be quite Central Processing Unit (CPU) intensive, because it requires further analysis to evaluate its possible advantages to the intended purpose.

Some other way to have objective metric, which some providers actually use, is to install a device that is able to measure the metrics at the users' side. This can be done by means of some middlebox to allow direct investigation and influence of traffic conditions. However, this method will rapidly prove itself highly challenging due to the addition of end-to-end encryption [40].

Naturally, how objective QoE assessments are performed depends strongly on the service under consideration. In the case of uncontrolled OTT networks, there are QoS and QoE-specific metrics that may be considered when focusing on OTT video streaming.

These metrics can be studied and, using computer simulations or algorithms, the network can be dynamically changed to have a rule on the QoE. For example, in [41], using only Buffer and Bit Rate information, it was possible to study in detail on how to improve a DASH stream performance and its impact on QoE.

2.6.3 Hybrid Assessment

Quantifying the quality of video being transmitted over heterogeneous networks is becoming critical in the field of digital video Communication. Thus, hybrid solutions turn to be an alternative, which involves both subjective and objective techniques. A possible way to improve QoE could be the usage of a neural network model that would be responsible for estimating the relation between QoS parameters and QoE, thus making sure that it would provide a positive final result to QoE [42]. This has been called a hybrid technique, since it can be based on both aforementioned methods. It has a high potential of improving overall QoE, but given the blackbox characteristic of neural networks, it shows very difficult to study and replicate the same results under different conditions.

2.7 QoE Optimization

The study in [43] proposed a model that uses multivariate correlation model to assess the correlation metrics over video services. The model shows a high correlation between metrics QoS/QoE and allows to calculate the MOS more precisely.

"Providing a high quality service presents the most challenging task among the advancements in networking". This information is from a work that presents a machine learning approach combined with adaptive coding in order to provide a better QoE for video streaming services in Software Defined Network (SDN) architectures [44]. This work uses a method that is useful to improve user-perceived video quality. Similarly, the work in [45] approaches the QoE optimization by using machine learning techniques, this time based on the GRadient Boosting method for QoE estimation and Reinforcement Learning for the video quality that maximizes the bandwidth usage, thus ensuring a final improvement in QoE.

However, these approaches do not resolve the root causes of congestion problems. The work in [46] presents an approach where bottlenecks are pinpointed by monitoring the network conditions of streaming flow in real time. Then, with a SDN approach, it proposes a solution by dynamically changing the routing paths using Multi-Protocol Label Switching (MPLS) to provide a reliable video watching experience. This solution approaches a little more to the goal of the present thesis, which is the monitoring of network parameters in order to take actions against the network and invisible to the end user, but still trying to provide a better overall QoE. In this case, instead of optimizing the network, the current solution targets the caching mechanism.

In [47] an approach is presented where clients on an LTE-Advanced wireless network, only with regard for Bandwith, an improvement in different low bandwidth scenarios was achieved by the Mobile edge Virtualization with adaptive Prefetching, effectively diminishing the amount and duration of rebuffers. This will directly lead to a smoother viewing experience and a higher QoE. Different backhaul conditions need potential different solutions, and it is proven that prefetch is able to grant a certain MOS level on bandwidth given enough time in advance.

Similarly, [48] presents a model that uses MPEG-DASH technology, and takes advantage of a cache close to the client in order to minimize the effects of network congestion that can arise from either the core or the wireless edge. The system is highly based on the predictability of available bandwidth for short future periods of time. Some conclusions drawn show that the achieved gains in increased future knowledge window might not justify the additional processing time costs, so a better strategy would be smaller time windows. Even though, by assuming the bandwidth consumption and network congestion has a coherence time, it was possible to increase the effectiveness and stability of the throughput by prefetching the content in a cache server.

Even though the studies in [49] approach multi-video HTTP Adaptive Streaming (HAS), they show that in the operation of HAS players, whenever a player reaches its maximum buffer occupancy, there was an on-off type of behavior in the requested video chunks, that leads to unused bandwidth and potencial optimizations.

It is based on these findings that this thesis aims to prefetch ahead in time with a relatively short time window, and based on the measured network conditions, to utilize unused available bandwidth to prepare the content before it is needed. Besides, metrics other than bandwidth also justify being measured, since they were not used in any found study and may lead to an indirect way to predict the available bandwidth, and increase the used efficiency because they also affect the way content is downloaded.

2.8 Machine Learning techniques in Prediction

Machine learning techniques, especially neural networks are being used to predict the way a human thinks. Based on this principle, the same technique can be applied to predict how a human perceives a video and be used to improve the QoE.

Despite the massive adoption of HTTP adaptive streaming technology, buffering is still the most harmful event for QoE in video streaming. The study in [50], besides improving prediction performance, shows that there is a clear influence of stalling pattern descriptors, in particular, those linked to memory effects and the occurrence of the last stalling event.

Particularly, machine learning is typically used to predict the MOS of a client depending on a future request with variable bitrate. The framework from [45] proposes the Optimized Quality of DASH (OQD) that selects the video quality based on the predicted users Mean Opinion Score (MOS) through a machine learning process. The optimal solution is to use a machine learning method for learning, and consequently minimizing the prediction error at each instant by using QoE influence factors (bandwidth, dropped video frame and video quality).

The paper in [51] proposes a policy-based management framework to enable QoS provisioning over SDN-based networks. By monitoring the network and enforcing policies, the results show possible routing optimization. Moreover, a neural network is used to identify the violating flows causing network congestion. Experimental test-bed results demonstrate the feasibility of integrating the proposed architecture over SDN and show that it outperforms the default SDN in terms of throughput, packet loss rate, and latency.

However, given the positive results in [45], it might be possible to use the QoS metrics as QoE influence factors. The author said itself "In the future work, we shall improve our QoE estimation system by building a larger dataset, and integrating more metrics such as delay, and screen parameters.". The author from [51] also proves this point since it reached the conclusion that it is possible to optimize a network with learning methods just from QoS factors.

In [52], those QoS parameters are assessed and provide exactly an estimation of MOS regarding a browsing experience using neural networks. The QoE value is estimated from five QoS parameters including the average of download bit rates, the average of upload bitrates, the average of latency, jitter, browsing performance rate and streaming performance rate. The results show the good agreement between QoE value from the proposed model and QoE value from the Average Mean Opinion Score (AMOS) applications, and the author reached an error variance of 3.63%.

It is, thus, possible to estimate user satisfaction just from the measured network parameters with relative accuracy. Adaptive streaming consumption shares most of these QoS parameters with network browsing, so the approach from [52] is very interesting and a similar one will be used in this thesis to predict the MOS, and choose the best predefined amount of prefetch to maximize the performance and efficiency.

2.9 Summary

The State of the Art chapter described several important concepts and some use cases deployed in real-world scenarios. It is possible to understand better the evolution of streaming technologies, and how the service providers adapted to such evolution. Also, together with that evolution, the appearance of new players focused on OTT content, the development of OTT multimedia networks, content delivery and multimedia caching systems are exposed.

Then, it presents the multimedia streaming technologies, and how they work, and the evolution through the years.

Finally, the state of the art about QoE Optimization shows some studies' proposals and their results in HTTP adaptive streaming. Most optimizations concentrate on the player itself, and the way the multimedia qualities are requested; however machine learning prediction methods focusing in network optimizations based on bandwidth or other QoS metrics present some impressive results, and provide a good starting point for this thesis.

CHAPTER 3

Proposed Models and Architecture

This thesis aims to implement a system capable of analyzing network attributes in the consumer's closest layers, also known as the Last Mile or Access networks (Table 3.1), which are supported by a 1^{st} tier caching.

By analyzing these, it is possible to achieve optimizations through Prefetching mechanisms in order to improve the final Quality of Experience (QoE) by having the content close to the client as soon as possible, while still trying to minimize the network overhead.

Is it possible to improve the global and long-term QoE. For that to happen, not only the current QoE should be measured and taken into account, but also the underlying QoS attributes and main network characteristics. Through the way MOS is measured, one can see how the users' perceived quality is an indirect consequence of these network characteristics. Thus, one can also adapt the behavior to each used technology.

Live content appears to be the most difficult to assure a high QoE level, given its real-time nature and small room for content buffering. As such, the best-found method that guarantees some improvement over the used technologies is to have the initial playout delayed by a few seconds. This delay allows for some prefetching and buffering to be done, thus trying to minimize problems in the access network, giving a margin equal to the amount of delay for the network to adapt and try to serve the content as smoothly as possible.

For Video on Demand (VoD), given that the entire content already exists, the system can deal with potential issues and prepare a solution much further in advance. It is easier than live streaming, and there is a wider margin to apply techniques and methodologies to improve the QoE of large groups of users. We can, for instance, apply content prefetch with several seconds, minutes or even hours in advance, as opposed to the impossibility of such in Live content. Given the most widely used technologies, we opted to choose the biggest drawbacks of each, to allow our QoE estimation to take those into account as early as possible, those being latency, jitter, packet loss and bandwidth.

Attributes	**FTTH**	**Cable**	**xDSL**	**3G/LTE**	**Satellite**
Delay	Very Low	Low	Medium	High	Very High
Jitter	Very Low	Low	Low	High	High
Bandwidth	Very High	High	Medium	High	Medium

Table 3.1: Last Mile technologies

3.1 Proposed Models

Considering the previous considerations, the proposed solutions for each network characteristic are the following:

- When high delays are present, it is ideal to have a large prefetch amount. Assuming the best case scenario, where the content can still be downloaded at high speeds but the delays might be an issue, it is possible to understand that, even if the content arrives with delay, as soon as it starts arriving, large amounts of data will arrive altogether. This technique can potentially be used for Live streaming, assuming a few seconds of delay that can be buffered in the client's device.
- When high jitter scenarios are present, it appears that one could also prefetch in advance, since the behavior of a jittery network resembles the one of a high latency network. The earlier local preparation of the content is what prefetching aims to achieve here.
- When low bandwidths are present, we believe that the most beneficial approach is to use the least prefetch possible. A network which is already congested will become even worse if there is more concurrent traffic in itself. When compared to high delays, this brings the issue of needing a very long time to download the data completely. Even if the communication channels between the upper tiers of caching have low latencies, the content might take too long to download, leading the client to a buffering event. This weakness is one of the reasons why it might be challenging to improve QoE when assuming either Live or VoD transmission.
- Packet Loss also appears to be an excellent case to use prefetch, since lost packets could delay a real-time transmission. It is highly variable, especially in wireless technologies due to interferences, and even in cabled networks, due to dynamic route changes and real-world influences, it could be present. Prefetch may address and in part deal with this. The lost packets can be requested as early as possible, and this process can be repeated until all data arrives.

With this in mind, this Thesis aims to create a model which takes into consideration the different QoS characteristics and behaves accordingly, by prefetching more or less amount of video depending on the current measured values. This amount shall dynamically adapt with the network variations in constant intervals.

3.1.1 Analytic Hierarchy Process Approach

An Analytic Hierarchy Process (AHP) approach shall be investigated as the first step. Keeping in mind that there are network characteristics with various degrees of importance, it is crucial to find a method that allows calculating a weight into each one of those. Also, due to the complexity in giving a specific weight to a parameter, an approach where a human can decide between a range of importance will be chosen.

The AHP is a method that allows making complex choices. It is not a way to obtain the best decision, but instead, a way to help comprehend a problem through a mathematical approach (Figure 3.1).

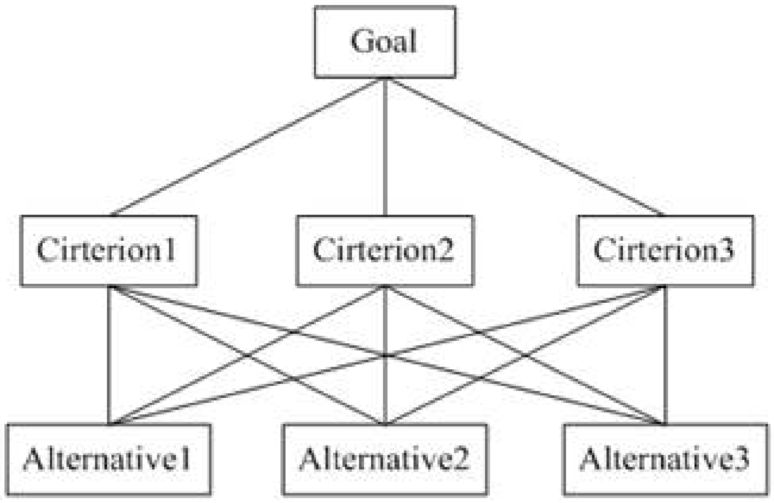


Figure 3.1: Analytic Hierarchy Process overview

First, the main problem is divided into elements that can be analyzed independently. In this case, each network characteristic. Then, the elements are compared two at a time, where to each pair, a numeric score of importance against each other is attributed. Finally, a matrix of importance between pairs is built. From this matrix, it is possible to calculate the overall weight of each element against the others through the application of algebraic methods.

The calculated weight vector can be applied to the measured elements, in order to have an ideal amount of prefetch. However, to have comparable values, every measured element must be normalized so that the weights can be equally applied.

Having that in mind, the AHP method is expected to calculate the ideal amount to prefetch, having the basic notion that each metric will influence positively or negatively the final MOS result.

3.1.2 Self-Learning Approach

Considering that there is much information in these tests to be dealt with, and that each network condition may have a different effect on the final MOS perceived by the user, as well as different units that are not directly comparable, besides testing the AHP approach, one should be able to find a dynamic technique to calculate different weight functions for each different condition, that is not based on a subjective human opinion. It is understandable that for ideal results, the method requires more than a simple mathematical strategy, but it requires a deeper statistical technique.

By virtue of the challenging problem that is being dealt with in this document, and the difficulty of creating algorithms to correctly weight the total variability of the testing data, it is possible that a better approach to this problem would be the use of artificial intelligence. More specifically, machine learning allows the use of statistical techniques over the data without being specifically programmed.

An artificial neural network is a type of machine learning algorithm that tries to model the way the human brain works. There are several types of learning paradigms, supervised learning being one of the major ones. Supervised learning is based on a minimization of a cost function, which is typically a mean-squared error. In other words, there is a given dataset of inputs and outputs; then, the neural network tries to adjust the weights of all its multilayer paths to minimize the error. This process is known as backpropagation

The mentioned technique is very good for this approach, since it is possible to previously generate all the data with simulations, feed that data to the neural network, and finally verify how close from reality is the result.

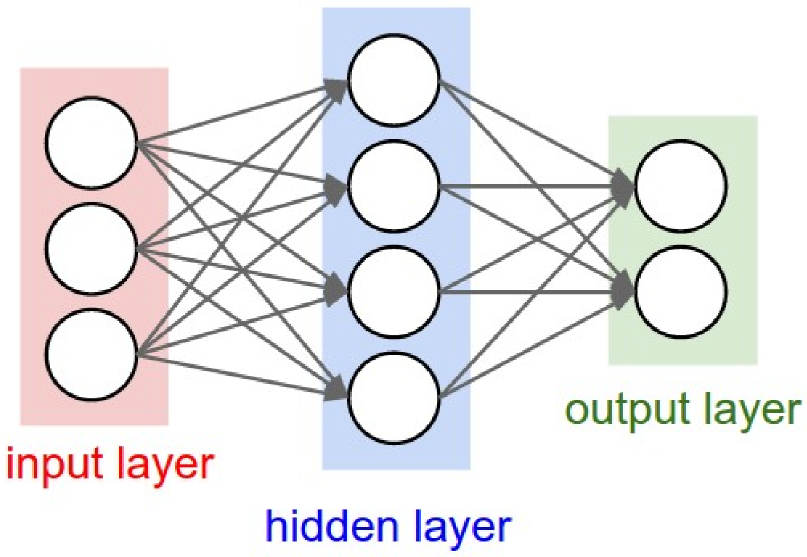


Figure 3.2: Simple Neural Network

Figure 3.2 shows how a simple neural network is organized. It has an input layer, also called features, hidden layers which decide how deep is the neural network, and an output layer also called labels. The input layer provides the test conditions to the neural network. The hidden layers are where the weights and correlations are automatically and randomly calculated by the neural network. The output layer is the output, in this case the calculated MOS.

By definition, the input layer will consist in the four networking conditions mentioned earlier, and also the amount of prefetch that is currently being applied. This allows for the output layer to predict a MOS value. This way, after the learning process, it is possible to test different amounts of prefetch and, in a short amount of time, get a MOS prediction. Also, by using prefetch as one of the inputs, it allows for the cache running the prefetching mechanism, if applicable, to predict the MOS within a limited amount of prefetch instead of basing the decision on a static predefined range.

3.2 Proposed Architecture

A 2-tier caching mechanism is the choice for the proposed architecture. It is one of the common architectures in use nowadays by CDNs. It is ideal for the approach to be as close to the user as possible, in such a way that QoS issues affecting the access network and the streaming content might be invisible to the user.

A client can be traveling on the street and have its cellphone working as a cache and client at the same time. This way, the technology is still working behind the scenes and entirely invisible for the client. The technology can also be applied, for example, to cars which nowadays have integrated hotspots and could be working as the first cache after the client, to minimize issues between the car's internal network and the outside base station over wireless technologies. Another example, and the one that is going to get more focus in this Thesis, is train travel where there is an internal Wi-Fi network, passing through several cities, with connections to the outside world over 4G/LTE which have variable conditions depending on the location, speed, and other characteristics. A client at home with Cable internet in a big city, at peak time, will also suffer quality degradation which will eventually lead him to a worse multimedia experience. These are some of the reasons why this Thesis approaches the caching closer to the client in such a way that it becomes technology agnostic. If improvements can be made between the first and second tier caching, they might eventually lead to a final improvement, and thus the goal will be achieved.

With all this in mind, the aim is to have a solution which is capable of improving QoE for the final user. At the same time, it should be easy to implement on small devices which sometimes are low-powered and without much processing power and memory. At last, it should be modular enough so that, with the appearance of new techniques, the needed changes to adapt such architecture and support new technologies are minor.

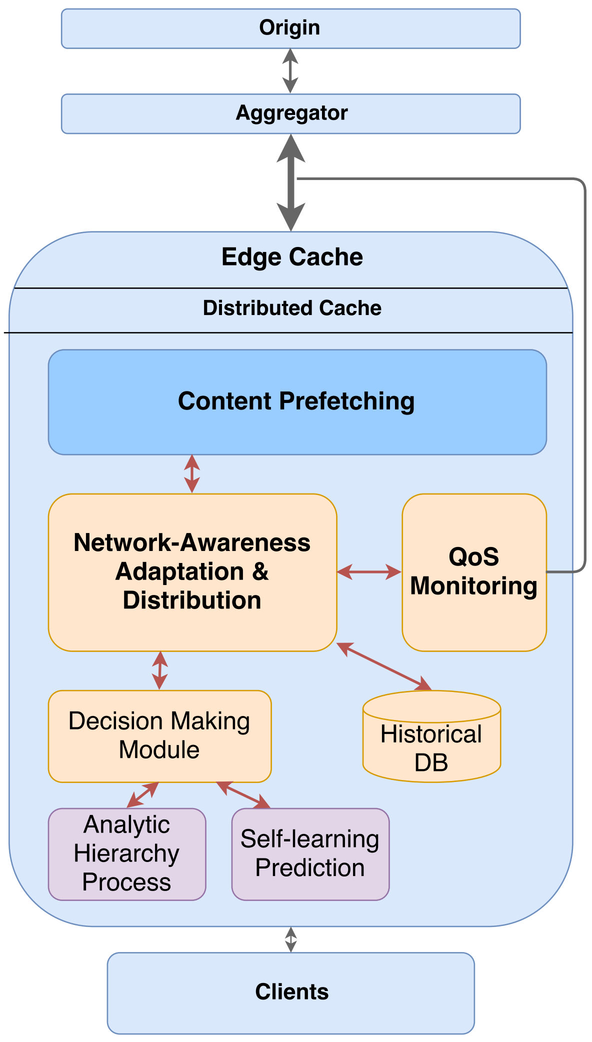


Figure 3.3: QoE Based on QoS Network-Awareness Architecture

In the proposed architecture depicted in Figure 3.3, the caches use Nginx [25], which is a free, open-source and high-performance HTTP server with reverse-proxy and caching technology. The caches get the content from an Origin server which is a Microsoft Windows machine running Internet Information Services, and is able to provide Microsoft Smooth Streaming content.

The solution has a main module called Content-Awareness Adaptation and Distribution. This module is expected to receive information from the QoS Monitoring module, and either process it or store it in the Historical Database for further review. Every action taken by every module and every measured QoS value is to be recorded in the Historical Database, so that the AHP and Self-Learning modules can use any previous values.

As depicted in Figure 3.3, there is a QoS Monitoring module, where its goal is to measure the real-time characteristics of the network and report back to the Network-Awareness Adaptation & Distribution (NAAD). As said previously, it is one of the objectives to have content closer to the user, as such the closest cache is the target, and the connection between that cache and the upper tiers is where the QoS is measured.

The Decision Making Module can either choose between no improvement at all, a static approach where AHP is used to get the normalized weights for each measured parameter, or a dynamic one where a self-learning approach is used to get an ideal prefetch amount, based on the same measured characteristics. The NAAD module might then configure the prefetch and potential cache distribution, based on previous self-learning or AHP iterations, to dynamically adapt and try to provide the overall users with higher QoE.

Also, this solution might have a distributed cache on top of it. As such, the Content Prefetching Module should be able to manage individual multimedia chunks, so that it is transparent for the distribution layer, and it may be able of handling any requests without any major changes.

The Origin and Aggregator are not meant to experience optimization by the algorithms, since they stay on the service provider's core network, or even on the content maker's network itself. They are mentioned and used in the Thesis to replicate an entire distribution network, although all technology is applied explicitly in the closest tier.

The clients are a single user simulation that has a behavior just like a typical adaptive multimedia player, and at the end gives a score regarding how good or bad the multimedia viewing experience was. These clients shall not directly intervene in the caches' technology, since that would mean different devices would need different and specific multimedia players. It is expected to observe variable behavior through time, which means that someone watching the multimedia content will give a variable score depending on the conditions. The best case is to maximize the mean opinion score (MOS) for as long as possible.

At last, the content prefetching module has the sole purpose of receiving a number of seconds or amount of chunks, and translate that into actual requests in advance. The module must be able to intercept current requests and cache the next few moments of video without interfering with the client or the upper caching tiers. Still, some form of communication with the local cache must occur, so the cache is able to keep this content until the client requests it.

3.3 Summary

This chapter takes into consideration what was presented in the State of the Art, and proposed two models that approach QoE improvements.

The AHP model is based on weighting different QoS metrics through a subjective evaluation, where two metrics are compared against each other using a fixed scale, and finally, through a mathematical approach, the final weight scale is calculated and applied on a function that returns an amount of prefetch.

The self-learning model uses the same strategy, although the learning process is not subjective, but is made through testing with exact QoS and MOS values.

The next chapter presents a possible implementation, and how the different modules interact.

CHAPTER 4

Implementation

This chapter details the implementation of the proposed approach described in chapter 3. For this purpose, it starts by describing the testbed implementation, the architecture, and finally the different approaches to determine the optimal amount of prefetching.

4.1 Testbed

The architecture described previously must be supported by a real testbed. This testbed must allow for network communication, for fast processing, for multiple clients to run tests at the same time, and a backend that controls the entire system.

Since this architecture involves many modules and different machines that may run different operating systems, a physical server is an excellent choice compared to separated physical machines for the simplicity of implementation.

The server must support virtualization, so the different machines can be replicated as Virtual Machines (VMs) and given different virtual hardware depending on the objective of said machine.

VMWare ESXi was the chosen bare-metal hypervisor. The choice of this operating system is due to be primarily used on the enterprise level and allowing for simplicity and customization when creating VMs. It is, thus, effortless to create, delete and alter any virtual hardware specification of any machine even after creation and software install. Besides, there are large support communities with years in experience and knowledge in this system, which verifies its stability and performance.

Given that there are many simulations, and results to analyze, concurrent tests is one expected purpose of the system. For this reason, the system must have several VMs with the aim of replicating 1 Origin, 1 Aggregator, 3 Edge caches, and 3 Clients (Figure 4.1).

The physical hardware supporting all this is a server with 32GB of RAM, 32 logical cores and a 10Gbps network card.

The choice of hardware division is as follows:

- Origin: 2 Intel Xeon cores with 6GB RAM
- Aggregator: 2 Intel Xeon cores with 2GB RAM
- Edge caches: 3x 2 Intel Xeon cores with 2GB RAM
- Clients: 3x 2 Intel Xeon cores with 2GB RAM

The reason for the multiple Edges and Clients is to allow being able to run more than one test, in this case three, and still have enough virtual hardware available to allow the VMs to be multi-purposed.

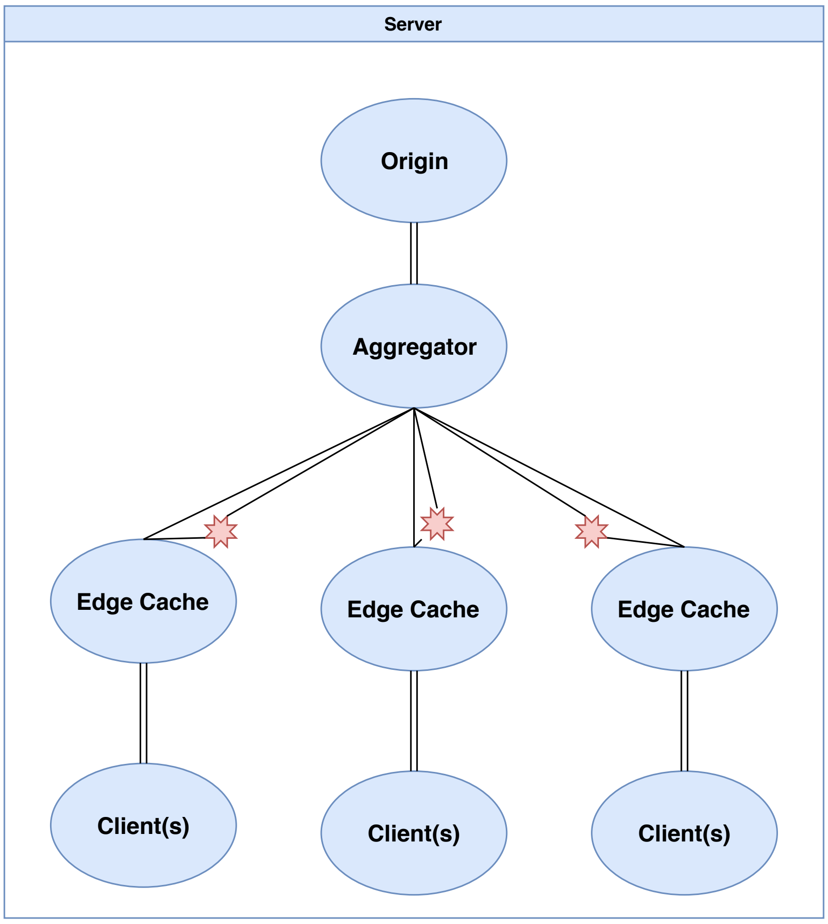


Figure 4.1: Testbed

Due to the goal of running tests with different networking characteristics, two virtual network interfaces are required. Therefore, every VM will have two subnets, one for testing and another for control. As shown in figure 4.1 only one of the network interfaces of the edge caches serves the purpose of simulating network interferences in each test. Thus, not only potential limitations do not affect the control network, but also in this way there is a guarantee that no outside influence can affect the test results. As such, each machine will be configured with two 1 Gbps virtual network cards.

4.1.1 Origin

The Origin is the Virtual Machine (VM) with more virtual hardware since it needs to be running Windows. In this case, it is Windows Server 2012, which has the following minimum requirements:

- Processor: Minimum 1.4 GHz 64-bit Processor
- RAM: Minimum 512 MB
- Disk Space: Minimum 32 GB

Even though the requirements appear to be low at first, extra processing and memory capabilities are added in the VM creation, not only because minimum guarantees under peak usage are essential to safeguard, but also because it must contain special software to allow Smooth Streaming distribution which requires extra processing power.

IIS Media Services 4.1 is an integrated HTTP-based media delivery platform, which integrates with the Windows Server's IIS, and grants access to the needed On-Demand Smooth Streaming modules via HTTP GET Requests. This platform is installed in the VM, so the HTTP requests are replied with smooth streaming content.

4.1.2 Aggregator and Edge Caches

Both the Aggregator and Edge Caches have the same virtual hardware and also the same software. This decision is important because it allows for the optimization technologies to be used not only in the Last Mile close to the user, which this Thesis will approach, but also anywhere in the ISP's core network.

As Operating System, a free open-source Linux distribution system is the choice. Therefore, Ubuntu Server (CLI) 14.04.5 Long Term Support (LTS) is installed in both levels of the cache. This operating system allows for rapid and simple software development, which is essential for this Thesis due to the various and different systems involved.

In this case, the chosen distribution has the following minimum requirements:

- 300 MHz x86 processor
- 256 MiB of system memory (RAM)
- 1.5 GB of disk space

It is obvious the lower amount of requires resources compared to the Origin's operating system. These machines will run a caching system (NGINX), and the caches alone will run all the desired optimization software for the proposed models.

4.1.3 Clients

The Client shall also run a similar open-source, command line interface system. The reason is that it is intended for this VM to be controlled remotely through the network, and Ubuntu Server allows this to happen upon installation without any special software, and any graphical user interface would be a waste of resources.

The client VMs consume multimedia content and give it a score. Extra virtual hardware is assigned at creation, so that any potential multimedia disruption does not happen due to resource depletion.

4.2 Probe and QoE estimation

Currently an implementation of a Probe [36] that analyses the Quality of Experience (QoE) has been done. This probe behaves like a consumer, which requests the Adaptive Streaming content. Given the characteristics of the network at the time, the QoE might improve/decrease due to heavy buffering, lowering bit rate or low playback Frames per Second (FPS), and the probe gives a final score based on MOS heuristics to each 2 seconds of video play.

The probe's implementation was highly based in the users' opinion and the inherent video quality. This means that the MOS result is due to the Bitrate, Frames per Second, Rebufferings and Screen resolution ratio. Also, different weights are given to each of these attributes, since they influence the viewer in different ways.

Furthermore, given that the human brain reacts differently to a stream session, those values were used in an equation to evaluate each chunk (in this case, 2 seconds of video) with a sampling rate of 10 samples per second (10Hz) with a later addition of a Human Memory Filter. An Infinite Impulse Response (IIR) filter was used to simulate the Human Memory, and thus approximate the MOS to the real user opinion as best as possible. This is due to the way the human brain works. The goal is to simply emulate the behaviour taking into account that the more recent events have a higher influence in the users' opinion than older ones.

An assessment was conducted with real users so that the proposed model could be benchmarked against real-life results. The assessment was conducted in accordance to International Telecommunication Union (ITU) recommendations.

It was then possible to prove that the MOS estimates from the survey fall in line with the estimates provided by the QoE model. It was also noticeable that the results were closer when the content was a stream of an animation video when compared with a sports stream. This was due to the fast moving scenes in sports videos that leads to a usually harder to estimate picture quality.

Because this probe appears to have excellent results in simulating a group decision on multimedia QoE, it will take part in this Thesis for the single purpose of providing a score.

4.3 Network-Awareness Adaptation and Distribution

NAAD is the main module in the entire architecture. It provides control over the whole simulation and is capable of connecting to remote machines through the dedicated control network. Figure 4.2 shows a flowchart corresponding to the simulation's life-cycle.

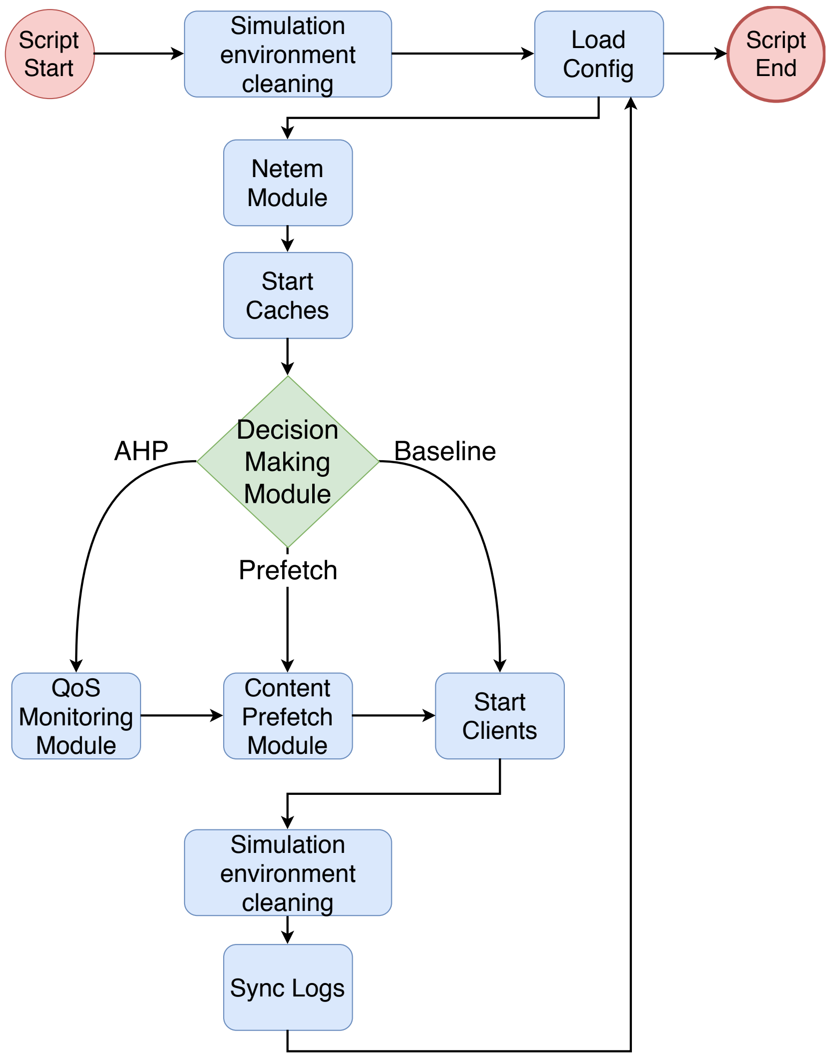


Figure 4.2: Network-Awareness Adaptation and Distribution life-cycle flowchart

Assuming that each VM has identical script configurations and that each script is located in the same path, the NAAD Module may be started from any machine in the network. By having a list of IP addresses associated with each VM, the module sends the expected shell scripts over Secure Shell (SSH) to start the simulations under the predefined order.

4.3.1 Environment cleaning

The first step is making sure all the machines have no active processes besides the basic operating system. This way, each run may start from a clean state. So, with a list of all the processes involved in the runs, a bash script will send "pkill" commands to every involved machine and it will force the termination of every single active process.

It works both ways, which is softly stopping all processes after the experiment, and in case any unusual action has happened and, for some reason, the experiment did not safely end, forcibly stop them, therefore, making sure that they will be fresh and clean start on the next run. On the aggregator, it is just the process related to Nginx; on the cache, it is the

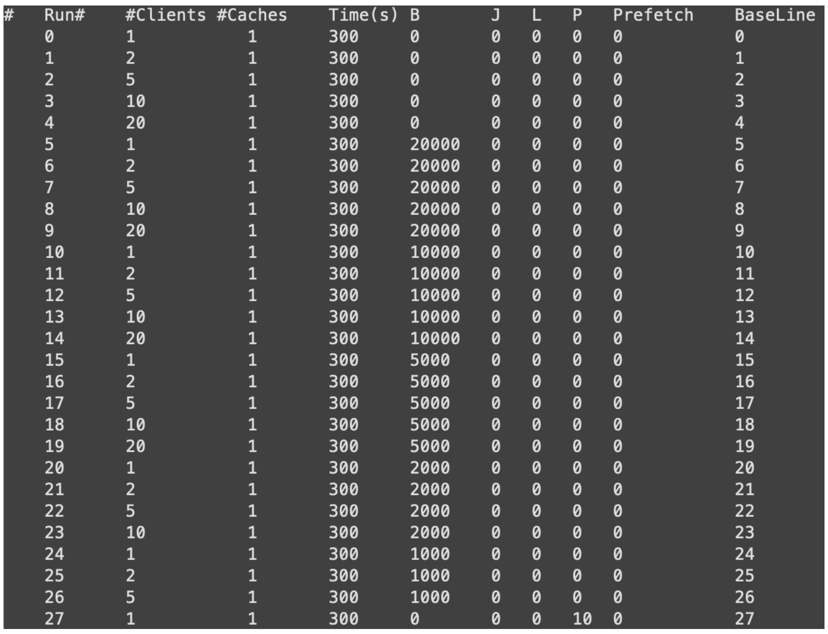
Nginx, and potentially the QoS Monitoring and content prefetch processes; and finally, the client VM contains the probe process.

This method has a blocking state to advance only with the assurance that every process has safely terminated. Besides terminating every process, this method shall also make sure that no limitations exist in the active testing network. As such, the Netem module, explained further ahead, should execute a call with the default values, so the connection between the Aggregator and Edge cache may be free of any hard limit previously imposed.

Regarding the log cleanup, the objective is to assure that a new run has a clean slate without regard from previous executions. For every edge cache, client and aggregator, both the Nginx and probe logs are erased by the script. By having this happening, at the end of each run, it is essential to copy every file and keep it in a separated registry for historical purposes.

4.3.2 Loading Experiment Settings

The Load config module has specific methods for loading a predefined file with specific instructions for the experiments. This file must have well-defined parameters consistent with the test itself. An example of file is available in Figure 4.3.



```
#    Run#    #Clients #Caches    Time(s) B        J   L   P   Prefetch    BaseLine
     0       1           1       300     0        0   0   0   0           0
     1       2           1       300     0        0   0   0   0           1
     2       5           1       300     0        0   0   0   0           2
     3       10          1       300     0        0   0   0   0           3
     4       20          1       300     0        0   0   0   0           4
     5       1           1       300     20000    0   0   0   0           5
     6       2           1       300     20000    0   0   0   0           6
     7       5           1       300     20000    0   0   0   0           7
     8       10          1       300     20000    0   0   0   0           8
     9       20          1       300     20000    0   0   0   0           9
     10      1           1       300     10000    0   0   0   0           10
     11      2           1       300     10000    0   0   0   0           11
     12      5           1       300     10000    0   0   0   0           12
     13      10          1       300     10000    0   0   0   0           13
     14      20          1       300     10000    0   0   0   0           14
     15      1           1       300     5000     0   0   0   0           15
     16      2           1       300     5000     0   0   0   0           16
     17      5           1       300     5000     0   0   0   0           17
     18      10          1       300     5000     0   0   0   0           18
     19      20          1       300     5000     0   0   0   0           19
     20      1           1       300     2000     0   0   0   0           20
     21      2           1       300     2000     0   0   0   0           21
     22      5           1       300     2000     0   0   0   0           22
     23      10          1       300     2000     0   0   0   0           23
     24      1           1       300     1000     0   0   0   0           24
     25      2           1       300     1000     0   0   0   0           25
     26      5           1       300     1000     0   0   0   0           26
     27      1           1       300     0        0   0   10  0           27
```

Figure 4.3: Simulation Configuration File

The config file has a single line for each independent run. Column separation is achieved through spaces or tabs, and the order of the values is important. The parameters start in the first column and, sequentially, follow the following specification:

- Run# = Experiment number. A way of identifying a group of experiment settings and the test number. Experiments follow a chronological order based on this value. The value must be unique.

- #Clients = Number of parallel clients running the test. In a single VM, this value corresponds to the number of concurrent probes executing the experiment. The value must be greater or equal to 1.
- #Caches = Number of distributed caches to serve the clients. A group of parallel clients can use single or multiple (distributed) caches. This value specifies the number of caches for a single experiment. The value must be greater or equal to 1.
- Time(s) = The duration in seconds of the experiment. The value is used to adjust, not only the time while a client is watching the multimedia content, but also the duration that Edge caches and Aggregations stay active and wait for requests. After the specified amount, it is safe to assume that the experiment has finished and the NAAD may proceed to backup all the logs and stop all the involved machines.
- B = Maximum Bandwidth in Kbps to apply to the simulation. This setting is used to introduce bandwidth limitations between the edge cache and aggregator. NAAD applies this value earlier than client multimedia consumption, to simulate real networking conditions.
- J = Jitter in ms to apply to the network. This setting, similar to Bandwidth, is used to apply limitations to the network. To better simulate actual conditions, it follows a normal distribution centered on the latency, and this value is the variation amplitude. Ex.: Latency 100ms with Jitter 10ms, means actual latency of the network will be 100ms±10ms.
- L = Average latency to apply to the network. This has been explained in the previous setting. These two settings follow an approximation, meaning that a small percentage of values may fall out of range, although most of the values are according to the specification.
- P = Packet loss in percentage to apply in the network. A random number of packets based on the value of this setting out of every 100 gets dropped.
- Prefetch = Value to set the Decision Making Module properties. This value is in the range [0,100]. If the value is equal to 0, then nothing is applied, and the Baseline test is assumed. If the value is higher than 0 and lower than 100, then normal prefetch is assumed, and the value is then sent to the prefetch module. If the value is equal to 100, then AHP is expected.
- Baseline = This value is for graphing purposes. If this value is different than the Run#, it means that it should be plotted together in the same image as the value here specified.

This concludes the specification of the config file.

4.3.3 Netem Module

The Netem module has the single purpose of setting the loaded configs regarding bandwidth, jitter, latency and packet loss, and apply the specific shell scripts via SSH to each Edge cache. It contains the usage of Iproute2 which is a collection of tools that provide control over the TCP/IP networking and traffic control to the system.

The focus of this module is over the traffic control utility, which allows for the emulation of properties of wide area networks. Not only it is possible to emulate variable delay, static delay, loss, but also duplication and re-ordering of packets. Token bucket filter is a queueing discipline available in linux for traffic control (tc). This discipline allows specific settings that are useful to limit the bandwidth in the experiments. With this in mind, the module first clears the system of any previous configs, and right after it sets the new ones.

An example setting with a maximum bandwidth of 4Mbps, 100 ms Latency, 10ms jitter, and 5% packet loss is as follows:

"tc qdisc add dev eth1 root handle 1: tbf limit 4Mbit rate 4Mbit burst 10K"

"tc qdisc add dev eth1 parent 1:1 handle 10: netem loss 5% delay 100ms 10ms distribution normal"

4.3.4 Starting Caches

The process to start the edge caches and aggregator is quite simple. First, it is necessary to modify the main Nginx configuration file.

The needed changes involve setting the proxy_pass parameter, which is used to define the upstream server, in other words, letting the Nginx cache to know where to try and get the content whenever it is not locally available in memory. Also, it is possible to define the cache log format and the location and size of the local cache. These are the only mandatory requirements needed for the experiments. The Nginx cache has documentation detailing the hundreds of adjustable settings and ways of configuring the cache itself available on the platform's website. Each edge cache sets the Aggregator as upstream server, and the aggregator will set the Origin server as its own.

After changing the desired settings in each VM's Nginx configuration file, the Nginx cache itself is now started with the new settings by calling the installed service.

4.3.5 Decision Making Module

Several tests must be conducted with the variation of each QoS parameter, and with the variation of multiple parameters altogether. This not only allows to assess the correlation between QoS and the final MOS in the proposed architecture, but it also asserts other studies' conclusions that affirm the indirect effect that QoS metrics have on QoE. Upon the realization of those tests, it is feasible to deduce the importance of each QoS parameter in the final MOS, thus leading to the creation of a weighted scale.

The fundamental Decision Making Module role is to provide the number of video chunks to prefetch in advance basing the decision on the stipulated methods. This module is strictly related with the QoS Monitoring and Prefetching modules, since it receives the network QoS metrics to output the chunks quantity number.

Besides, the decision making module has a network-based communication protocol to be able to receive data concerning the current networking values. By defining a communication protocol, modularity and interoperability are achievable, meaning that the technologies used to report and monitor the actual conditions may change, but as long as the protocol is respected, the system will continue to work in the same way. Modularity is also important, so the capture methods may work in any operating system, and interoperability is also a requirement, since they may operate anywhere in the system architecture.

This protocol is expected to work under UDP and has four socket listeners in different UDP ports, one for each collectible metric as shown in Figure 4.4. The four defined ports and its units are as follows:

UDP:10001 = Bandwidth (Kbps)

UDP:10002 = Jitter (ms)

UDP:10003 = Latency (ms)

UDP:10004 = Packet loss (%)

Every measurement is expected to be received as a string and then converted into float for precision. A specific arrival frequency is not required; however a 10-second average is applied to smooth the values, and the resultant is both saved internally for consumption and into a file for historical record.

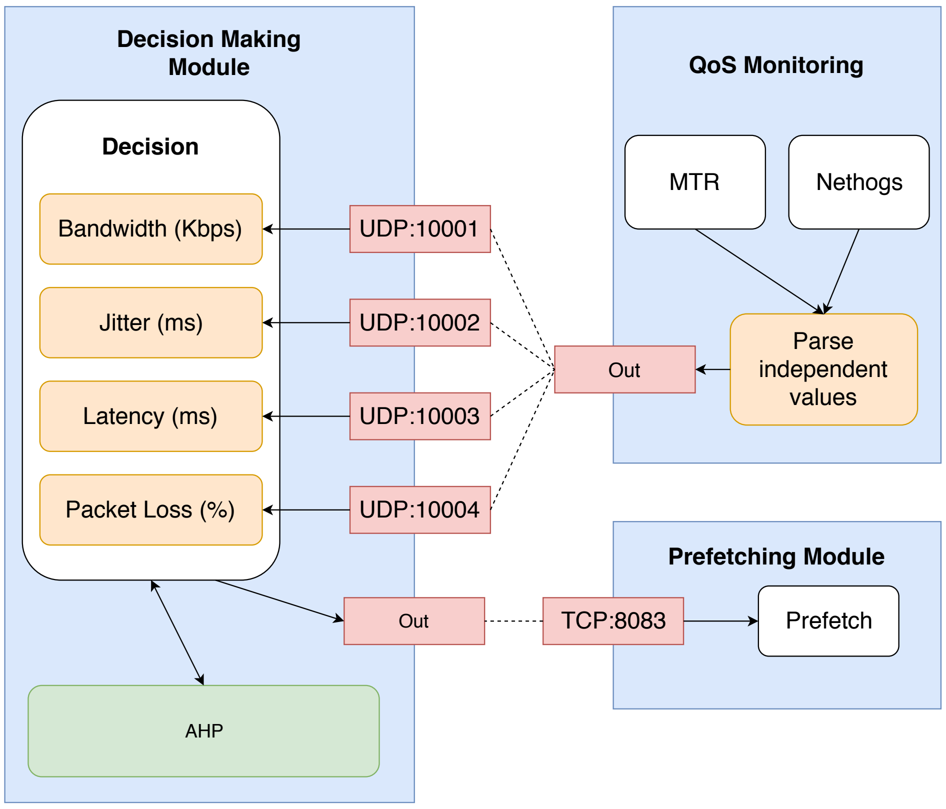


Figure 4.4: Networking Communication Protocol between Modules

In figure 4.4 it is possible to see how the different modules interact with each other through the network. Ahead in chapter 4.3.6 and 4.3.7, it is detailed the purposes of the respective modules and how they interact with the Decision Making Module.

4.3.6 QoS Monitoring Module

The QoS Monitoring module is strictly used together with the AHP approach. Both the baseline and fixed prefetch do not need to be aware of the network conditions, since they either always prefetch the same amount or do not prefetch at all. With this in mind, the QoS module must have methods that allow for independent condition measurements.

Two tools for QoS monitoring were used, namely Nethogs and MTR. Nethogs is an open-source tool, available for Linux, which groups the bandwidth usage by process. This tool is lightweight and allows an input parameter to specify the update rate, which ideally should match the 10 seconds refresh rate of the decision making module, causing linearly less traffic than sending with a higher rate. So, every 10 seconds a new measurement of bandwidth is taken and transmitted through the network via the following shell command:

"nethogs -d 10 eth1"

"mtr combines the functionality of the 'traceroute' and 'ping' programs in a single network diagnostic tool.". This tool's purpose is investigating the conditions of a link between two hosts. Such a mechanism is ideal for the goal of the QoS module, since the aim is precisely diagnostic and measure the link characteristics. In particular, latency, jitter and packet loss are the goals of the module.

Thus, the QoS Module executes the tool with the following parameters:

"mtr -r -oJNL -c 10 -B 0 Aggregator_IP"

It is a little intrusive given the fact that, to calculate the packet loss, the tool must send actual packets. To achieve this, the parameters "-c 10 -B 0" are added which make the tool send and expect to receive 10 Internet Control Message Protocol (ICMP) Request and Replies. Each request or reply is 78 bytes which translate into 780 bytes sent and 780 bytes of received overhead every second. The packet loss resolution used is 1/10, meaning that each step in the measurable packet loss is 10%. Having better resolution is possible; however, it is important to keep in mind that for higher resolution, there is higher overhead.

Latency and jitter are measured taking into account the timestamps between sent and received packets. And again, the QoS module transmits the three measured values over the network to the predefined ports with a 10 seconds rate.

4.3.7 Content Prefetch Module

Content prefetch module has the purpose of requesting chunks in advance. In [1] it is shown that positive results are possible, and in some instances, QoE MOS can increase with this technique.

The way this module works is by sniffing HTTP traffic going into the Edge cache, and through the content's Manifest it check the timestamp of the chunks ahead, and request those. By asking in advance, it means that the chunks are expected to be cached before the needs of the client, and thus upon the client request, they can be promptly delivered. Besides, there is more margin for errors ahead of the actual video consumption necessities, allowing for some form of control over bad network interferences.

Moreover, this module has a listening TCP socket which accepts a numeric integer value which will be translated into prefetch amount.

4.3.8 Starting Clients

The start clients' module has the purpose of checking how many clients are supposed to be parallel tested from the configurations file. With this number and a static list of clients' IPs, the module remotely connects to each client and executes the probe with the intended duration. After the remote execution, it will block execution for the same amount of time as the most extensive test's duration. By blocking for the said duration, it allows for the experiment's life-cycle to continue only when all the tests finish.

4.3.9 Sync Logs

Given that there are different VMs running different scripts and activities, the Sync Logs module backups and clears all the outputs and logs from individual experiments.

First, from the Aggregator, the Nginx cache logs are backed up, and these logs contain each HTTP requests information just like the two following examples taken from the aggregator log file:

1. "192.168.122.116 - MISS [04/Jul/2018:05:04:34 +0000] "GET `http://192.168.122.121:80/development/smoothstreaming/train/output.ism/QualityLevels(962000)/Fragments(video=800000000)` HTTP/1.1" "prefetching_Bot" "360681" "0.040""
2. "192.168.122.116 - HIT [04/Jul/2018:05:04:35 +0000] "GET `/development/smoothstreaming/train/output.ism/QualityLevels(2962000)/Fragments(video=800000000)` HTTP/1.1" "-" "360681" "0.000""

The Log file may be customized to add more or less information according to the needs. In this case, the planned information fields are respectively:

- Remote IP address: 192.168.122.116
- Cache status: HIT or MISS
- Timestamp of the request's arrival: [04/Jul/2018:05:04:34 +0000]
- Request URL: `http://192.168.122.121:80/development/smoothstreaming/train/output.ism/QualityLevels(2962000)/Fragments(video=80000000)?agent=1`
- HTTP user agent: prefetching_Bot
- Bytes sent: 360681
- Request time: 0.040

Then, for each edge cache, all files from Nginx, QoS Monitoring and Content Prefetch are also backed up, and the same for the clients with the probe output.

With the use of rsync all the files are placed hierarchically in remote shared folders, one for each VM in the architecture, and inside, one for each independent experiment with the following structure:

/mnt/tv/[VirtualMachineID]/[Run#]/

With that, the following examples:

/mnt/tv/aggregator121/run123/NginxCache.log

/mnt/tv/cache116/run124/prefetch.log

/mnt/tv/client29/run20/ProbeOutput.log

After the copy is complete, the module empties all files, so there is no record of previous ones in the logs when new experiments start.

4.3.10 Simulation End

Finally, after there are no more experiments to run, the life-cycle comes to an end, and the script ends. All the results and logs of involved modules are possible to review at any time.

4.4 Analytic Hierarchic Process Approach

To accomplish the AHP approach, values from monitored QoS metrics are used in a weighted average that, following a floor function, calculates the number of chunks. Formally, let

$$M = \{M_1, M_2, ..., M_n\} \tag{4.1}$$

be the set of last monitored values of network QoS metrics, and

$$w = \{w_1, w_2, ..., w_n\} \tag{4.2}$$

the weights for each one of the network QoS metrics, respectively. Equation 4.3 shows how the number of chunks is calculated:

$$n_{chunks} = \left\lfloor (\sum_{i=1}^{n} M_i * w_i) * mc_f) \right\rfloor \tag{4.3}$$

The mc_f represents a maximum chunks factor used to establish an upper limit to the number of chunks, and n represents the quantity of QoS metrics involved. The mc_f factor is needed because values from each one of the monitored and collected QoS values are expected to be normalized between 0 and 1. Consequently, mc_f must be greater or equal to 1.

To normalize each one of the QoS values, a maximum value for each metric must be set up. Thus, when the QoS metric value is monitored according to its own value scale, it must be divided by the metrics maximum value and then sent to Network Awareness Adaptation & Distribution that provides it to the Decision Making Module. At the end, the number of chunks to be prefetched complies to the current network conditions smoothed or raised by the given metric's weights. The floor function is used to conservatively get the integer part, ranging from $[0, ..., mc_f]$, resulting from mc_f multiplication.

Thus, weights comprising equation 4.3 must be normalized between 0 and 1, this way the following equation is true:

$$\sum_{i=1}^{n} w_i = 1 \tag{4.4}$$

These weights are responsible to model the influence of the QoS metrics to the final QoE. All in all, the quantity of chunks is a combination of the current network state and the influence of network QoS metrics on the final user QoE. Although it is possible to leave to a user the responsibility to set up the right weight for each one of the QoS metrics involved, a less arbitrary way should be pursued. In this sense, an efficient technique for calculating each QoS metric weight is using a matrix of judgment [53] found in the multi-criteria decision making method Analytical Hierarchy Process (AHP) [54]. A judgment matrix aims to model relationships (e.g.: importance, necessity, discrepancy, value, etc.) between the judged elements. Therefore, in this case the user provides how important is a QoS metric regarding the others comprising the overall QoE, instead of a weight value.

Thus, the judgment matrix is a $n*n$ matrix, wherein each row and each column represents a different QoS metric, and the values must represent the paired line x column comparison according the Saaty scale values [55], from 1 to 9, where 1 represents the less important comparison value and 9 the greatest one. Thus, all values in the judgment matrix diagonal are 1, and the values from the inferior diagonal are inverted from the upper diagonal.

This work considered the following QoS metrics in order to assess the improvement on user's QoE:

- Bandwidth (B)
- Jitter (J)
- Latency (L)
- Packet Loss (P)

Table 4.1 presents a possible example of judgment matrix with these four different QoS metrics. The pairwise comparison values, from Saaty scale, corresponding to the QoE more influential QoS metrics, comes from the results analysis on a series of experiments performed to compare the variation on the QoE comprising the variation on these QoS metrics.

Thus, in this case, packet loss is 5 times more influential than bandwidth, 7 times than latency and 9 times than jitter. The last line presents the sum of each column element in order to be used in the next step to find the weights, which is this matrix normalization. Table 4.2 shows table 4.1 normalized, with the last column depicting the QoS metrics final weights. To accomplish that, a simple arithmetic average for each line suffices.

QoS Metrics	Bandwidth	Jitter	Latency	Packet Loss
Bandwidth	1	5	3	1/5
Jitter	1/5	1	1/3	1/9
Latency	1/3	3	1	1/7
Packet Loss	5	9	7	1
Col. Sum	98/15 $\approx$ 6,53	18	34/3 $\approx$ 11,33	458/315 $\approx$ 1,45

Table 4.1: Judgment Matrix - Importance between different QoS parameters

QoS Metrics	Bandwidth	Jitter	Latency	Packet Loss	**Weights**
Bandwidth	≈ 0.153	≈ 0.277	≈ 0.265	≈ 0.138	≈ 0.208
Jitter	≈ 0.031	≈ 0.056	≈ 0.029	≈ 0.076	≈ 0.048
Latency	≈ 0.051	≈ 0.167	≈ 0.088	≈ 0.098	≈ 0.101
Packet Loss	≈ 0.765	≈ 0.5	≈ 0.618	≈ 0.688	≈ 0.643
Col. Sum	1	1	1	1	1

Table 4.2: Normalized Judgment Matrix - QoS metric and weight calculation

Considering the QoS metrics used and weights calculated according to the jugdment matrix, equation 4.9 rewrites 4.3 taking into account the actual QoS metrics and calculated weights.

$$w_B \approx 0.20827486744333834 \tag{4.5}$$

$$w_J \approx 0.04799969478329905 \tag{4.6}$$

$$w_L \approx 0.10104391101418733 \tag{4.7}$$

$$w_P \approx 0.6426815267591753 \tag{4.8}$$

$$n_{chunks} = \lfloor (B * w_B + J * w_J + L * w_L + P * w_P) * 10) \rfloor \tag{4.9}$$

In this case, mc_f is considered equal to 10, which represents a considerable number of chunks bearing in mind optimal network conditions and a scalable content distribution scenario, i.e., many consumers requesting video content at the same time from a single Edge cache.

It is very important to note that each different parameter has a different reaction against the QoE and final MOS. Also, these metrics all have different units so it is not possible to calculate an exact weight to be used on the judgment matrix. An approach was taken where the given weights represent not only the real MOS decay along the normalized vector of each parameter, but also the human perception of this interaction.

Since the n_{chunks} is calculated, its value is made available to the Decision Making Module that delivers it to the Content Prefetching module, where it is enforced.

4.5 Self-Learning Approach

With deeper analysis, it is possible to conclude that a variation of a single network condition might cause the different weight functions to be affected on the others.

Similarly to the AHP testbed, the starting condition variations are chosen to define the complete dataset for tests. Since the volume of tests in this dataset is much larger than the previous one, which would highly increase the simulation time in the testbed, a single concurrent client is used per cache, but still with multiple repetitions of the same test to provide a viable average:

- Bandwidth (Kbps): 200, 400, 800, 1200, 2000, 4000, 8000, 16000
- Jitter (ms): 0, 50, 100, 175, 250, 400, 600, 900
- Latency (ms): 0, 50, 100, 175, 250, 400, 600, 900
- Packet loss (%): 0, 1, 3, 5, 7, 10
- Prefetch (chunks): 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

All the combinations give a total of 33.792 unique tests times 5 repetitions of each test, given it is an emulated environment and the results are prone to variations. Thus, it results in 168.960 tests. It is important to notice that each video has a duration of 5 minutes and each test will definitely last a bit longer due to logging reasons and other delays in the life-cycle. It is possible to see that the total duration is extremely large and amounts to at least 586 simulation days.

4.5.1 Testbed modifications

To decrease the total duration of the simulations, the system architecture as used in the AHP testing is changed so it could be optimized for the new testing requirements. It is still based on a client-server structure, but the control is now on a centralized system (Master) and multiple parallel client machines still calculate their own MOS. For this, it is necessary to have an embedded cache on the same client virtual machine, and each of these single VMs now behave both as Edge Cache and Client. This way, it is possible to replicate each edge cache and client virtual machine, so that each one has parallel tests running and do not influence each other.

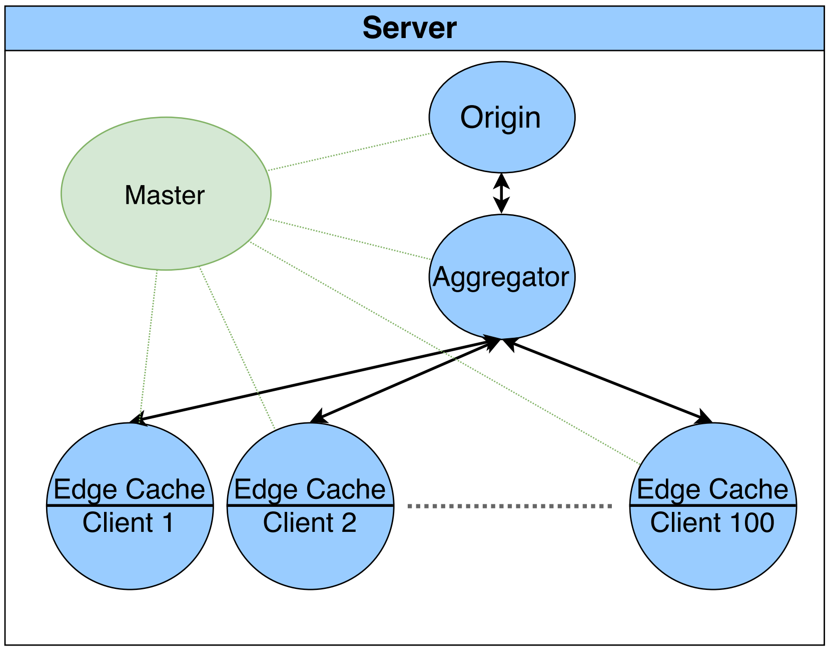


Figure 4.5: Self-Learning Testbed Architecture

These changes do not alter the results ran in the experiments, because the networking variability is still introduced between the edge cache and the client, keeping the original scope of improving the client QoE as close as possible and whenever there are networking issues. So, the final architecture is depicted in Figure 4.5, as it presents the final testbed architecture. 100 virtual machines will be used to run the tests (Clients), giving in practice 6 days of simulation time which is a much more realistic approach.

Each virtual machine runs a minimal Linux operating system. Alpine [56], which is based on busybox, was chosen given its low minimum requirements. After all the software is installed, and considering peak usage, each VM ends up with 1 virtual core, 256MB of RAM and 2GB of disk.

4.5.2 Virtualization environment

To support the virtualization, a different hypervisor was used. Proxmox Virtual Environment 5.1-41 is chosen. This hypervisor has a public Application Programming Interface (API), which allows for an easier development of software to operate the platform compared to VMWare. As such, a script to help the implementation of this testbed was developed, including a cloning script, which can clone a template VM and automatically add it to the available Edge Cache/Client resource pool. The script initially creates a clone of a fully configured VM. Then, the VM is booted up, and given a new ID and IP in the subnet so it can become associated with the resource pool. At last, the VM is rebooted and is now autonomously ready to run simulations.

The bare-metal hardware that supports this tested has the following Hardware specifications:

- 2x E5-2640 (8 Cores 16 Threads)
- 128GB of RAM
- 16TB of Hard Drive disk space

This script helps to speed up the process of creating and configuring VMs since it only needs an amount of VMs to create and the whole process is then automatic.

4.5.3 Flufl Lock

Instead of Nginx being on a separate edge cache VM and the simulated interferences being applied in said VM, it is installed locally together with each Client VM and the restraints can be added in the network interfaces described in black in the Figure 4.5. In Green, the control network interface, which is used to load configurations from an Network File System (NFS), hosted in the Master VM, and for log syncing.

For this testbed, a different approach was taken for the simulation settings load. Rather than having a central VM that injects the tests in single machines, each VM must be responsible for itself and the upper levels of cache hierarchy do not need any setting change in-between experiments. As such, instead of having a single configuration file as depicted in 4.3, there must be in the mentioned NFS share, a file per test. A script will parse the said configuration file into multiple, corresponding to one per test and per line.

Given that the configuration file has a simulation number identifier, the naming and store of new files follow the following convention:

/mnt/data/Tests/{Run#}.todo

For the purpose of randomness with concurrent simulations, the simulation settings load follows a random distribution, being the universe of values every file with ".todo" extension.

Given that many VMs may concurrently access the folder share to load new tests, it is crucial to have a way of guaranteeing that each file is only opened by one instance. "flufl.lock" [57] is a "NFS-safe file locking with timeouts for POSIX systems for Python.". This library is added to the system under the Loading Simulation Module 4.3.2.

To begin, a random test file is chosen. Then, this library creates a "lock" file which has the same name as the original but a ".lock" extension added to it, right after it renames the original extension into ".running". Then, after calling the library's lock method, the file is assumed as locked. An "AlreadyLockedError" exception may be thrown if a race condition occurs. However, by catching this exception, a new simulation file may be loaded and the previous one discarded. At the simulation's end, the library's unlock method unlocks the file, and the original ".running" file finally changes into ".done". This way, every independent Client VM has a way of knowing if each test is to be done, running or finished allowing for safe concurrency.

The library allows the dynamic integration of clients in the whole architecture, and it is thus possible to launch more or fewer clients dynamically, due to the fact that each can run an independent simulation.

4.6 Real-world train scenario emulation

Given the proposed methods, it is essential to match the results with real scenarios, thus assuring that the models can contemplate real-world circumstances and may implement architectures which will be prone to potential improvements due to the conditions.

For this, the chosen scenario is a high-speed train trip between large cities, where inside of the train there is a Wi-Fi network, which is also a caching system, connecting the clients, and an LTE/4G connection connecting the train to the provider's core network. In the Portuguese context, the high-speed train is called "Alfa Pendular" and the chosen cities are Aveiro, Coimbra, and Porto.

Two trips are used for the emulation, one starting in Aveiro and ending in Coimbra, and another larger one from Coimbra to Porto. The first shorter trip connects two cities that distance 57,2Km and the second 120,9Km. The first trip has a duration of 27 minutes and the second 65 minutes.

Since many tests are meant to be examined under these conditions, this thesis approaches it by emulating this scenario. As the first step, local measurements need to be taken, then

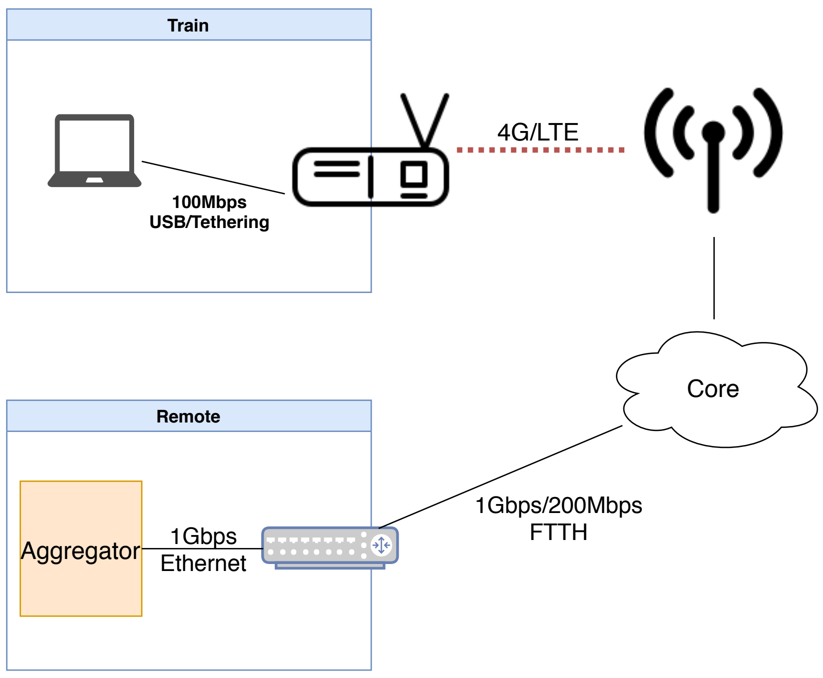


Figure 4.6: Train scenario testbed

they are applied in the deployed self-learning testbed, and finally, results can then be analyzed under the different methods.

The architecture depicted in Figure 4.6 is a LTE/4G mobile hotspot inside a train, locally connected to a laptop via USB/Tethering and connected to the internet service provider (ISP), where it can then measure the metrics to a remote server.

Given that the aim of this architecture is the measurement of parameters between the hotspot and the remote server which are object of technological impairments, USB/Tethering is favored against Wi-Fi for local connection to the Laptop. Also, to minimize routing interferences, the same Portuguese internet service provides was used for both the mobile hotspot and the remote server. The core network is expected not to have any hard-limitations and only being an aim of passive conditions, so in this architecture the remote server is connected to the same operator's core network as the 4G hotspot via fiber-optics with 1Gbps of download bandwidth and 200Mbps of upload bandwidth.

For measurement, two tools are used, one that measures latency, bandwidth and packet loss, was fully developed, and iPerf which measures bandwidth. The developed tool uses *ntplib* which is a Python library that "offers a simple interface to query Network Time Protocol (NTP) servers from Python.". With this library, it is possible to have the tool sync with the current time, and thus each executed action timestamped with precision to the microsecond (10^{-6}s).

Given that two different types of data need to be measured, to avoid potential interferences of bandwidth in the other metrics, two minutes of measurements are repeatedly taken, where one half considers bandwidth, and the other half considers the other three metrics. The bandwidth measurement via iPerf sends TCP data from the remote server to the laptop (Download) inside the train, and the measurements are averaged to the minute.

Upload measurements do not appear to be important in this scenario, since the heavy traffic is downstream due to the video being streamed into the client equipment. The latency measurements are also averaged, although, for precision, they follow a 10hz refresh rate during the whole minute, meaning that each minute of analysis creates 600 data vectors.

Only the *netem* module needed to suffer modifications to add the emulation support in the general architecture. Instead of having a constant setting for the whole experiment, *netem* has been changed to update on a constant refresh rate. For this scenario and because of the measurements period, the *netem* module updates every two minutes with new values. All the other modules stay the same.

4.7 Conclusions

This chapter proposed an architecture and its different modules integration with support for the testing of the models from Chapter 3.

The architecture supports individual experiments and the way its organization allows for concurrent tests with specific individual network QoS settings. In the end, there is a report with the MOS from the whole experiment.

Each experiment may run with no optimizations, fixed prefetch, AHP or with self-learning. It was, thus, possible to achieve a lightweight architecture, and simulations may run in small VMs without incurring in bottlenecks.

The next chapter will present the results that each of those optimizations unveils.

CHAPTER 5

Proof of Concept and Evaluations

This chapter describes the tests and discuss its results. Every optimization approach is compared with previous ones and against the baseline. The baseline used is a test precisely with the same conditions as the one being compared, except for the prefetch amount, which is none.

5.1 Impact of QoS in the MOS

This section presents measurement results on how the availability of QoS metrics influences the MOS. These measurements have been taken from the average MOS of tests with a 5 minutes video and a single concurrent client.

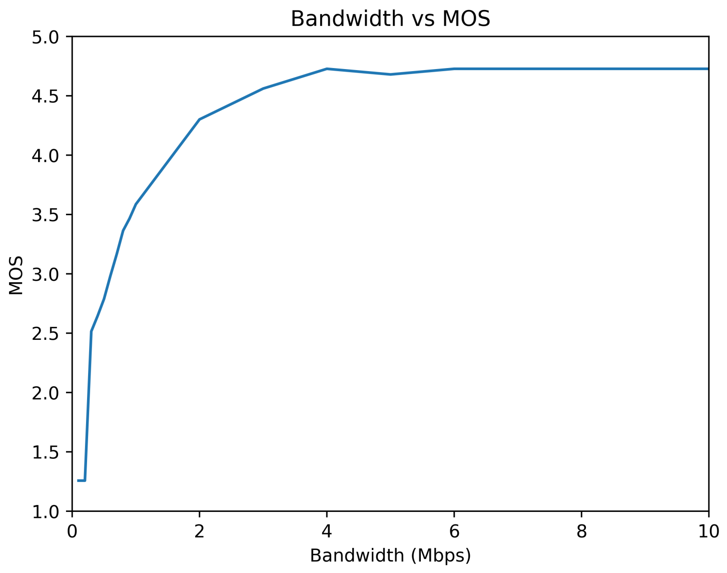


Figure 5.1: MOS affected by Bandwidth

Bandwidth deeply affects the MOS. 4 Mbps of availability is enough to reach peak MOS as figure 5.1 exhibits and appears to be the boundary between either a maximum MOS or a decaying one. With bandwidth higher than 4 Mbps there is unused bandwidth which can potentially be used to prefetch without any negative cost.

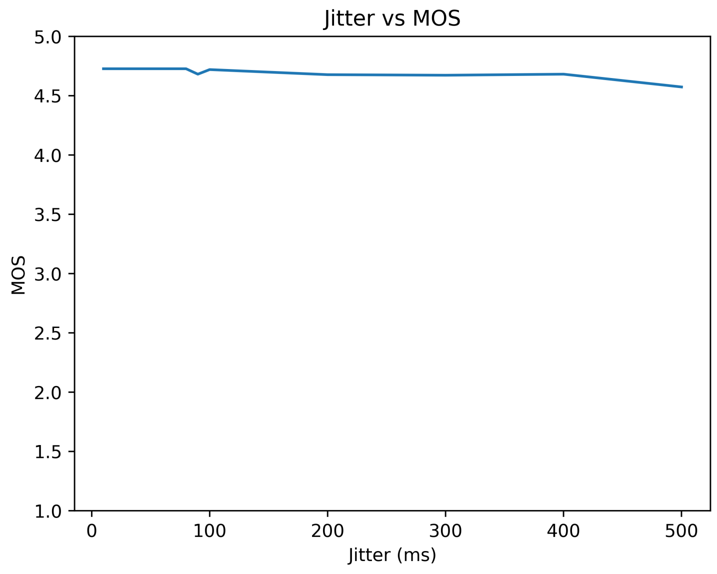


Figure 5.2: MOS affected by Jitter

The curve of MOS vs Jitter is very stable, and besides a small reduction in MOS along the entire range of limitations, even with 500ms, it presents a MOS higher than 4.5. Jitter is the metric that less affects the MOS variation, as shown in figure 5.2.

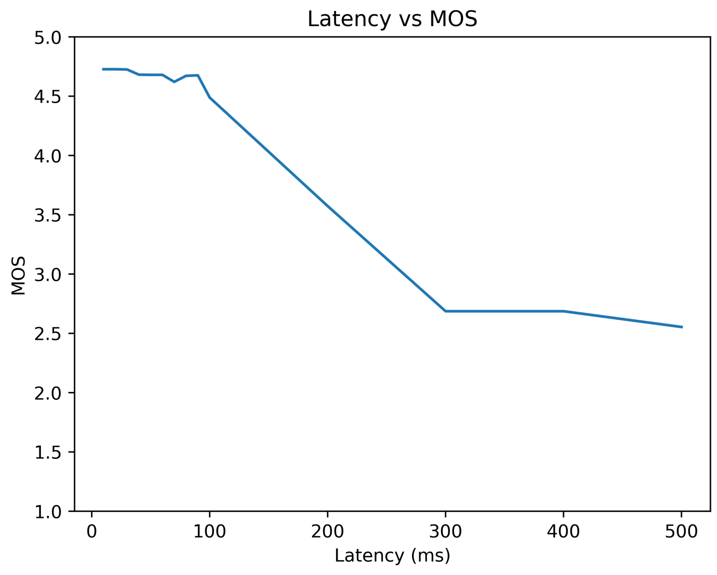


Figure 5.3: MOS affected by Latency

Presented in figure 5.3, the latency shows a stable behavior until it reaches the 100ms perimeter. From that point ahead, there is a constant decrease in MOS until the latency reaches the 300ms limitation, which from then on has a more stable reaction. This behavior is due to the probe reaching the lowest bitrate available, and being unable to lower the quality of the requested video.

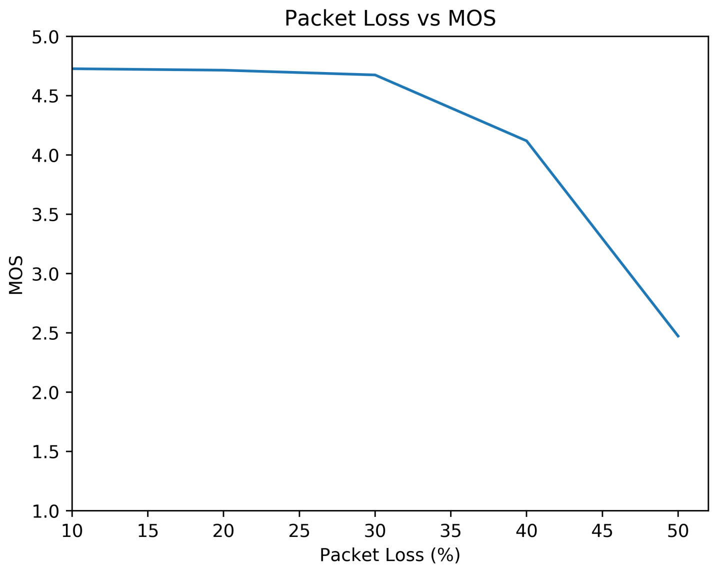


Figure 5.4: MOS affected by Packet Loss

Packet loss significantly affects MOS as displayed in figure 5.4. Similar to the bandwidth, there is a margin at 30% packet loss, where the MOS behavior is decreased.

Based on these results, the next sections will present how the proposed methods deal with this behavior, taking into account that each metric individually affects the experience, all of them having different levels of consequence, and that together, they may have an even more disruptive behavior against the MOS value. Moreover, a case where real-world values are measured and used is examined.

5.2 AHP-based Results

The testbed needed to support such environment is based on the typical client-server structure. Given the need to test different networking conditions, more than 1000 different testing combinations were defined, where not only said conditions vary as well as multiple concurrent clients simulated over the same link to provide better average values. As such, some of those independent variations are:

- Bandwidth (Mbps): 1, 2, 5, 10, 20, 50
- Jitter (ms): 0, 200, 400, 600
- Latency (ms): 0, 200, 400, 600, 800, 1000
- Packet loss (%): 0, 10, 20, 50

- Concurrent clients: 1, 2, 5, 10, 20

The testbed supporting the virtual machines follows an independent network topology according to figure 5.5:

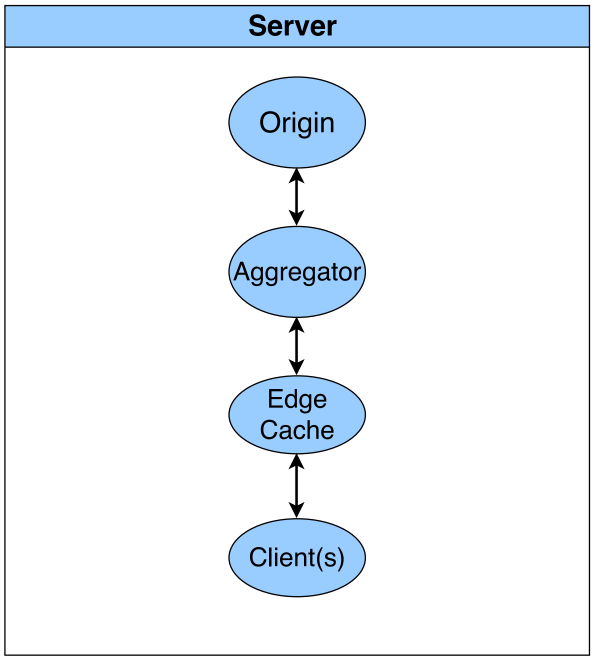


Figure 5.5: AHP Testbed architecture

Thus, it is possible to test all variations in the same VM by launching sequential tests as soon as the previous one has finished, following the lifecycle presented previously in Figure 4.2. The result logs may be collected at the end of the whole set of tests from each of the machines independently. For performance comparison purposes, not only the baseline was used, but also fixed prefetch amounts:

- Prefetch (chunks): 0, 1, 2, 5, 10

This way it is observable how the AHP Static approach behaves compared to the baseline as well as fixed prefetch solutions.

AHP is a static approach to solve a problem in defining the best weights for different characteristics regarding the QoS of a network. By introducing prefetching techniques which consist of downloading 2 seconds of the playing video, we aim to improve the Quality of Experience (QoE) of a user, so that he can have a smoother experience while watching his video. A smoother experience includes a higher resolution video compared to the baseline, or even a video which does not suffer from stuttering or buffering events. On a more practical approach, by using AHP and in defining the weights we ended up with the Equation 4.9.

Taking this into account and with the network variations, all the different unique tests were ran and ended up presenting some interesting results. The baseline is a test ran without any prefetch (blue line) where a normal client was emulated and received content along the 5 minutes of the test. The AHP (orange line) is a test where the mentioned equation is used to measure the current characteristics of the network and give a number called n_{chunks}. This number is the amount of prefetch that is used during the test with AHP. The same 5 minutes video duration was used for all the tests.

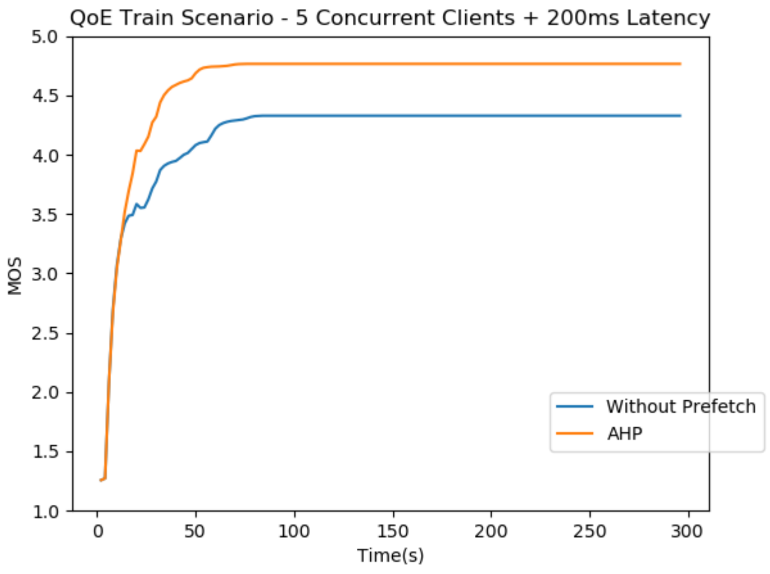


Figure 5.6: Baseline vs AHP with latency

It is possible to observe in figure 5.6 that, approximately, after the 20th second, the MOS obtained by the test running the AHP method is clearly higher: most of the time MOS is 0.5 higher when AHP is used. At the beginning of the test, both are very similar and what this shows is that the player (video) is still adapting to the conditions and trying to get higher qualities until it reaches a peak. Then, from that moment on, the client cannot obtain the higher resolutions in viable time. This fact is dealt by AHP method by requesting chunks in advance, and it appears to deal with latency in advance as well, thus allowing for higher resolutions to be grabbed and played without any problem. As such, AHP is a good method to use when high latency networks are in place.

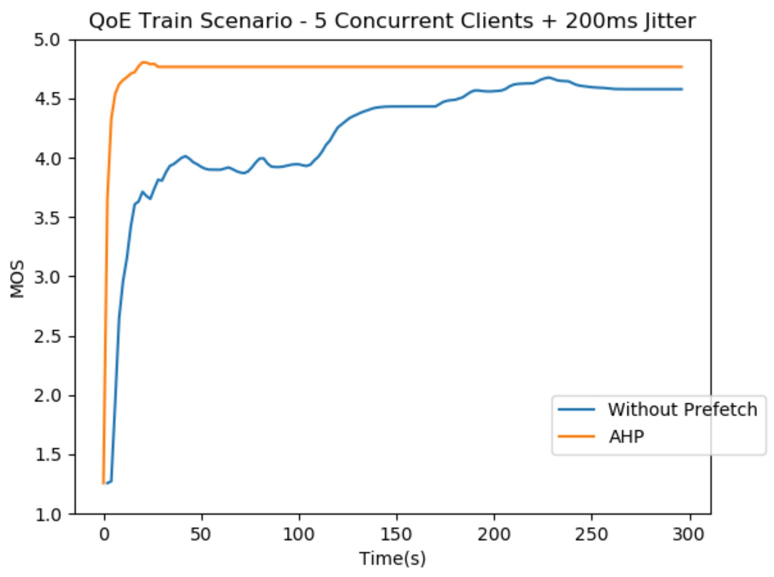


Figure 5.7: Baseline vs AHP with jitter

Jitter variations, as shown in figure 5.7, are as important as any other characteristic because they introduce large inconsistencies in the stability of the network connection. Comparing to the latency graph, this one presents itself with a baseline which is very unstable. In any case, even with such inconsistencies, the AHP method gets better results at every moment. Again, we can observe that it is a good method to use with networks suffering from jitter or latency.

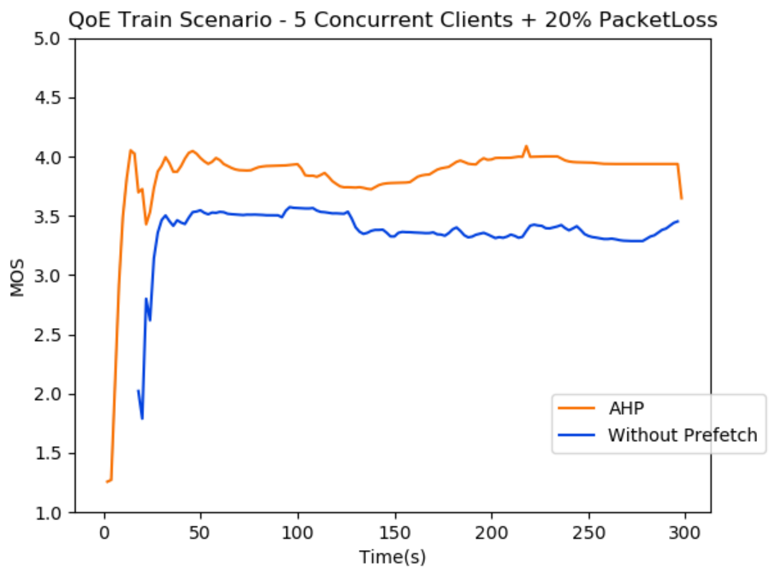


Figure 5.8: Baseline vs AHP with packet loss

Packet loss has similar results, where AHP shows better results compared to the baseline as visible in figure 5.8. Loosing 20% of the packets appears to be a very problematic situation, even though the player was able to play videos at lower resolutions in both cases (without prefetch and AHP). Definitely, asking for chunks in advance is good, assuming only packet loss is happening. By giving enough time for the TCP session to keep requesting the lost packets, it is observed in the graph that it is possible to get a consistent improvement.

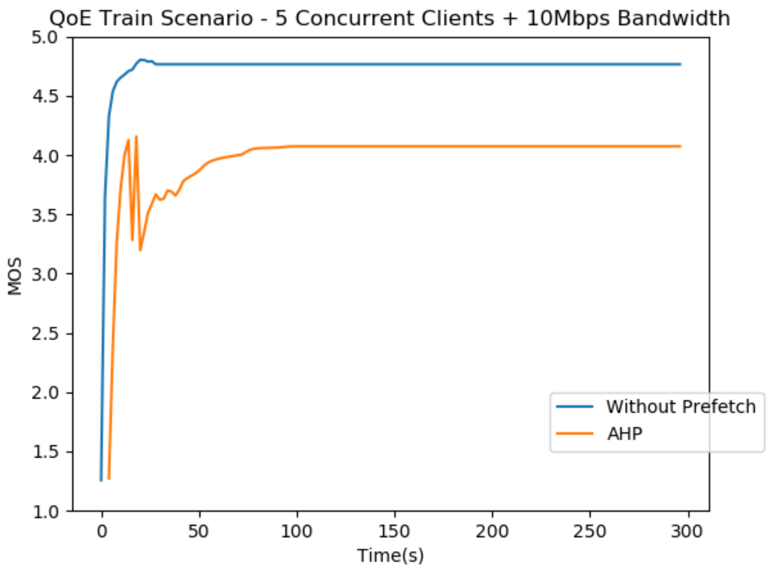


Figure 5.9: Baseline vs AHP with bandwidth limit

Bandwidth variation, though, behaves in a complete different way. Figure 5.9 presents a case where MOS was not improved. This was the first case where improvements were not seen and, in fact, it was much worse than the baseline. This is justified because the prefetching mechanism, by asking for future chunks, is in fact using more bandwidth that, in this case, is not available. As such, it is possible to conclude that, if the network does not have available bandwidth to support both the normal requests and the prefetch side by side, it is not good to be using AHP as it will deal with request concurrency, thus bringing down the quality and, as a result, the MOS also lowers.

To further study how AHP performs, we will compare it not only with the baseline, but also with static prefetch solutions with different degrees of prefetching.

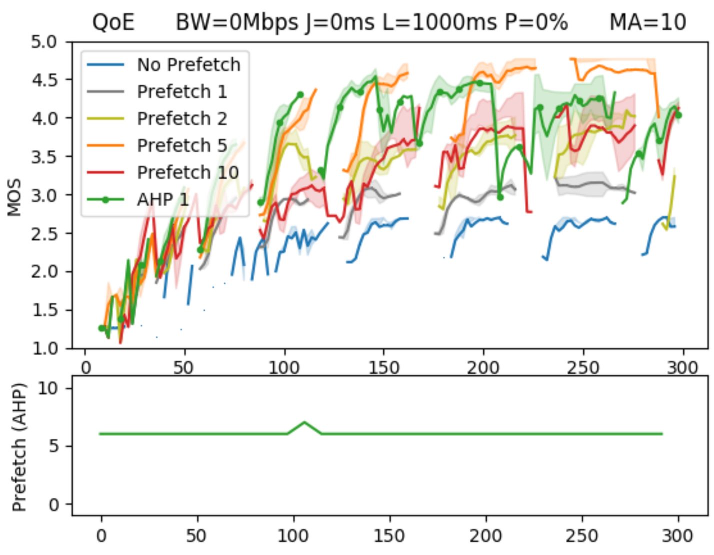


Figure 5.10: AHP vs fixed prefetch with latency

Again, AHP is definitely showing improvements, as shown in figure 5.10, over the case without prefetch, and we can see now that it is high up in the MOS result, between the fixed prefetch of 5 and 10 chunks. What is most interesting about this graph is that it shows that the AHP has chosen almost the same prefetch along the duration of the test (6 prefetched chunks). To observe this, we will test with high packet loss, since it is the characteristic with the biggest weight (64%) in our equation.

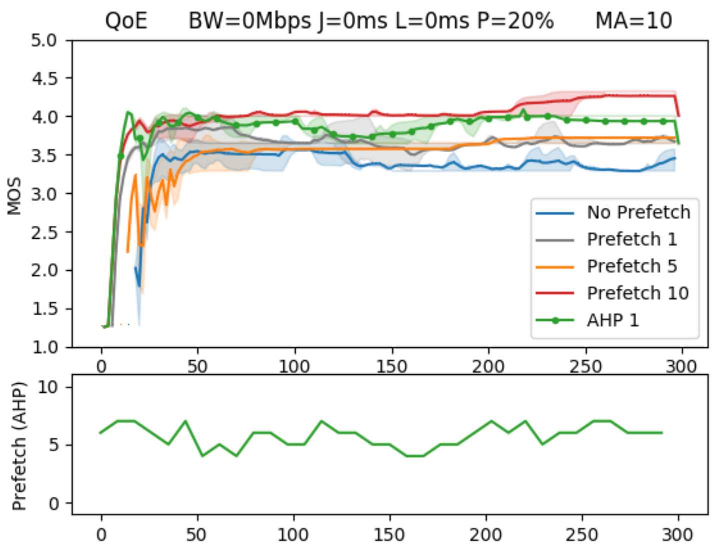


Figure 5.11: AHP vs fixed prefetch with packet loss

Figure 5.11 presents visible higher variations in the calculated prefetch by the AHP. There were improvements from the baseline, and given that some fixed prefetching is still higher than AHP, it gives the idea that there is still room from improvement. This is also possible to be seen with limited bandwidth.

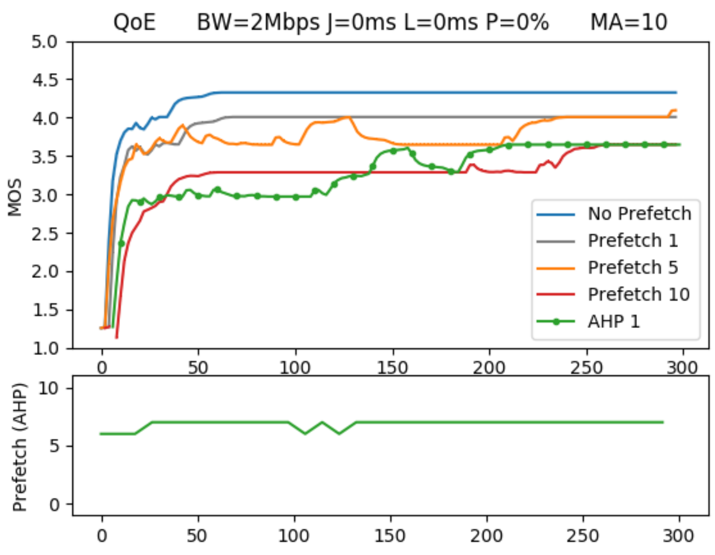


Figure 5.12: AHP vs fixed prefetch with bandwidth limit

It is visible in figure 5.12 that prefetching is not good for low bandwidth situations, and the AHP decision is nearly static along the test (6-7 prefetched chunks). This means that most weight was given to packet loss and the other metrics' results, and the small weight given to bandwidth is not enough to deal with these cases.

5.3 Self-learning based Results

The self-learning approach differs from the AHP because it has the goal of weighting all the network characteristics not separately but altogether, while weighting the correlation between the characteristics using the tested data results. Another goal of this self-learning is to improve the MOS similarly to AHP and to generate a model which takes into consideration the MOS decrease while prefetching chunks if no bandwidth is available.

The neural network was modeled with Keras [58], a high-level API, that can run on top of TensorFlow [59]. It is developed in Python which was chosen from its easiness of implementation, user-friendliness, and modularity.

The input layer of this approach consists of a sequential model with a linear stack of layers. The input shape is defined as 5 dimensions (one dimension for each metric and another for the prefetch amount). This model is defined by an implementation of densely-connected neural network layers. After several testing, the best results were obtained with two hidden layers and a single output which is the predicted MOS. Each one of the hidden layers is a dense layer which, from a single output, connects to 100 outputs as depicted in the following figure 5.13.

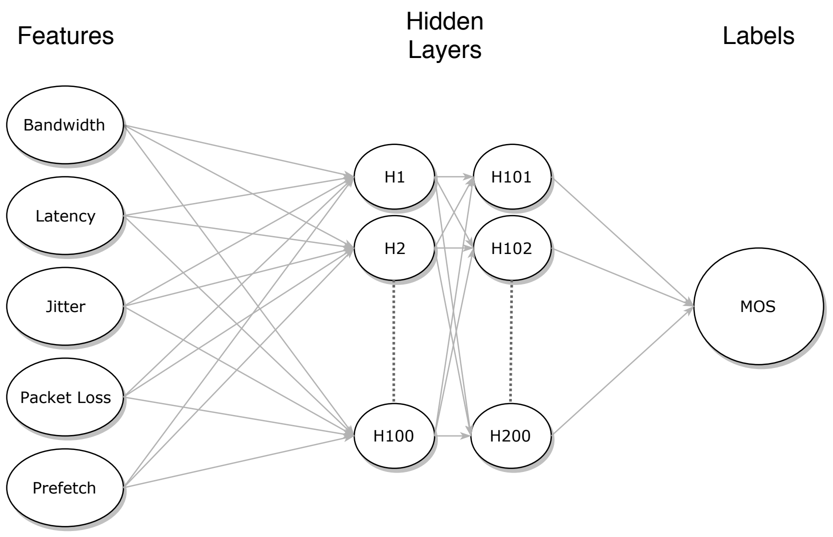


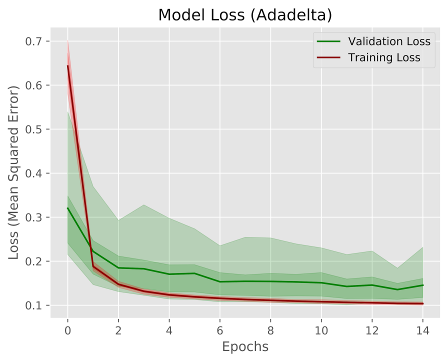
Figure 5.13: Deployed deep neural network

The generated models will first randomly try to mimic the behavior of those tests by reaching the result, and then increasingly try to minimize the mean-squared error between the predicted and actual results. Some of the best results were achieved with 10 epochs while using training batches of 100 values, where 20% of those are split and used for validation purposes.

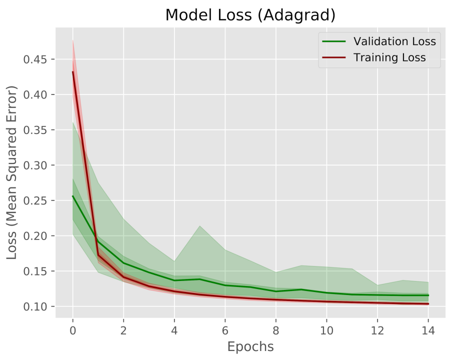
Optimization algorithms try to minimize an objective function, which is just a mathematical function. From a set of values, the neural network will change the optimizer weights and bias in order to minimize loss through a mathematical process based on the Neural Network model's training iterations. From the measured outcomes, some optimizers ended up with high loss disparity (Figure 5.14). This outcome means that the model is not able to precisely predict the actual MOS value when the metric values and prefetch are feed as input. As such, the algorithms, namely "Adadelta", "Adagrad", "Nadam", and "RMSProp", are discarded from the intention of predicting the quality of experience, since all of them have high validation error rates. "SGD" has low variation error, although compared to "Adam" and "Adamax" it needs more training epochs to reach a similar small loss. As such, it is also dismissed as an alternative.

Between the two best-performing optimizers, "Adam" is the one which offers best overall results even against "Adamax". While "Adamax" takes 8 epochs to reach 0.1 loss, "Adam" can reach the same loss with nearly half of that. Moreover, it even goes further as being constantly lower than 0.1 after the 5th epoch. As such, it is faster to create a prediction model and is more accurate.

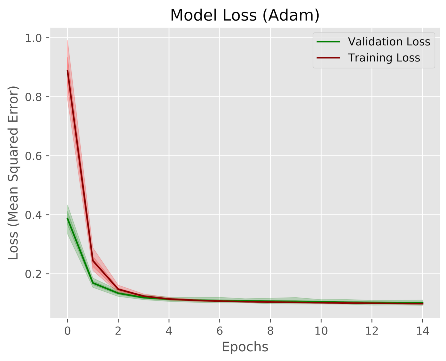
Some of the best results achieved a mean-squared error of 0.09. For this, the complete study of optimizers is presented ahead, showing how the different algorithms behave with the training and testing data, allowing to choose the best one for the specific purpose in this Thesis. The study averages 100 models training for each optimizer with 15 epochs each.



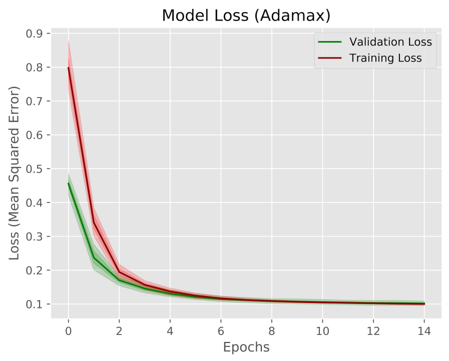
(a) Adaptive Delta



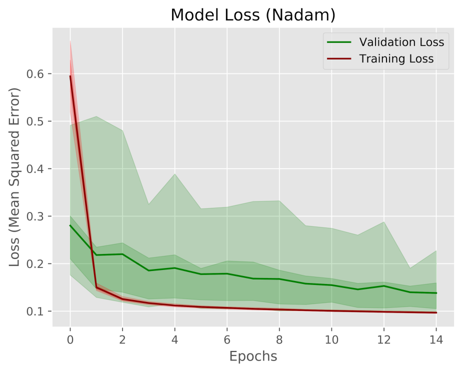
(b) Adaptive Gradient



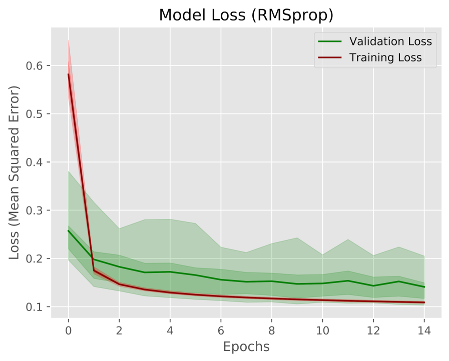
(c) Adaptive Momentum Estimation



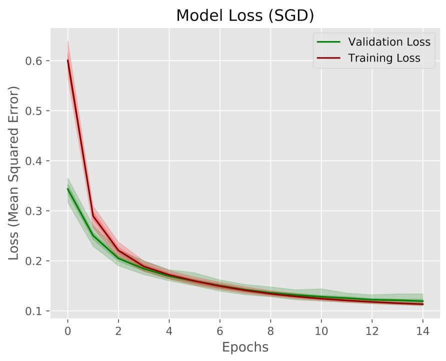
(d) Adaptive Momentum Estimation Infinity norm



(e) Nesterov-accelerated Adaptive Momentum Estimation



(f) Root Mean Square Propagation



(g) Stochastic Gradient Descent

Figure 5.14: Model Loss with several Optimizer algorithms

After the learning process of the neural network, the predictions on the whole range of values are requested on single network characteristics with different amounts of prefetch. This process allows picturing how the neural network predicts the variation along the characteristics against the whole range of prefetch.

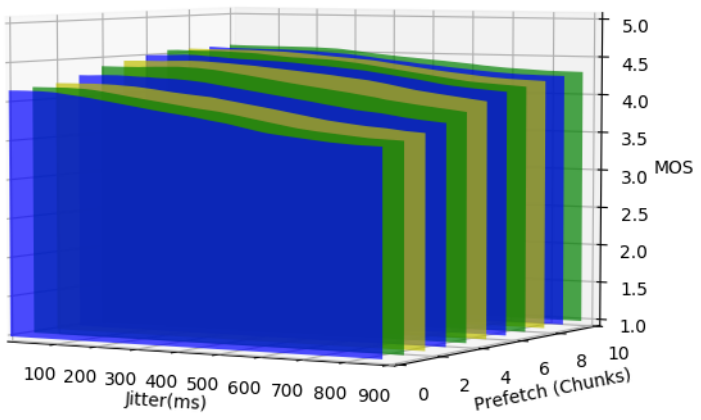


Figure 5.15: Predicted MOS with jitter variation

Figure 5.15 shows the variation of MOS along the whole defined jitter scale, with 0 chunks of prefetch to a maximum of 10 chunks.

This result is very similar to AHP, where we can observe the MOS increasing with more prefetched chunks. The results show in some cases an increase in MOS of nearly an entire unit. Taking a closer look at the MOS with 900ms of jitter and 0 prefetched chunks, the value is close to 3.5, while in the case where the same 900ms of jitter are introduced but with 10 chunks of prefetch, the value is closer to 4.5. This shows that the model can predict an increase in MOS with an increase in the prefetch over networks suffering from jitter.

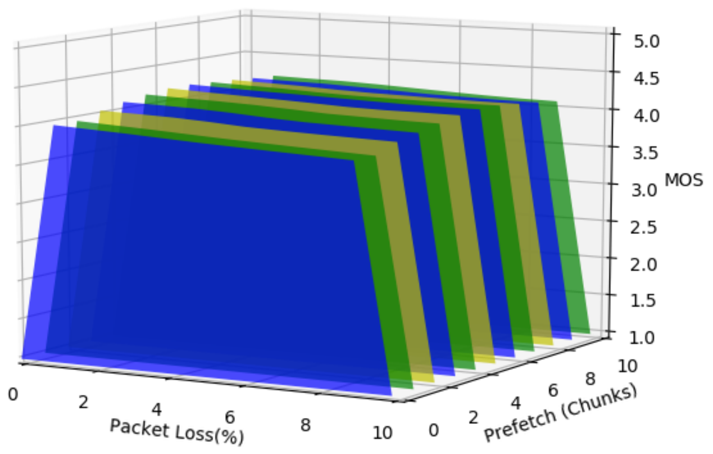


Figure 5.16: Predicted MOS with packet loss variation

Figure 5.16 shows the variation of MOS along the whole defined packet loss scale, with 0 chunks of prefetch to a maximum of 10 chunks. With packet loss, there is still an increase in MOS with the prefetch increase; however, less noticeable than jitter. With respect to packet loss, it still appears to suffer around a 0.5 MOS increase along the 10% packet loss line.

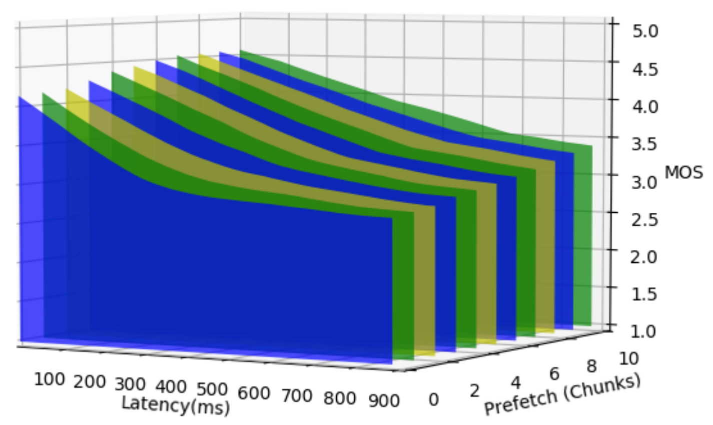
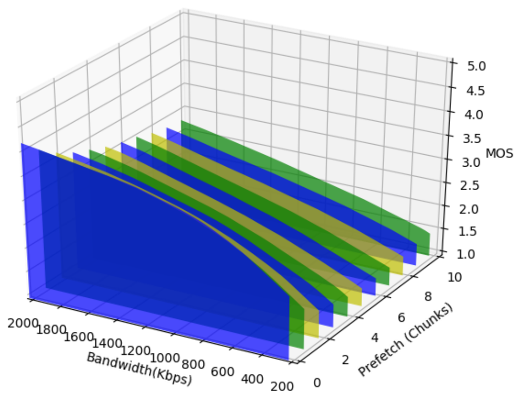
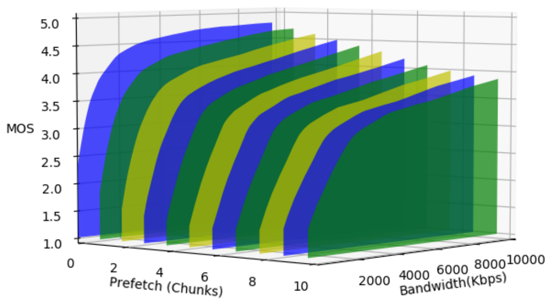


Figure 5.17: Predicted MOS with latency variation

Figure 5.17 shows the variation of MOS along the whole defined latency scale, with 0 chunks of prefetch to a maximum of 10 chunks. The model shows significant improvements in every range of the latency: an increase of nearly 0.5 MOS along the 100ms latency line, nearly 1 MOS along the 900ms latency line, and even 1.5 for larger latency values.



(a) From 200 Kbps to 2 Mbps



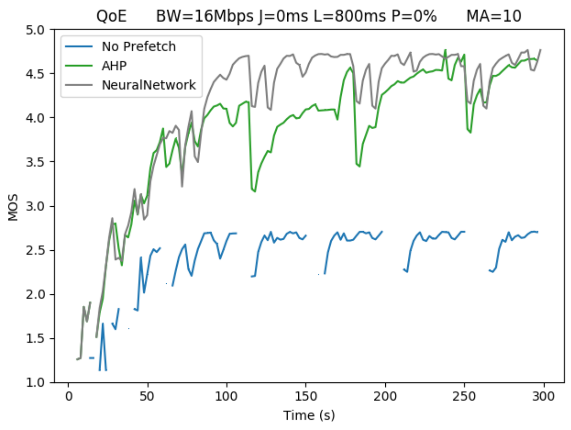
(b) From 200 Kbps to 10 Mbps

Figure 5.18: Predicted MOS with bandwidth variation

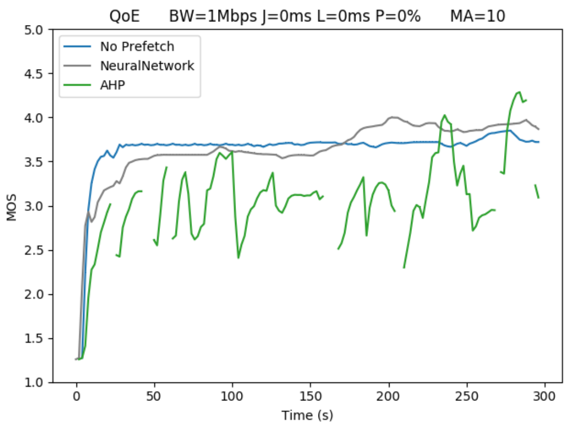
Figure 5.18 shows the variation of MOS along the whole defined bandwidth scale, with 0 chunks of prefetch to a maximum of 10 chunks.

The model was definitely capable of predicting a worse MOS with higher prefetch, which is expected to happen in low bandwidth networks. This means that the model is accurate with the real results and confirms the small error attained in the prediction. As such, it might prove crucial in choosing the correct prefetch to use depending on the current network characteristics.

To conclude, the model should be compared with the baseline as well as the AHP results, in order to confirm if the results improve in most cases. As such, tests were done as shown in figure 5.19, which show two examples, one where there is high bandwidth but high latency, and the other where there is low bandwidth.



(a) Latency Limitations (800 ms)



(b) Bandwidth Limitations (1 Mbps)

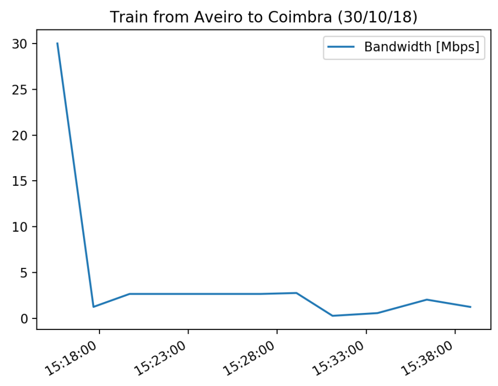
Figure 5.19: Self-Learning vs Baseline vs AHP

In the example shown above, the learning-based approach was able to attain similar or even better MOS than the AHP model with high bandwidth and high latency, and it goes as further as also keeping a similar MOS as the case without prefetch in the low bandwidth case.

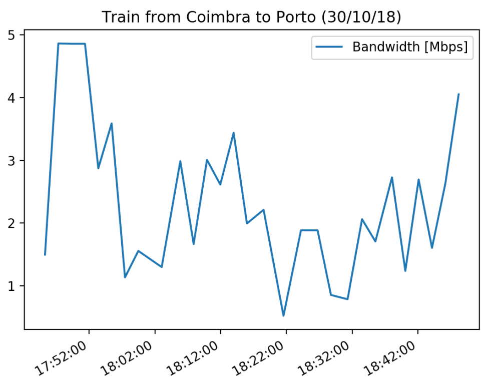
These results are proof that the learning-based approach did learn about the indirect result of network metrics in the behavior of MOS, and was able to positively adapt the prefetch to maximize the experience given the poor network quality exhibited.

5.4 Real-world Train Scenario

The real-world train scenario present some interesting results. At first, it is essential to analyze how the independent metrics behave along both train travels. From both travels, and according to the specified measurement methods, it is essential to retain that 38 minutes have been used to measure latency, jitter and packet loss, and 38 minutes for bandwidth. Every metric has been averaged to emulate with the testbed which appears to be a good indicator, except for latency which is explained further ahead.



(a) First train travel



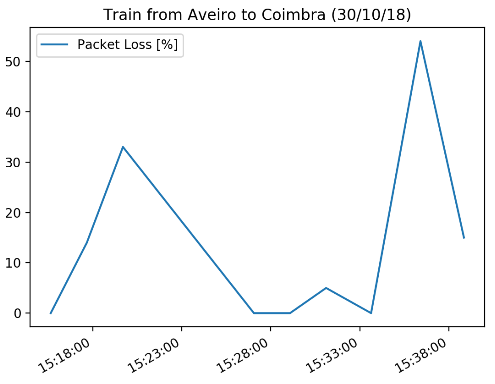
(b) Second train travel

Figure 5.20: Measured Average Bandwidth in Train

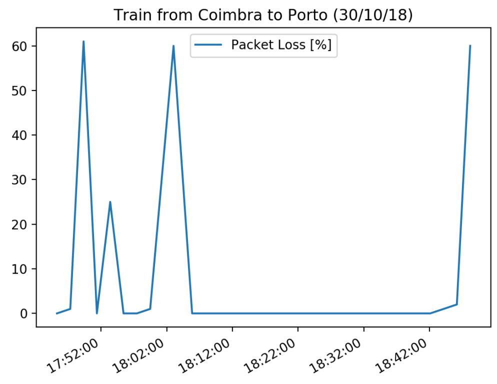
According to Figure 5.20, the first travel has a smoother bandwidth although there is a higher peak at the beginning reaching 30 Mbps. This peak happened because the train had not left the Aveiro's station and was still. So, there is a high chance the 4G Hotspot had a direct line of sight with a base station at the time of measurement, thus being nearly ideal conditions for the bandwidth test. Besides, it is possible to observe that, whenever the train is moving, which is for all the rest of the travel, low bandwidth is achieved. Also, higher variations are observable, which is expected from a typical 4G environment.

To bear in mind that, with a refresh rate of 10Hz, each minute has 600 measurements. Since the three metrics were measured for 38 minutes, 22800 replies were expected. From that, and since only 20780 packets received a reply, the rest of the packets are assumed as lost, meaning that around 10% of the whole trip suffered packet loss. Also, 5000ms was the defined maximum timeout for a reply; beyond that threshold it is considered lost.

It is noticeable in figure 5.21 that, contrary to the previous environments, with 4G, packet loss appears to have either very high values during a short amount of time or very low. A



(a) First train travel

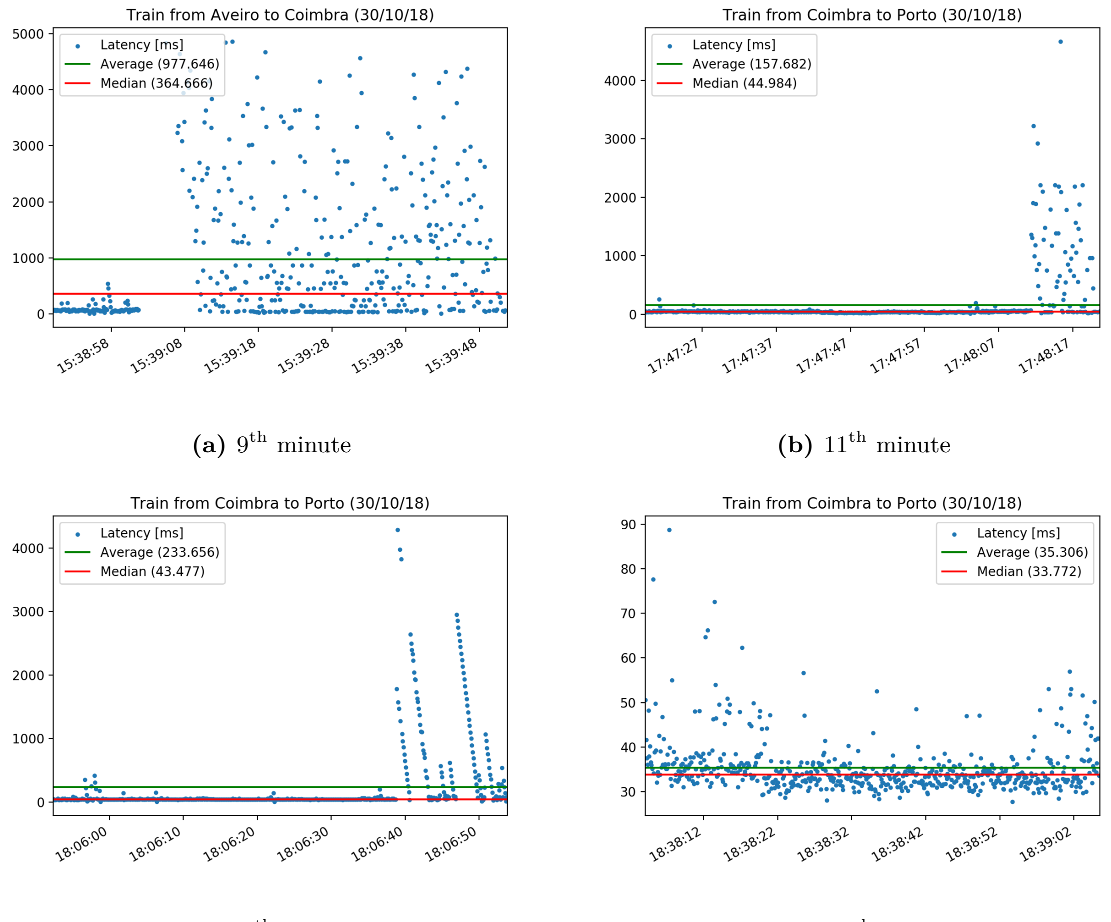


(b) Second train travel

Figure 5.21: Measured Average Packet Loss in Train

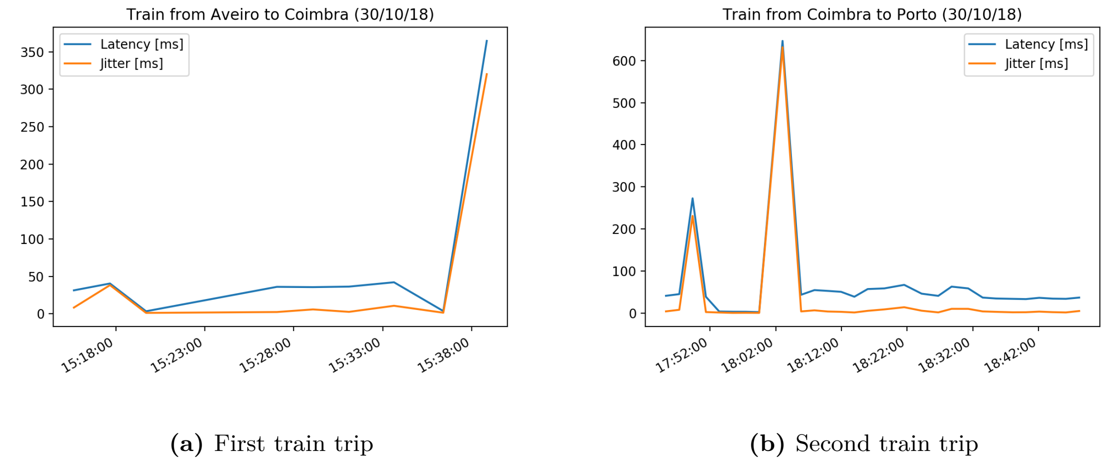
factor that might be of high influence to this behavior is the cellular handovers caused by the high speed of the train.

Because latency is not evenly spread across the whole range, and that there are some exceptionally high values measured, instead of mean, the median is used which allows for a smaller overall influence caused by the obtained values. As seen in figure 5.22, which represents a minute of data collection in each graph, the median shows a more realistic statistical approach to the raw collected data. It is also noticeable in the 19th minute, that there is a queueing effect, that presents itself as a linearly decreasing effect. This is due to a lack of connectivity with the base station, and the packets are kept in a queue until either connection is re-established or the 5000ms timeout window expired.



(a) 9th minute **(b)** 11th minute **(c)** 19th minute **(d)** 33rd minute

Figure 5.22: Measured Latency and Jitter in Train (Raw distribution)



(a) First train trip **(b)** Second train trip

Figure 5.23: Measured Median Latency and Jitter in Train

Figure 5.23 shows the median of latency along the train trip and the calculated jitter from the original values. This metric is highly variable and suffers significant influence from external factors: objects around the city, the train speed, buildings and the distance of the base station, are all of high impact for the stability of this metric.

Finally, after all metrics suffer the mentioned statistical treatments, they are emulated in the testbed to test the proposed methods in the scenario. Since for the purpose there is no difference if the train is traveling in one direction or the other, or even the time of the test, in the emulated scenario, both train trips are assumed to be sequential and without interruption. Thus, the emulation considers a train trip lasting 76 minutes.

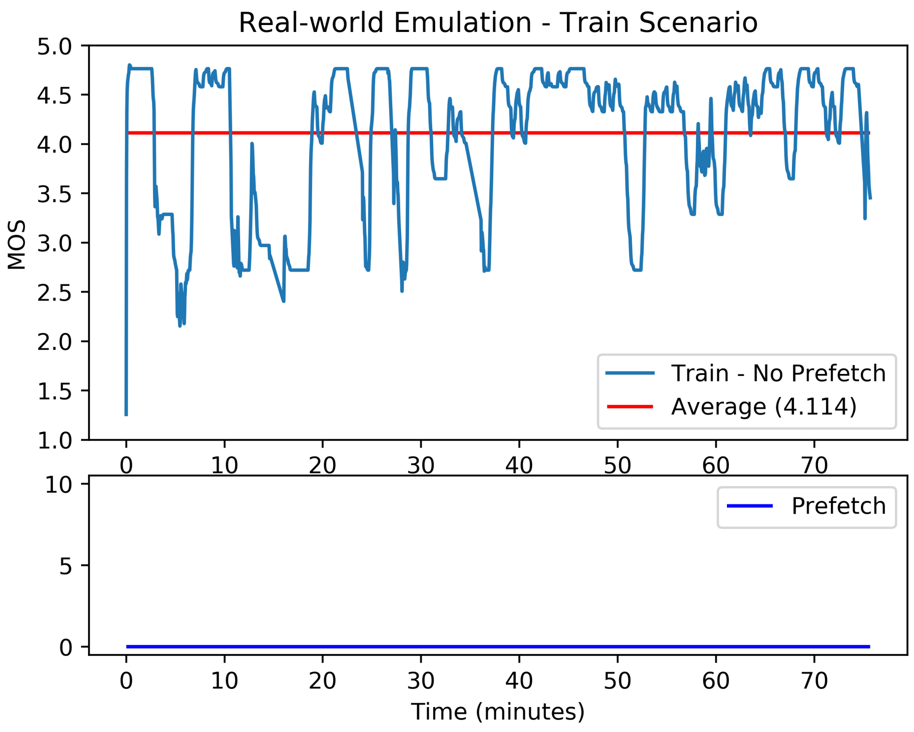
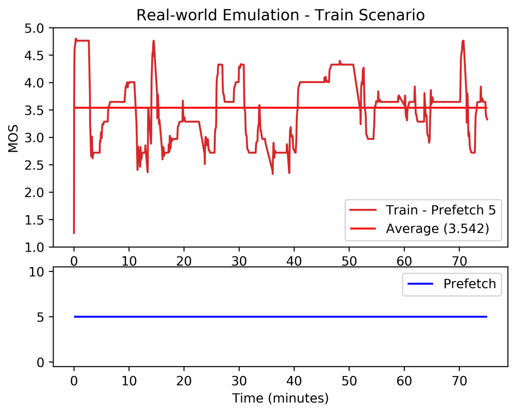
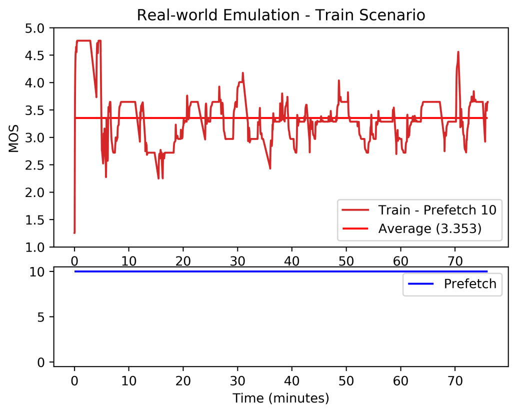


Figure 5.24: Train scenario - Baseline

The baseline is presented in figure 5.24, which represents simple train travel streaming a video without any prefetching. It is possible to observe that there are many periods where the MOS is as low as 3.0 and even 2.5, with an average of 4.114.



(a) 5 Chunks



(b) 10 Chunks

Figure 5.25: Train scenario - Fixed prefetch

Depicted in figure 5.25 are two fixed prefetch emulations, where five and ten chunks, respectively, were used. In this scenario, it was possible to observe that the higher the prefetch, the lower the average MOS. Higher MOS can be seen in short periods of time, although overall it is lower with 10 chunks.

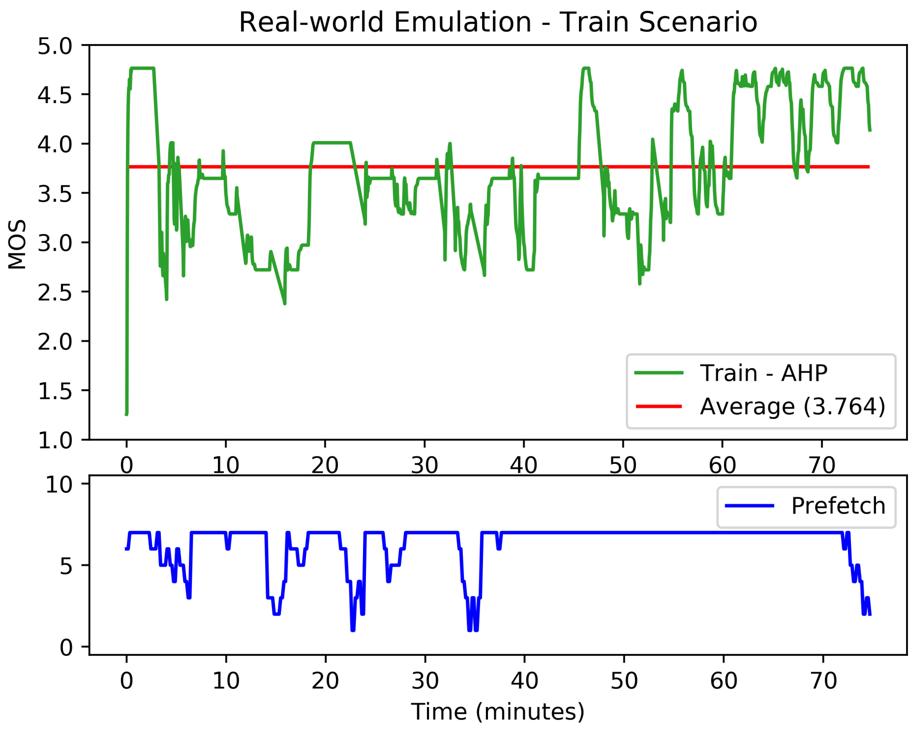


Figure 5.26: Train scenario - AHP approach

Figure 5.26 also confirms the previous findings that, even though prefetch adapts according to the measurement of every metric, the weights given through the AHP method are not enough to sustain higher MOS along the whole test without enough bandwidth. Only potential local improvements are made which are not counterbalanced by the lower MOS found in the rest of the test.

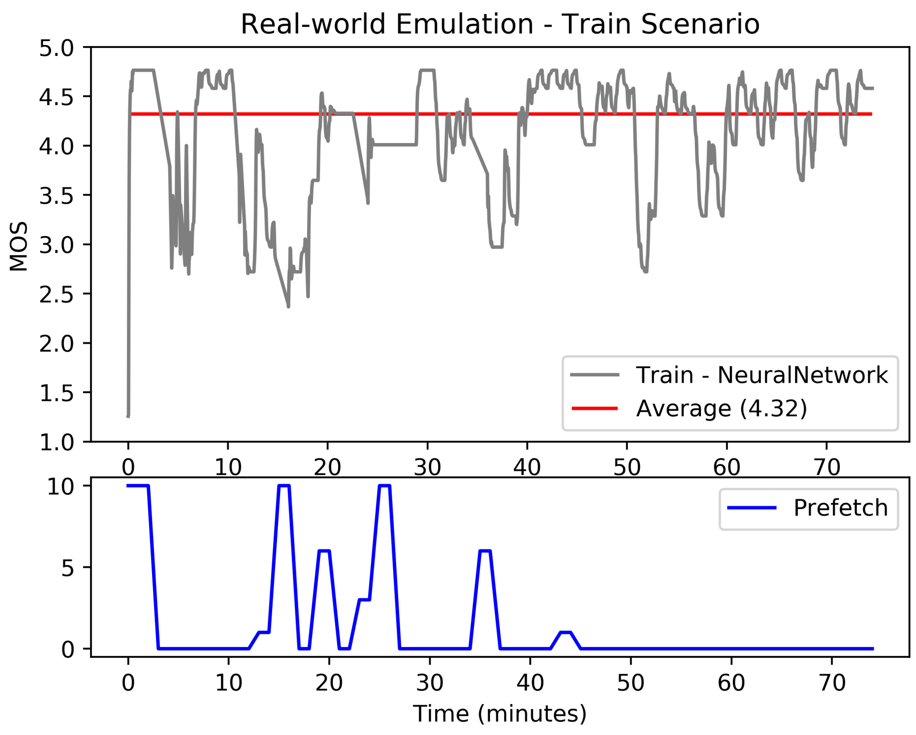
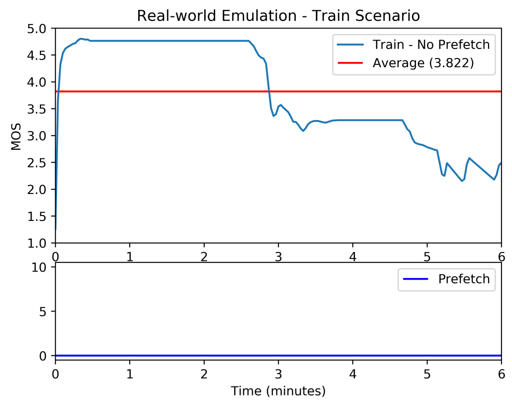
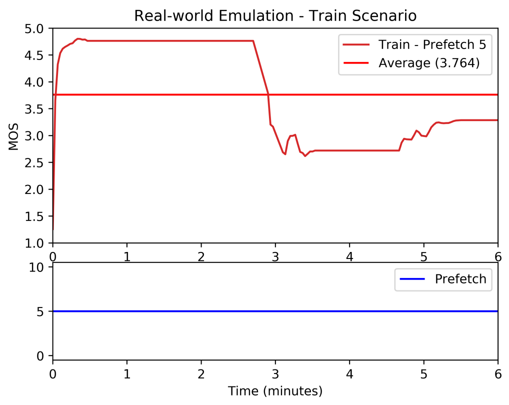


Figure 5.27: Train scenario - Self-learning approach

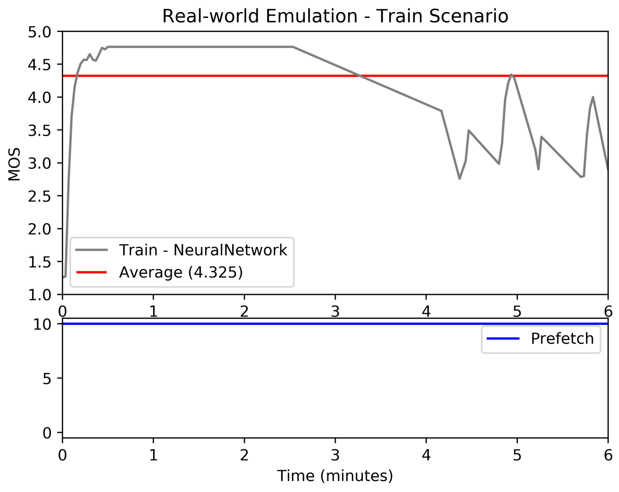
At last, figure 5.27 shows that, with self-learning it is possible for the model to recognize the lack of available bandwidth, thus it only prefetches in very specific moments, that are close to areas where most bandwidth could be used according to 5.20. With this, 4.135 MOS was achieved with this model which is a minimal improvement over the baseline.



(a) Baseline - No prefetch



(b) 5 fixed chunks



(c) Self-learning

Figure 5.28: First minutes with high available bandwidth

The first minutes of the test have show an area where clear improvements over the baseline are visible as shown in figure 5.28. The self-learning approach is able to sustain a high MOS during longer, and where the baseline has fallen, it is still able to reach some higher peaks. The shows, once again, that the lack of bandwidth precludes prefetching: in this train scenario, the available bandwidth was only enough to help improve the MOS in specific areas of the test.

5.5 Conclusions

This chapter presented the results obtained from the implemented architecture from Chapter 4 with the proposed models from Chapter 3.

First, the impact of QoS in the MOS is evaluated in a single client. Jitter ended up being the metric which less affected the MOS. Even with an increase of 500 ms, there was not a significant drop in QoE. Latency has a significant deterioration after 100ms and after 300ms the decay is smaller, even though, there is a significant drop of around 2.0 MOS between those values. Bandwidth and packet loss are the ones that have a bigger effect. Bandwidth has an exponential-like drop in MOS when there are less than 4 Mbps, and packet loss also presents a similar loss around 30%.

The AHP has provided worse results in bandwidth but better in every other metric. The model gives a small weight to the measured bandwidth and, even without enough bandwidth available, it might still try to prefetch a large number of chunks.

The self-learning approach ended up presenting a low Mean Squared Error with training, which in return means that the predictions are accurate. The model output per metric appears to match the findings in individual QoS impact, and still provide an accurate MOS prediction with prefetch. The results of its prediction show similar or better average MOS compared to the AHP, without the disadvantage of worsening the QoE with low bandwidth.

In the real-world train scenario, the results once again prove the findings in the different approaches and AHP worsens the average QoE due to the lack of bandwidth, while the self-learning approach can keep it sustained. More research would be useful to try and improve this use case with the proposed models.

CHAPTER 6

Conclusions and Future Work

The ever-increasing growing consumption of OTT services presents technological challenges, due to consumer expectation of being granted the highest service quality and experience without disruption.

The way the content is delivered is identified, and the technologies supporting such content and distribution are examined in the State of the art. As such, it is proposed and deployed an architecture based on CDN which reflects the common multimedia distribution methods in use nowadays, and two optimization models for evaluation and optimization of such delivery structure, aiming at the improvement of Quality of Experience (QoE).

A first approach is used where tests seek to measure how each QoS-related metric affect the final user's experience. Both proposed models take into account the current metrics conditions and, based on those, increase or decrease the amount of prefetch done in order to minimize the effects of atypical QoS values. These models are AHP and self-learning based which is based on a neural network. The AHP model is fed with subjective values according to a weight scale, to compare pairs of metrics, and outputs a weighted function. This function can input exact measured QoS metrics, and output a normalized value maximized by a scalar value regarding the maximum prefetch duration to be done. The self-learning module learns from previous tests how to weight the QoS-related metrics. The learning process is improved the more tests are done, and in this thesis, a dataset of more than 14.000 hours of video under different QoS combinations was evaluated. With the generated model, it is possible to predict the MOS of a video. With this prediction, at the time of prefetch decision, the highest prediction is chosen.

Results show improvements with AHP regarding severe latency, jitter and packet loss. Even though AHP does not give the best result, it is a fine approach when there is an assurance of bandwidth availability. The same case is not verified with lack of bandwidth though, where a mere single chunk in advance results in worse results for the final client.

The results from the self-learning approach show that the model can accomplish good results with realistic cases. It is lightweight since, after the model is trained, hundreds of predictions can be generated in few milliseconds. The neural network was able to adapt itself to cases where bandwidth and prefetch negatively influence the results, as well as cases where prefetch by itself can improve the users perceived quality.

The models did not achieve large positive results in the real-world train emulation due to the lack of measured available bandwidth nearly everywhere in the train path. It is important to notice that these results are as expected due to the measured metric behaviour.

Finally, the proposed architecture may show potential improvements in specific scenarios, thus providing content delivery networks with dynamic optimization methods. Useful insights were achieved with this Thesis and may be of importance for future academic research.

Other possible improvements and future work, which would require further studies and goes in consonance with the use cases, would include:

- The overnight download of a high-bitrate content or even parts of the video that are not sequential;
- Use the available bandwidth as a scalar value to maximize the prefetch amount rather than fixed chunks in advance;
- Use the caches hit/miss ratio as an input to the neural network as a way to describe if the decisions that the model is taking are good or bad;
- Train the neural network to predict lack of efficiency in the link, and prefetch even further chunks.
- Use the requested chunk quality as an input to the neural network, to provide means to predict and deliver higher quality chunks while at the same time decreasing the cache miss ratio;
- Deploy the learning-based approach in a real train scenario, and assess its performance for different types of travels, videos, and prediction of the areas with low network performance.

References

[1] J. Silva, A. Dias, J. Nogueira, L. Guardalben, and S. Sargento, "Content-aware prefetching in Over-The-Top wireless networks", in *Proceedings - IEEE Symposium on Computers and Communications*, IEEE, Jul. 2017, pp. 515–522, ISBN: 9781538616291. DOI: 10.1109/ISCC.2017.8024580. [Online]. Available: http://ieeexplore.ieee.org/document/8024580/.

[2] *Broadband technologies | Digital Single Market.* [Online]. Available: https://ec.europa.eu/digital-single-market/en/broadband-technologies.

[3] "Cisco Visual Networking Index: Forecast and Trends, 2017–2022 - Cisco", Tech. Rep., 2018. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.pdf.

[4] F. Coppinger and D. Piehler, "RF video overlay in an Ethernet passive optical network", in *2006 Optical Fiber Communication Conference and the National Fiber Optic Engineers Conference*, IEEE, 2006, 3 pp. ISBN: 1-55752-803-9. DOI: 10.1109/OFC.2006.215756. [Online]. Available: http://ieeexplore.ieee.org/document/1636787/.

[5] The Nielsen Company, "The Digital Consumer", Tech. Rep., 2014. [Online]. Available: https://www.nielsen.com/content/dam/corporate/us/en/reports-downloads/2014%20Reports/the-digital-consumer-report-feb-2014.pdf.

[6] J. Abreu, J. Nogueira, V. Becker, and B. Cardoso, "Survey of Catch-up TV and other time-shift services: a comprehensive analysis and taxonomy of linear and nonlinear television", *Telecommunication Systems*, vol. 64, no. 1, pp. 57–74, Jan. 2017, ISSN: 1018-4864. DOI: 10.1007/s11235-016-0157-3. [Online]. Available: http://link.springer.com/10.1007/s11235-016-0157-3.

[7] A. B. Kumar R, L. C. Reddy, and P. S. Hiremath, "RTSP Audio and Video Streaming for QoS in Wireless Mobile Devices", Tech. Rep. 1, 2008, p. 96. [Online]. Available: https://pdfs.semanticscholar.org/7562/a93084e2e028c109e76d2de06cd0be5848c8.pdf.

[8] H. Schulzrinne, A. Rao, and R. Lanphier, "Real Time Streaming Protocol (RTSP)", Tech. Rep., 1998. DOI: 10.17487/rfc2326. arXiv: arXiv:1011.1669v3. [Online]. Available: https://tools.ietf.org/html/rfc2326%20https://www.rfc-editor.org/info/rfc2326.

[9] A. Begen, T. Akgul, and M. Baugher, "Watching Video over the Web: Part 1: Streaming Protocols", *IEEE Internet Computing*, vol. 15, no. 2, pp. 54–63, Mar. 2011, ISSN: 1089-7801. DOI: 10.1109/MIC.2010.155. [Online]. Available: http://ieeexplore.ieee.org/document/5677508/.

[10] A. Zambelli, *The Birth of Smooth Streaming | Alex Zambelli's Streaming Media Blog.* [Online]. Available: http://alexzambelli.com/blog/2009/02/04/the-birth-of-smooth-streaming/.

[11] *How to optimize PPPoE connections? | SonicWall.* [Online]. Available: https://www.sonicwall.com/en-us/support/knowledge-base/170505851231244.

[12] J. Wijering, *What is Video Streaming?*, 2011. [Online]. Available: https://www.jwplayer.com/blog/what-is-video-streaming/%20http://www.longtailvideo.com/blog/19578/what-is-video-streaming.

[13] A. Zambelli, "IIS Smooth Streaming Technical Overview", *Microsoft Corporation*, no. March, 2009. [Online]. Available: https://www.bogotobogo.com/VideoStreaming/Files/iis8/IIS%7B%5C_%7DSmooth%7B%5C_%7DStreaming%7B%5C_%7DTechnical%7B%5C_%7DOverview.pdf.

[14] *HTTP Live Streaming (HLS) - Apple Developer.* [Online]. Available: `https://developer.apple.com/streaming/`.

[15] *Smooth Streaming : The Official Microsoft IIS Site.* [Online]. Available: `https://www.iis.net/downloads/microsoft/smooth-streaming`.

[16] *Live video streaming online | Adobe HTTP Dynamic Streaming.* [Online]. Available: `https://www.adobe.com/pt/products/hds-dynamic-streaming.html`.

[17] A. Vakali and G. Pallis, "Content delivery networks: Status and trends", *IEEE Internet Computing*, vol. 7, no. 6, pp. 68–74, Nov. 2003, ISSN: 1089-7801. DOI: `10.1109/MIC.2003.1250586`. [Online]. Available: `http://ieeexplore.ieee.org/document/1250586/`.

[18] G. Pallis and A. Vakali, "Insight and perspectives for content delivery networks", *Communications of the ACM*, vol. 49, no. 1, pp. 101–106, Jan. 2006, ISSN: 00010782. DOI: `10.1145/1107458.1107462`. [Online]. Available: `http://portal.acm.org/citation.cfm?doid=1107458.1107462`.

[19] "PROXY SUPPORT FOR STREAMING ON THE INTERNET", Tech. Rep., 2004. [Online]. Available: `http://www.cs.sfu.ca/%7B~%7Djcliu/Papers/comm04.pdf`.

[20] J. Zhang, "Replacement Strategy of Web Cache Based on Data Mining", in *2015 10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC)*, IEEE, Nov. 2015, pp. 821–823, ISBN: 978-1-4673-9473-4. DOI: `10.1109/3PGCIC.2015.75`. [Online]. Available: `http://ieeexplore.ieee.org/document/7424675/`.

[21] C. Cobârzan and L. Böszörményi, "Further Developments of a Dynamic Distributed Video Proxy-Cache System", Tech. Rep., 2006. [Online]. Available: `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.92.8648%7B%5C&%7Drep=rep1%7B%5C&%7Dtype=pdf`.

[22] J. P. B. F. Nogueira, "Over-the-top multimedia delivery: a catch-up TV case study", PhD thesis, Universidade de Aveiro, Jun. 2017. [Online]. Available: `http://hdl.handle.net/10773/22721`.

[23] *Apache Traffic Server.* [Online]. Available: `http://trafficserver.apache.org/`.

[24] *Application Request Routing : The Official Microsoft IIS Site.* [Online]. Available: `https://www.iis.net/downloads/microsoft/application-request-routing`.

[25] I. NGINX, *NGINX | High Performance Load Balancer, Web Server & Reverse Proxy.* [Online]. Available: `https://www.nginx.com/`.

[26] *Varnish Software | World-class Content Delivery Solutions.* [Online]. Available: `https://www.varnish-software.com/`.

[27] *squid : Optimising Web Delivery.* [Online]. Available: `http://www.squid-cache.org/`.

[28] I. K. Chaniotis, K.-I. D. Kyriakou, and N. D. Tselikas, "Is Node.js a viable option for building modern web applications? A performance evaluation study", *Computing*, vol. 97, no. 10, pp. 1023–1044, Oct. 2015, ISSN: 0010-485X. DOI: `10.1007/s00607-014-0394-9`. [Online]. Available: `http://link.springer.com/10.1007/s00607-014-0394-9`.

[29] Z. Zhao, L. Guardalben, M. Karimzadeh, J. Silva, T. Braun, and S. Sargento, "Mobility Prediction-Assisted Over-The-Top Edge Prefetching for Hierarchical VANETs", Tech. Rep. [Online]. Available: `http://www.swiss-sense-synergy.ch/wp-content/uploads/2018/02/JSAC%7B%5C_%7DMP.pdf`.

[30] G. Nencioni, N. Sastry, J. Chandaria, and J. Crowcroft, *Understanding and Decreasing the Network Footprint of Catch-up TV*, ISBN: 9781450320351. [Online]. Available: `http://www.bbc.co.uk/mediacentre/latestnews/2012/`.

[31] J. M. C. Silva, "Content distribution in OTT wireless networks", 2016. [Online]. Available: `https://ria.ua.pt/handle/10773/22733`.

[32] S. P. Vanderwiel and D. J. Lilja, "Data Prefetch Mechanisms", *ACM Computing Surveys*, vol. 32, no. 2, pp. 174–199, Jun. 2000, ISSN: 03600300. DOI: `10.1145/358923.358939`. [Online]. Available: `http://portal.acm.org/citation.cfm?doid=358923.358939`.

[33] B. Wu and A. Kshemkalyani, "Objective-optimal algorithms for long-term Web prefetching", *IEEE Transactions on Computers*, vol. 55, no. 1, pp. 2–17, Jan. 2006, ISSN: 0018-9340. DOI: 10.1109/TC.2006.12. [Online]. Available: http://ieeexplore.ieee.org/document/1545747/.

[34] P. Orosz, T. Skopkó, Z. Nagy, P. Varga, and L. Gyimóthi, "A case study on correlating video QoS and QoE", in *IEEE/IFIP NOMS 2014 - IEEE/IFIP Network Operations and Management Symposium: Management in a Software Defined World*, IEEE, May 2014, pp. 1–5, ISBN: 9781479909131. DOI: 10.1109/NOMS.2014.6838399. [Online]. Available: http://ieeexplore.ieee.org/document/6838399/.

[35] Z. Ning, Y. Liu, X. Wang, Y. Feng, and X. Kong, "A novel QoS-based QoE evaluation method for streaming video service", in *Proceedings - 2017 IEEE International Conference on Internet of Things, IEEE Green Computing and Communications, IEEE Cyber, Physical and Social Computing, IEEE Smart Data, iThings-GreenCom-CPSCom-SmartData 2017*, vol. 2018-Janua, IEEE, Jun. 2018, pp. 956–961, ISBN: 9781538630655. DOI: 10.1109/iThings-GreenCom-CPSCom-SmartData.2017.147. [Online]. Available: http://ieeexplore.ieee.org/document/8276867/.

[36] A. Salvador, J. Nogueira, and S. Sargento, "QoE assessment of HTTP adaptive video streaming", in *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST*, vol. 146, Springer, Cham, Nov. 2015, pp. 235–242, ISBN: 9783319188010. DOI: 10.1007/978-3-319-18802-7_32. [Online]. Available: http://link.springer.com/10.1007/978-3-319-18802-7%7B%5C_%7D32.

[37] Recommendation Itu-, "R-REC-BT.500 Methodology for the subjective assessment of the quality of television pictures", no. 11, pp. 500–11, 1974. [Online]. Available: https://www.itu.int/rec/R-REC-BT.500-13-201201-I.

[38] P. Brooks and B. Hestnes, "User measures of quality of experience: Why being objective and quantitative is important", *IEEE Network*, vol. 24, no. 2, pp. 8–13, Mar. 2010, ISSN: 08908044. DOI: 10.1109/MNET.2010.5430138. [Online]. Available: http://ieeexplore.ieee.org/document/5430138/.

[39] M. Mu, P. Romaniak, A. Mauthe, M. Leszczuk, L. Janowski, and E. Cerqueira, "Framework for the integrated video quality assessment", *Multimedia Tools and Applications*, vol. 61, no. 3, pp. 787–817, Dec. 2012, ISSN: 1380-7501. DOI: 10.1007/s11042-011-0946-3. [Online]. Available: http://link.springer.com/10.1007/s11042-011-0946-3.

[40] C. Moldovan and F. Metzger, "Bridging the Gap between QoE and User Engagement in HTTP Video Streaming", in *2016 28th International Teletraffic Congress (ITC 28)*, IEEE, Sep. 2016, pp. 103–111, ISBN: 978-0-9883-0451-2. DOI: 10.1109/ITC-28.2016.122. [Online]. Available: http://ieeexplore.ieee.org/document/7809639/.

[41] Shuai Zhao, Z. Li, D. Medhi, P. Lai, and Shan Liu, "Study of user QoE improvement for dynamic adaptive streaming over HTTP (MPEG-DASH)", in *2017 International Conference on Computing, Networking and Communications (ICNC)*, IEEE, Jan. 2017, pp. 566–570, ISBN: 978-1-5090-4588-4. DOI: 10.1109/ICCNC.2017.7876191. [Online]. Available: http://ieeexplore.ieee.org/document/7876191/.

[42] T. Phan-Xuan and E. Kamioka, "Network management based QoE estimation for adaptive streaming over HTTP", in *2016 17th International Telecommunications Network Strategy and Planning Symposium (Networks)*, IEEE, Sep. 2016, pp. 182–187, ISBN: 978-1-4673-8991-4. DOI: 10.1109/NETWKS.2016.7751173. [Online]. Available: http://ieeexplore.ieee.org/document/7751173/.

[43] D. J. Botia and N. Gaviria, "Strategies for Improving the QoE Assessment over iTV Platforms Based on QoS Metrics", in *2012 IEEE International Symposium on Multimedia*, IEEE, Dec. 2012, pp. 483–484, ISBN: 978-1-4673-4370-1. DOI: 10.1109/ISM.2012.98. [Online]. Available: http://ieeexplore.ieee.org/document/6424711/.

[44] A. Ben Letaifa, G. Maher, and S. Mouna, "ML based QoE enhancement in SDN context: Video streaming case", in *2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC)*, IEEE, Jun. 2017, pp. 103–108, ISBN: 978-1-5090-4372-9. DOI: 10.1109/IWCMC.2017.7986270. [Online]. Available: http://ieeexplore.ieee.org/document/7986270/.

[45] L. Amour, M. S. Mushtaq, S. Souihi, and A. Mellouk, "QoE-Based Framework to Optimize User Perceived Video Quality", in *2017 IEEE 42nd Conference on Local Computer Networks (LCN)*, IEEE,

Oct. 2017, pp. 599–602, ISBN: 978-1-5090-6523-3. DOI: 10.1109/LCN.2017.96. [Online]. Available: http://ieeexplore.ieee.org/document/8109418/.

[46] H. Nam, K. H. Kim, J. Y. Kim, and H. Schulzrinne, "Towards QoE-aware video streaming using SDN", in *2014 IEEE Global Communications Conference, GLOBECOM 2014*, IEEE, Dec. 2014, pp. 1317–1322, ISBN: 9781479935116. DOI: 10.1109/GLOCOM.2014.7036990. [Online]. Available: http://ieeexplore.ieee.org/document/7036990/.

[47] C. Ge, N. Wang, G. Foster, and M. Wilson, "Toward QoE-Assured 4K Video-on-Demand Delivery Through Mobile Edge Virtualization With Adaptive Prefetching", *IEEE Transactions on Multimedia*, vol. 19, no. 10, pp. 2222–2237, Oct. 2017, ISSN: 1520-9210. DOI: 10.1109/TMM.2017.2735301. [Online]. Available: http://ieeexplore.ieee.org/document/8000391/.

[48] F. Bronzino, D. Stojadinovic, C. Westphal, and D. Raychaudhuri, "Exploiting network awareness to enhance DASH over wireless", in *2016 13th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, IEEE, Jan. 2016, pp. 1092–1100, ISBN: 978-1-4673-9292-1. DOI: 10.1109/CCNC.2016.7444942. [Online]. Available: http://ieeexplore.ieee.org/document/7444942/.

[49] V. Krishnamoorthi, N. Carlsson, D. Eager, A. Mahanti, and N. Shahmehri, "Bandwidth-aware Prefetching for Proactive Multi-video Preloading and Improved HAS Performance", in *Proceedings of the 23rd ACM international conference on Multimedia - MM '15*, New York, New York, USA: ACM Press, 2015, pp. 551–560, ISBN: 9781450334594. DOI: 10.1145/2733373.2806270. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2733373.2806270.

[50] P. Casas and S. Wassermann, "Improving QoE prediction in mobile video through machine learning", in *2017 8th International Conference on the Network of the Future (NOF)*, IEEE, Nov. 2017, pp. 1–7, ISBN: 978-1-5386-0554-7. DOI: 10.1109/NOF.2017.8251212. [Online]. Available: http://ieeexplore.ieee.org/document/8251212/.

[51] A. Al-Jawad, P. Shah, O. Gemikonakli, and R. Trestian, "Policy-based QoS Management Framework for Software-Defined Networks", in *2018 International Symposium on Networks, Computers and Communications (ISNCC)*, IEEE, Jun. 2018, pp. 1–6, ISBN: 978-1-5386-3779-1. DOI: 10.1109/ISNCC.2018.8530994. [Online]. Available: https://ieeexplore.ieee.org/document/8530994/.

[52] P. Anchuen, P. Uthansakul, and M. Uthansakul, "QOE model in cellular networks based on QOS measurements using Neural Network approach", in *2016 13th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*, IEEE, Jun. 2016, pp. 1–5, ISBN: 978-1-4673-9749-0. DOI: 10.1109/ECTICon.2016.7561318. [Online]. Available: http://ieeexplore.ieee.org/document/7561318/.

[53] L. Borges de Moraes, A. Fiorese, and F. Matos, "A Multi-criteria Scoring Method based on Performance Indicators for Cloud Computing Provider Selection", in *Proceedings of the 19th International Conference on Enterprise Information Systems*, SCITEPRESS - Science and Technology Publications, 2017, pp. 588–599, ISBN: 978-989-758-247-9. DOI: 10.5220/0006289305880599. [Online]. Available: http://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0006289305880599.

[54] T. L. Saaty and J. M. Katz, "How to make a decision: The Analytic Hierarchy Process", Tech. Rep., 1990, pp. 9–11. [Online]. Available: http://vpp.sbuf.se/Public/Documents/ProjectDocuments/06f167ef-b243-48ed-8c45-f7466b3136eb/WebPublishings/How%20to%20make%20decision%20AHP.pdf.

[55] T. L. Saaty, "Decision making — the Analytic Hierarchy and Network Processes (AHP/ANP)", *Journal of Systems Science and Systems Engineering*, vol. 13, no. 1, pp. 1–35, Mar. 2004, ISSN: 1004-3756. DOI: 10.1007/s11518-006-0151-5. [Online]. Available: http://link.springer.com/10.1007/s11518-006-0151-5.

[56] *about | Alpine Linux*. [Online]. Available: https://www.alpinelinux.org/about/.

[57] B. A. Warsaw, *flufl.lock - An NFS-safe file lock — flufl.lock 3.2 documentation.* [Online]. Available: https://flufllock.readthedocs.io/en/latest/README.html.

[58] Chollet François, *Keras: The Python Deep Learning library*, 2015. DOI: 10.1086/316861. [Online]. Available: https://keras.io/.

[59] TensorFlow, *TensorFlow: An open source machine learning framework for everyone*, 2017. [Online]. Available: https://www.tensorflow.org/.