



**Tiago  
Oliveira**

**Otimização de Distribuição de Conteúdos  
Multimédia utilizando Software-Defined Networking  
Multimedia Content Distribution Optimization using  
Software-Defined Networking**





**Tiago  
Oliveira**

**Otimização de Distribuição de Conteúdos  
Multimédia utilizando Software-Defined Networking**

**Multimedia Content Distribution Optimization using  
Software-Defined Networking**

Dissertação 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 com Agregação do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro, e do Doutor Adriano Fiorese, Professor auxiliar convidado da Universidade do Estado de Santa Catarina.



**o júri / the jury**

presidente / president

**Professor Doutor Paulo Miguel Nepomuceno Pereira Monteiro**

Professor Associado do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro

vogais / examiners committee

**Professora Doutora Susana Isabel Barreto de Miranda Sargento**

Professora Associada com Agregação do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro (orientadora)

**Professora Doutora Marília Pascoal Curado**

Professora Associada com Agregação do Departamento de Engenharia Informática da Universidade de Coimbra (arguente)



## **agradecimentos / acknowledgements**

A conclusão deste importante percurso só foi possível com o apoio de várias pessoas. Sem a ajuda da minha família jamais estaria aqui. À Joana Laranjeira, por acreditar por mim nos momentos mais complicados, pela ajuda ao longo deste percurso, pois não estaria aqui sem ela. Quero também agradecer à professora Susana Sargento por toda a orientação dada, pela atenção e pela oportunidade de fazer parte de um grupo de investigação chamado NAP. Também a todos os meus colegas do NAP, um obrigado. Ao meu coorientador, Adriano Fiorese, um obrigado pela ajuda. Ao Guilherme Cardoso, pelo impacto e apoio no meu percurso académico. Ao André Dias pelo companheirismo. Um agradecimento também a todos colegas do #workers. Aos Biotecos pelas tertúlias na cantina ao almoço. À Rita e à Liliana da minha residência, um obrigado pela companhia. Um especial agradecimento à Bárbara Cruzeiro, pela paciência, carinho e ajuda na recta final deste percurso. E a tantos outros que também fizeram parte do meu percurso académico, obrigado.



## Palavras Chave

Televisão Digital, IPTV, OTT, DASH, redes SDN, Qualidade-de-Experiência, Balanceamento de Carga.

## Resumo

A generalização do acesso à Internet e equipamentos pessoais como smart-phones, tablets e computadores pessoais, está a criar uma nova onda de consumo de conteúdos multimedia. Nas ultimas duas décadas, a indústria de transmissão de Televisão atravessou várias evoluções e alterações, evoluindo da distribuição analógica para a digital, de canais de Televisão de definição padrão para alta definição, do método de distribuição IPTV, até ao último conjunto de tecnologias na distribuição de conteúdos, OTT. A tecnologia IPTV introduziu novas funcionalidades que mudaram o papel passivo do cliente para um papel activo, revolucionando a forma como os utilizadores consomem conteúdos televisivos. Assim, os hábitos dos clientes começaram a moldar os serviços oferecidos, levando à oferta de consumo de conteúdos em qualquer lugar e em qualquer altura. A entrega de vídeo OTT é um reflexo destes hábitos, indo ao encontro dos desejos dos utilizadores, que introduz inúmeras vantagens sobre outras tecnologias discutidas neste trabalho. No entanto, a entrega de conteúdos OTT cria diversos problemas de escalabilidade e ameaça o modelo de negócio das Operadoras de Telecomunicações, porque os fornecedores de serviço OTT usam a infraestrutura das mesmas sem quaisquer custos. Consequentemente, os Operadores de Telecomunicações devem preparar a sua infraestrutura para o consumo futuro ao mesmo tempo que oferecem novos serviços para se manterem competitivos. Esta dissertação visa contribuir com conhecimento sobre quais alterações uma Operadora de Telecomunicações deve executar com o modelo de previsão de largura de banda proposto. Os resultados obtidos abriram caminho para o método de entrega de conteúdos multimedia proposto, que visa ao melhoramento da qualidade de experiência do utilizador ao mesmo tempo que se optimiza o processo de balanceamento de carga. No geral os testes confirmam uma melhoria na qualidade de experiência do utilizador usando o método proposto.



**Keywords**

Digital Television, IPTV, OTT, DASH, SDN networks, Quality-of-Experience, Load Balancing.

**Abstract**

The general use of Internet access and user equipments, such as smart-phones, tablets and personal computers, is creating a new wave of video content consumption. In the past two decades, the Television broadcasting industry went through several evolutions and changes, evolving from analog to digital distribution, standard definition to high definition TV-channels, from the IPTV method of distribution to the latest set of technologies in content distribution, OTT. The IPTV technology introduced features that changed the passive role of the client to an active one, revolutionizing the way users consume TV content. Thus, the clients' habits started to shape the services offered, leading to an anywhere and anytime offer of video content. OTT video delivery is a reflection of those habits, meeting the users' desire, introducing several benefits discussed in this work over the previous technologies. However, the OTT type of delivery poses several challenges in terms of scalability and threatens the Telecommunications Operators business model, because OTT companies use the Telcos infrastructure for free. Consequently, Telecommunications Operators must prepare their infrastructure for future demand while offering new services to stay competitive. This dissertation aims to contribute with insights on what infrastructure changes a Telecommunications Operator must perform with a proposed bandwidth forecasting model. The results obtained from the forecast model paved the way to the proposed video content delivery method, which aims to improve users' perceived Quality-of-Experience while optimizing load balancing decisions. The overall results show an improvement of users' experience using the proposed method.



# Contents

<b>Contents</b>	<b>i</b>
<b>List of Figures</b>	<b>v</b>
<b>List of Tables</b>	<b>vii</b>
<b>Glossary</b>	<b>ix</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Motivation . . . . .	1
1.2 Objectives and Contributions . . . . .	2
1.3 Document Organization . . . . .	2
<b>2 State of the Art</b>	<b>5</b>
2.1 Multimedia Technologies . . . . .	5
2.1.1 TV Broadcasting . . . . .	5
2.1.2 Streaming . . . . .	6
2.1.3 Adaptive Streaming Protocols . . . . .	8
2.2 Multimedia Infrastructures . . . . .	13
2.2.1 TV Cable Networks . . . . .	13
2.2.2 IPTV Networks . . . . .	14
2.2.3 CDNs . . . . .	15
2.2.4 OTT Networks . . . . .	16
2.3 Multimedia Services . . . . .	18
2.4 SDN . . . . .	20
2.4.1 Defining SDN . . . . .	20
2.4.2 Terminology . . . . .	22
2.4.3 SDN Architecture . . . . .	23
2.4.4 Evaluation of SDN Controllers . . . . .	27
2.5 QoE . . . . .	29

2.6	Related Work . . . . .	30
2.6.1	Bandwidth Forecast . . . . .	30
2.6.2	SDN Load Balancing . . . . .	30
2.7	Summary . . . . .	31
<b>3</b>	<b>Forecasting OTT Bandwidth Consumption</b>	<b>33</b>
3.1	The Proposed Forecasting Model . . . . .	33
3.1.1	Multicast vs Unicast Model Envisioned Architecture . . . . .	34
3.1.2	Approaches . . . . .	35
3.1.3	Implementation . . . . .	36
3.1.4	OPEX Estimation . . . . .	37
3.1.5	Platform . . . . .	38
3.2	Experiments . . . . .	42
3.2.1	Portuguese Scenario . . . . .	42
3.2.2	North American Scenario . . . . .	45
3.3	Summary . . . . .	47
<b>4</b>	<b>Proposed Content Delivery Method and Architecture</b>	<b>49</b>
4.1	Proposed Content Delivery Method . . . . .	49
4.2	Client Player . . . . .	51
4.2.1	QoE Probe . . . . .	51
4.2.2	SDN Communication . . . . .	52
4.3	SDN Load Balancing . . . . .	53
4.3.1	Processing Packets . . . . .	53
4.3.2	Registering Clients . . . . .	54
4.3.3	Managing Clients . . . . .	55
4.4	Summary . . . . .	59
<b>5</b>	<b>Implementation</b>	<b>61</b>
5.1	Clients . . . . .	61
5.2	Origins . . . . .	64
5.2.1	Content Preparation . . . . .	64
5.2.2	Content Transmission . . . . .	64
5.3	SDN Controller . . . . .	66
5.3.1	The Controller . . . . .	66
5.3.2	Deployment . . . . .	67
5.4	Summary . . . . .	68
<b>6</b>	<b>Proof of Concept and Evaluation</b>	<b>69</b>

6.1	Scenarios . . . . .	69
6.2	Testbed . . . . .	73
6.2.1	Hardware . . . . .	73
6.2.2	Software & Challenges . . . . .	73
6.3	Evaluation . . . . .	79
6.3.1	Adaptive Bitrate (ABR) Algorithms Evaluation . . . . .	79
6.3.2	Evaluation of the Proposed Content Delivery Approach . . . . .	81
6.4	Summary . . . . .	90
<b>7</b>	<b>Conclusions and Future Work</b>	<b>91</b>
	<b>References</b>	<b>95</b>
	<b>Appendix</b>	<b>101</b>



# List of Figures

2.1	HyperText Transfer Protocol (HTTP)-based Adaptive Bitrate Streaming [12] . . . . .	8
2.2	Smooth Streaming Sequence [14] . . . . .	9
2.3	Components of an HTTP Live Stream (HLS)[15] . . . . .	10
2.4	Evolution of Adaptive Streaming Protocols and Standards [17] . . . . .	11
2.5	Media Presentation Description (MPD) [17] . . . . .	11
2.6	MSE Browser compatibility [21] . . . . .	12
2.7	Hybrid Fiber-Coaxial example (HFC) [24] . . . . .	13
2.8	Telco voice services integrated with video services, IPTV [27] . . . . .	14
2.9	The unmanaged network problem, at the right-side the ISP network [31] . . . . .	17
2.10	Layered view of an SDN [33] . . . . .	21
2.11	Traditional structure versus SDN structure [35] . . . . .	23
2.12	OpenFlow enable devices [35] . . . . .	25
2.13	SDN versus traditional networking [33] . . . . .	26
2.14	SDN Controllers feature comparison [39] . . . . .	28
3.1	Vision for cache and buffer implementation . . . . .	35
3.2	Altice Headend and SR values . . . . .	36
3.3	Initial Platform Menu . . . . .	39
3.4	Extended options for growth functionality . . . . .	40
3.5	Extended options for more flexibility . . . . .	40
3.6	Extended options for Custom forecast . . . . .	41
3.7	PT SR bandwidth uplink forecast 2017 . . . . .	43
3.8	PT SR and OLT forecast 2017 . . . . .	43
3.9	PT OPEX forecast 2017 . . . . .	44
3.10	PT Total SR uplink bandwidth 2020 . . . . .	45
3.11	PT OPEX forecast 2020 . . . . .	45
3.12	USA SR and OLT forecast 2017 . . . . .	46
3.13	USA OPEX forecast 2017 . . . . .	46
3.14	USA OPEX forecast 2020 . . . . .	47

4.1	Traditional Load Balancing versus SDN proposed method . . . . .	50
4.2	SDN Load Balancing Overview . . . . .	51
4.3	SDN load balancing overview . . . . .	53
4.4	Client registration high-level view . . . . .	55
4.5	Client and Origin table fields . . . . .	56
4.6	Client restarting the Transmission Control Protocol (TCP) session with the new Origin .	58
4.7	Controller terminating the TCP session with the old origin . . . . .	59
5.1	Client's Player components integration . . . . .	63
5.2	Content preparation and transmission overview . . . . .	65
5.3	Ryu framework overview . . . . .	66
5.4	Mininet and Open vSwitch integration . . . . .	68
6.1	Baseline vs Proposed Infrastructure . . . . .	70
6.2	Baseline vs Proposed Infrastructure with Load Creator . . . . .	72
6.3	Clients' Desktop . . . . .	74
6.4	Vegeta 13 requests comparison, on HAProxy with Ubuntu 16 and 18 . . . . .	75
6.5	Vegeta 15 requests comparison, on HAProxy with Ubuntu 16 and 18 . . . . .	76
6.6	Scenarios Orchestration Overview . . . . .	78
6.7	Wired scenario sub-type A between Buffer Occupancy based Lyapunov Algorithm (BOLA) and Throughput (ci=95) . . . . .	79
6.8	ADSL scenario sub-type C between BOLA and Throughput (ci=95) . . . . .	80
6.9	4G scenario sub-type D between BOLA and Throughput (ci=95) . . . . .	81
6.10	Wired scenario sub-type A between Baseline and SDN-based infrastructure, part 1 (ci=95)	82
6.11	Wired scenario sub-type C between Baseline and SDN-based infrastructure, part 1 (ci=95)	83
6.12	ADSL scenario sub-type D between Baseline and SDN-based infrastructure, part 1 (ci=95)	83
6.13	Average CPU of the Wired scenario, sub-type A, part 1 . . . . .	84
6.14	Buffer comparison of the ADSL-A scenario, part 2 (ci=95) . . . . .	85
6.15	Buffer and QoE comparison of the Wired-B scenario, part 2 (ci=95) . . . . .	85
6.16	Buffer and QoE comparison of the Regular 4G-D scenario, part 2 (ci=95) . . . . .	86
6.17	MOS comparison of the Regular 4G-E scenario, part 2 (ci=95) . . . . .	87
6.18	Buffer and QoE comparison of the Wired-A scenario, part 3 (ci=95) . . . . .	88
6.19	Total network write and read of the Wired-A scenario, with part 1 and part 3 testing . .	88
6.20	QoE comparison of the Regular 4G-B scenario, part 3 (ci=95) . . . . .	89
6.21	QoE comparison of the Regular 4G-C scenario, part 3 (ci=95) . . . . .	89
6.22	QoE, average QoE, and Buffer comparison between Wifi-C part 1 and part 3 on SDN infrastructure (ci=95) . . . . .	90

# List of Tables

6.1	Technology Scenarios . . . . .	71
6.2	Sub-Scenarios Throttle . . . . .	71
6.3	Testbed machines specification . . . . .	73



# Glossary

<b>3GPP</b>	3rd Generation Partnership Project	<b>GOP</b>	Group of Pictures
<b>4K</b>	Ultra High Definition	<b>HD</b>	High Definition
<b>AAC</b>	Advanced Audio Coding	<b>HDR</b>	High Dynamic Range
<b>ABR</b>	Adaptive Bitrate	<b>HE-ACC</b>	High-Efficiency Advanced Audio Coding
<b>API</b>	Application Programming Interface	<b>HEVC</b>	High Efficiency Video Coding
<b>APL</b>	Application Layer	<b>HFC</b>	Hybrid fiber-coaxial
<b>ARP</b>	Address Resolution Protocol	<b>HFC</b>	Hybrid Fiber-Coaxial
<b>ATM</b>	Asynchronous Transfer Mode	<b>HLS</b>	HTTP Live Streaming
<b>BOLA</b>	Buffer Occupancy based Lyapunov Algorithm	<b>HSTCP</b>	Hot Swap TCP
<b>CAPEX</b>	Capital Expenditure	<b>HTTP</b>	HyperText Transfer Protocol
<b>CATV</b>	Cable TV	<b>IETF</b>	Internet Engineering Task Force
<b>CATV</b>	Cable TV	<b>IIS</b>	Internet Information Services
<b>CDN</b>	Content Delivery Network	<b>IoT</b>	Internet-of-Things
<b>CLI</b>	Command Line Interface	<b>IP</b>	Internet Protocol
<b>CP</b>	Control Plane	<b>IPTV</b>	IP Television
<b>CR</b>	Core Router	<b>ISP</b>	Internet Service Provider
<b>DASH-IF</b>	DASH-Industry Forum	<b>ITU-T</b>	International Telegraph Union Telecommunication Standardization Sector
<b>DASH</b>	Dynamic Adaptive Streaming over HTTP	<b>Mbps</b>	Megabits per second
<b>DP</b>	Data Plane	<b>MDC</b>	Multiple Description Coding
<b>DRM</b>	Digital Rights Management	<b>MOS</b>	Mean Opinion Score
<b>DSL</b>	Digital subscriber line	<b>MPD</b>	Media Presentation Description
<b>DTT</b>	Digital Terrestrial Television	<b>MPEG-DASH</b>	MPEG Dynamic Adaptive Streaming over HTTP
<b>DVB-C</b>	DVB Cable	<b>MPEG</b>	Moving Picture Experts Group
<b>DVB-S</b>	DVB Sattelite	<b>MPLS</b>	Multi Protocol Label Switching
<b>DVB-T</b>	DVB Terrestrial	<b>MSE</b>	Media Source Extensions
<b>DVB</b>	Digital Video Broadcasting	<b>NAT</b>	Network Address Translation
<b>DVR</b>	Digital Video Recorder	<b>NBI</b>	Northbound Interface
<b>EST-VoD</b>	Electronic Sell-Through VoD	<b>NCPs</b>	Network Control Points
<b>ETSI</b>	European Telecommunications Standards Institute	<b>NOS</b>	Network Operating System
<b>FD</b>	Forwarding Devices	<b>OLT</b>	Optical Line Termination
<b>FHD</b>	Full HD	<b>OPEX</b>	Operational Expenditure
<b>FPS</b>	Frames Per Second	<b>OTT</b>	Over-the-Top
<b>FTTH</b>	Fiber-to-the-Home	<b>OVS</b>	Open vSwitch
		<b>OXM</b>	OpenFlow Extensible Match

<b>PVR</b>	Personal Video Recording	<b>SSL</b>	Secure Sockets Layer
<b>QoE</b>	Quality-of-Experience	<b>STB</b>	Set-top Box
<b>QoS</b>	Quality-of-Service	<b>SVC</b>	Scalable Video Coding
<b>QUIC</b>	Quick UDP Internet Connections	<b>T-VoD</b>	Transaction VoD
<b>REST</b>	Representational State Transfer	<b>TCP</b>	Transmission Control Protocol
<b>RTCP</b>	RTP Control Protocol	<b>Telco</b>	Telecommunication Operator
<b>RTP</b>	Real-time Transport Protocol	<b>Telcos</b>	Telecommunications Operators
<b>RTSP</b>	RTP Streaming Protocol	<b>TLS</b>	Transport Layer Security
<b>S-VoD</b>	Subscription VoD	<b>UDP</b>	User Datagram Protocol
<b>SBI</b>	Southbound Interface	<b>UHD</b>	Ultra High Definition
<b>SD</b>	Standard Definition	<b>URL</b>	Uniform Resource Locator
<b>SDK</b>	Software Development Kit	<b>VoD</b>	Video-on-Demand
<b>SDN</b>	Software-Defined Networking	<b>VoIP</b>	Voice over IP
<b>SLA</b>	Service Level Agreement	<b>WMA</b>	Windows Media Audio
<b>SR</b>	Service Router	<b>XML</b>	Extensible Markup Language

# Introduction

## 1.1 Motivation

Video content production is at a level never seen before, and customers are more demanding than ever. They want higher video quality, a broader video catalog, and a real anytime-anywhere offer. A forecast made by [1] predicts that, by 2021, 82% of all consumer Internet traffic generated will be video. Another obstacle faced by ISPs will be the continued decrease of Linear-TV in share, and by 2020, Video-on-Demand (VoD) and Catch-Up TV will claim almost half of the viewership time and half of the time a mobile screen will be the chosen device, which is an incompatible multicast device [2]. Therefore, native IP-Multicast, the Linear-TV type of transmission, becomes less attractive and a hurdle to maintain. OTT pure companies are already deploying new types of CDN (Content Delivery Network) solutions to handle bandwidth demand and rising subscribers, increasing the overall flexibility and enabling new features. On the ISPs side, multicast compromises the flexibility due to restraints like the need for multicast compatible hardware on the customer side. The Set-top Box (STB), in most cases, has proprietary standards and closed technologies making innovation hard. Hence, choosing a new way of content delivery will have a profound impact on ISPs infrastructures, top to bottom. At the beginning of the millennium, IP-multicast began to be adopted, and today it is still the most efficient way of delivering video to a group of subscribers inside an ISP network. However, this efficiency is only achievable if the content delivery is Live/Linear-TV, and off-schedule content like VoD and Catch-Up TV have been rising steadily. A study by Ericsson [2] indicates that roughly 60% of customers prefer off-schedule content. On the contrary, unicast transmissions are the most basic ones, using a one-to-one-type communication between a sender and a receiver. In other words, unicast does not scale well, because for a large number of receivers requesting some content, the infrastructure would have to be massive and most of the time idle. To deal with this problem, the design and deployment of CDNs has been used as the answer, bringing content closer to users allowing the utilization of unicast as the standard transmission way of non-linear content. Therefore,

with the characteristics introduced by CDNs, core network traffic can be reduced, achieving faster streaming times, improving the user Quality-of-Experience (QoE). Notwithstanding, if unicast type-of traffic is on the rise due to the increase of VoD, Catch-Up TV, and Internet video content, then, naturally, two critical questions emerge:

- Is it feasible to drop multicast altogether for unicast or another method of transmission?
- How to prepare an ISP content delivery infrastructure for the future?

To answer these questions, bandwidth estimation with data from ISP operators is essential, as well as estimation of the operational costs regarding transmission methods.

## 1.2 Objectives and Contributions

The main concern laid out in the motivation section was the fact that unicast content is rising, Video-on-Demand (VoD), Catch-Up TV and so on, and will continue to rise in the foreseeable future. These types of content have a greater impact on network resources than Live-TV which is distributed using the very efficient multicast networks. Thus, the work on this thesis is two-fold:

- A proposed forecasting model: considering Live-TV, Catch-Up TV, and VoD, three approaches for content delivery are compared: multicast, unicast-only and hybrid. The comparison between these approaches focuses on how bandwidth spending will grow in the future as well as a theoretical Operational Expenditure (OPEX) comparison. The findings of this work lead to the second part of this thesis;
- A proposed content delivery method: with the insight from the proposed forecasting model, where the new edge network hardware is a must to prepare an Internet Service Provider (ISP) infrastructure for the future, a new method for load balancing Over-the-Top (OTT) content is proposed. This method aims to improve the users' perceived Quality-of-Experience (QoE) while, at the same time, it aims to improve infrastructure resources.

The proposed forecasting model has been one of the results of the P2020 UltraTV project (Altice Labs, Universidade de Aveiro, and Instituto de Telecomunicações).

Also, the proposed forecasting model work resulted in a submitted, accepted and presented paper to the IEEE Symposium on Computers and Communications, June 2018 - Natal, Brazil.

## 1.3 Document Organization

The table of contents provides an overview of how the document is organized, but it is important to provide a little bit more context to each chapter:

- **Chapter 2, State of the Art:** presents a high-level overview of the digital Television evolution to the new OTT technologies and new network innovations (SDN);
- **Chapter 3, Forecasting OTT Bandwidth Consumption:** displays the proposed forecasting model architecture, implementation, evaluation and conclusions;

- **Chapter 4, Proposed Content Delivery Method and Architecture:** describes the proposed method and the architecture of both the client-side and the server-side components;
- **Chapter 5, Implementation:** presents the technologies and mechanisms used to implement the previously discussed components;
- **Chapter 6, Proof of Concept and Evaluation:** describes the scenarios elaborated to test the proposed approach, the challenges of creating a testbed for such scenarios, and finally, it discusses the obtained results;
- **Chapter 7, Conclusions and Future Work:** exposes a critic viewpoint of the results obtained and lays out future improvements and features of this work.



## State of the Art

*This chapter focuses on the evolution of multimedia content delivery, from digital Television broadcasting to HTTP adaptive streaming and all of the related challenges. Further, due to the importance of SDN networks in this work, an in-depth analysis is made. Additionally, Quality-of-Experience and its challenges are described as well. The chapter divides into multimedia technologies, infrastructures, services, SDN networks, and Quality-of-Experience.*

### 2.1 Multimedia Technologies

This section will describe the major technologies of content distribution, from the early days of Television broadcasting to HTTP adaptive streaming.

#### 2.1.1 TV Broadcasting

Television broadcasting started to transition to digital in the 1990s with a project called Digital Video Broadcasting (DVB) Project [3], that included more than 200 companies between broadcasters, network operators, regulatory entities and others. The DVB Project started in Europe, but now it is a worldwide project with the common goal of standardizing specifications for digital media delivery systems. However, in 1990, digital television broadcasting was considered unfeasible and expensive to deploy. The year 1991 brought manufacturers and broadcasters to discuss how to build a platform that would lead to a standard that fitted digital broadcasting needs, forming the European Launching Group that would later become the DVB Project that started in 1993. The project's central philosophy was to develop a complete suite of technologies that would serve as a foundation for both satellite, cable, and terrestrial broadcasting, avoiding one-to-one technologies that would break if used on different realms. In sum, the DVB Project created multimedia containers that would work on different mediums of distribution. The DVB Sattelite (DVB-S) and DVB Cable (DVB-C) were the first to be deployed due to market priorities, DVB Terrestrial (DVB-T) came last. Moreover, terrestrial broadcasting usually belongs to the public sector rather than the private sector,

and when compared to the other two, DVB-T had more constraints of deployment due to its size and coverage. The first DVB broadcast was from a DVB-S system, and it was in 1995 by a pay-TV french operator [4]. The first DVB-T broadcast began in 1998 in Sweden and the UK. Portugal began working on DVB-T in 1999 [5], coordinating efforts with Spain due to frequency planning between the two countries, but only in 2009, Portugal started transmitting in compliance with DVB-T, with DVB-T2 freshly standardized [6]. The next generation of DVB brought first the DVB-S2 in 2003 [7] following the same trend of developing first the technology for satellites, with overall improvements and a gain of  $\approx 36\%$  of useful bitrate. Then the tradition (first satellite, then cable and lastly terrestrial) was stopped because DVB-T2 arrived first, in 2009, not DVB-C2, that arrived in mid-2010 [8]. DVB-T2 brought a more robust signal, improving the maximum data rate of 40 Mbit/s beating the 31.7 Mbit/s of DVB-T. The DVB-C2 evolution came later due to the evolving market drivers. The cable TV is stagnating [9] and seems that it is not part of the future convergence comparing, for instance, to the Fiber-to-the-Home (FTTH) medium. One of the key reasons for the evolution was the need to keep up with the other two evolutions (DVB-S2, DVB-T2) because some Cable TV (CATV) networks are retransmitting to satellites for example. More than twenty years later, the DVB Project is considered successful with deployments all over the world. The DVB-T is the least adopted around the world, being Europe the main adopter, and DVB-S is the most adopted worldwide. In total, around the world, it is estimated that nearly 1 Billion of DVB receivers are in use [4].

### 2.1.2 Streaming

Streaming can be defined as a continuous flow of data from a sender to a capable receiver, sharing the same medium, usually a video transmission on an IP network. In a non-streaming scheme, to exist any content play, the sender has to send all the data before the receiver can play it. Since the late 90s, streaming began to gain popularity with the RTP Streaming Protocol (RTSP) [10], which was standardized the same year of the first terrestrial digital broadcast in 1998. However, the streaming boom would come with the increase of Internet speeds, pushed by the adoption of Digital subscriber line (DSL), 3G and later Hybrid Fiber-Coaxial (HFC), pushing video compression technology, that made video content delivery through IP networks achievable. Since then, streaming matured enough to compete and disrupt the multimedia delivery industry [11].

#### Traditional Streaming

The creation of RTSP was a response to the growing popularity of multimedia streaming over IP networks. It works alongside other two protocols, Real-time Transport Protocol (RTP) and RTP Control Protocol (RTCP) to meet the specific needs of real-time on-demand delivery. It can control multiple data sessions and is able to choose between User Datagram Protocol (UDP) and TCP on a RTP data channel. The RTCP helps RTSP by getting QoS metrics such as packet loss and delay. Adding all features, RTSP Protocol has some hurdles regarding

scalability issues because each session must be managed. For a service like YouTube would be impossible to use this protocol. Also, the use of UDP packets can be fructiferous to avoid video stalls. However, using those types of packets on an unmanaged network where there are no guarantees of a path open to the UDP packets, those packets can be dropped, making video playback impossible. These types of problems limit the use of RTSP on modern multimedia services, though in other scenarios like Voice over IP (VoIP) it is still widely used.

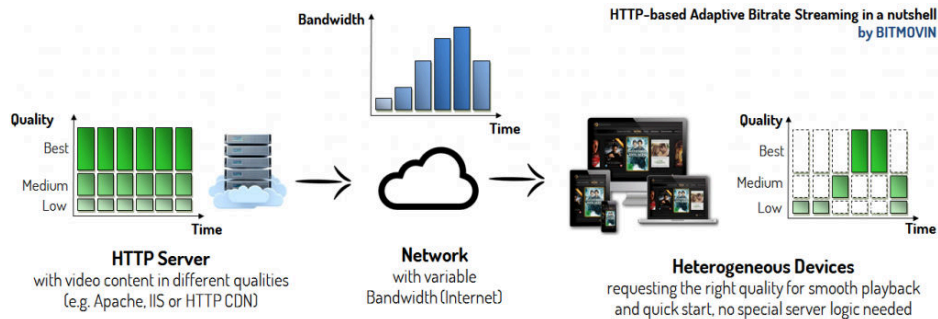
## **Progressive Download**

Progressive download is a technique that transforms a bulk of data into a readable as you get data. Ordinarily it is used over HTTP (HTML5 has native support) or in P2P networks. The client controls the playback, and it needs a "warm-up" to feed sufficient data to the decoder, intra-frames, so that the subsequent data can have frames depending on past or future frames, P-frames and B-frames respectively. It is possible to do operations like seeking because codec video formats have intra-frames along the video to make seeking possible, and therefore making seeking possible. Because progressive download can work through HTTP and HTTP is stateless, the use of proxy servers, Content Delivery Network (CDN)s is possible, as well as traversing throughout the entire network without being blocked by a firewall (HTTP is generally allowed in every firewall). These characteristics are essential to scale unicast traffic. Comparing with RTSP some features are missing like the gathering of Quality-of-Service (QoS) metrics to improve connection and live-streaming, but because of its scalability and ease of access, it became widely adopted for streaming on-demand content.

## **Adaptive HTTP-based Streaming**

In a world of unmanaged networks and mobility scenarios, conditions can have a significant deal of variations, and as humans adapt to their surroundings, the content must adapt as well to the network conditions, like bandwidth, latency, jitter and packet loss. Streaming adaptation is essential, especially nowadays given the mobility of equipments that can play content. Therefore content adaptation is critical to improve the observed user QoE. However, for a real adaptation to be plausible, it is needed the type of metrics that RTSP provides, the scalability from the progressive download and different types of encoding techniques. Multiple Description Coding (MDC) and Scalable Video Coding (SVC) brought a new way to encode multiple qualities for a video with slightly distinct approaches, but the core idea is to encode in different qualities to send more or fewer data depending on the client capabilities, building frames on the client side with more or less quality. This progress paved the way for the natural evolution of progressive download technique, the segmented HTTP-based streaming. The concept behind this new method is to take advantage of scalable and complementary encoding techniques and apply it to the HTTP realm. The core principle is to encode the original content into different quality streams and break those streams segments or "chunks" that can have 2 to 10 seconds of video content. Each chunk has an intra-frame at the beginning, so it can be decoded independently from other chunks; and chunks are numbered, making it

possible to switch qualities without affecting the timeline of the video content. This granularity benefits the playback avoiding stalls or "drag" in the video. One of the main advantages is that the client controls which chunks to download, which quality stream to choose based on its conditions, based on the equipment capability.



**Figure 2.1:** HTTP-based Adaptive Bitrate Streaming [12]

Therefore, server-side computational resources are saved because the client controls what to download, since the client is in the best position to assess its environment [13]. Nonetheless, the client does not have 100% of control, and it is common for video players running on the client to have restrictions on buffer size, due to the fact that sometimes the client will not see the totality of the video, wasting server capacity. Considering the approaches addressed, segment HTTP-based streaming adds upon the missing features of progressive download that RTSP provided, maintaining the same scalability and advantages of being HTTP and, because of the symbiotic growth of HTTP and Internet, HTTP delivery is extremely optimized.

### 2.1.3 Adaptive Streaming Protocols

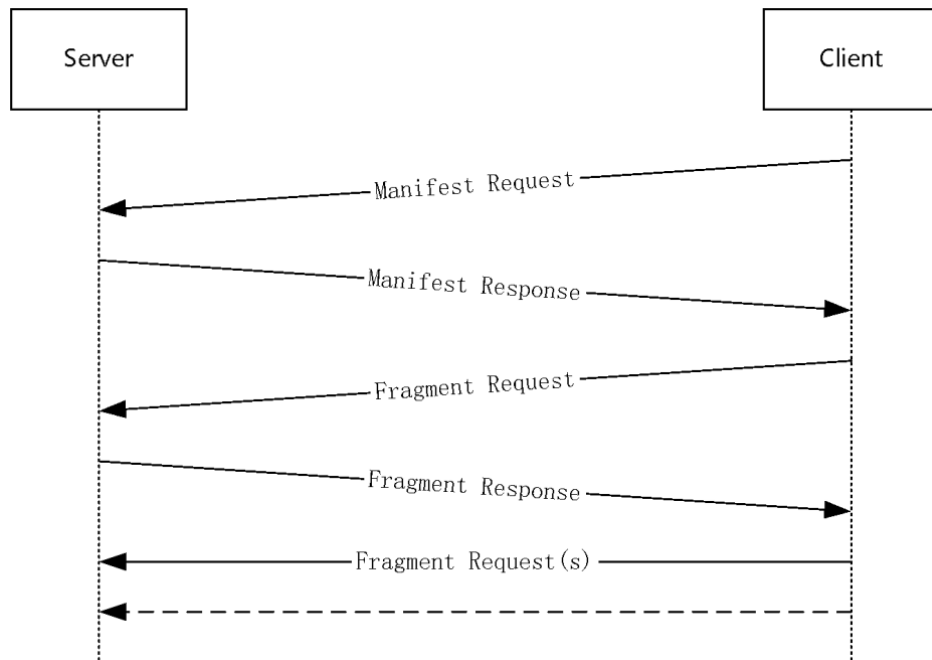
The popularity growth of adaptive HTTP streaming technologies lead significant industry players to develop their solutions:

- Microsoft defined Smooth Streaming;
- Adobe created HTTP Dynamic Streaming (HDS);
- Apple produced HTTP Live Streaming (HLS);
- Moving Picture Experts Group (MPEG) created the first Dynamic Adaptive Streaming over HTTP (DASH) ISO.

#### Microsoft Smooth Streaming

Developed by Microsoft, Smooth Streaming is part of the solutions offered in the realm of web servers, more specifically the Microsoft Internet Information Services (IIS) Media Services. Based on the fragmented MPEG-4 standard, it supports Windows Media Audio (WMA), Advanced Audio Coding (AAC) for audio codecs and H.264/VC-1 as video codecs and has a maximum resolution of "1080p" [14]. Smooth streaming is essentially an integrated proprietary take on the adaptive HTTP-based streaming, divided into three main components: the encoder, Microsoft Expression Encoder, the server, Microsoft's IIS Media Services, and the client, a

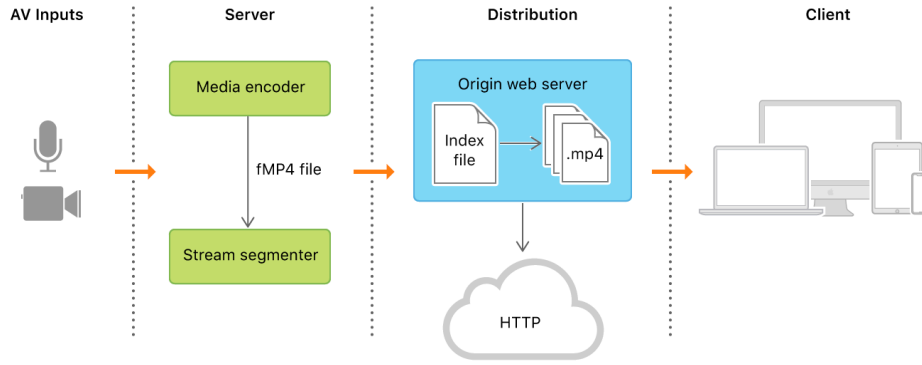
Microsoft Silverlight player that runs on Microsoft platforms (but a Software Development Kit (SDK) provides means to implement players for other platforms). This integrated solution was targeted to pay TV operators because of the Digital Rights Management (DRM) support and features that Digital Video Recorder (DVR) usually offer, making this solution adequate for commercial deploy. Figure 2.2 depicts a typical communication sequence between a IIS Media Service web server and a Smooth Streaming client.



**Figure 2.2:** Smooth Streaming Sequence [14]

### Apple HTTP Live Streaming

Started as an Internet Engineering Task Force (IETF) Internet Engineering Task Force draft, it was early on picked up by Apple to also, as Microsoft, push its own standard of HTTP-based streaming. HLS supports H.264/AAC and as of 2017, it also supports High Efficiency Video Coding (HEVC)/H.265/High Dynamic Range (HDR) video codecs and Dolby Atmos audio codec [15]. Differentiating from Smooth Streaming, HLS uses MPEG-2 Transport Streams instead of fragmented MPEG-4, and the media files description is done by "m3u8" playlists. Comparing with Smooth Streaming, HLS gained more traction over the Internet because of the millions of Apple devices that support it by default.



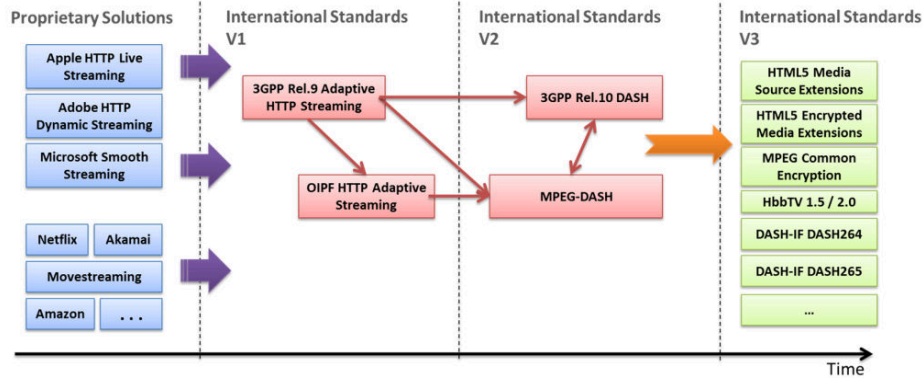
**Figure 2.3:** Components of an HTTP Live Stream (HLS)[15]

### Adobe HTTP Dynamic Streaming

To complete its suite of media services, Adobe also created its own take of segmented HTTP-based streaming, creating a new container, the F4V, which contains video encoded in H.264/AAC, using the MPEG-4 standard to create chunks. The technology is very similar to HLS and Smooth Streaming, but 100% incompatible; only Adobe Flash Player or players based on Adobe Integrated Runtime can reproduce HDS streams.

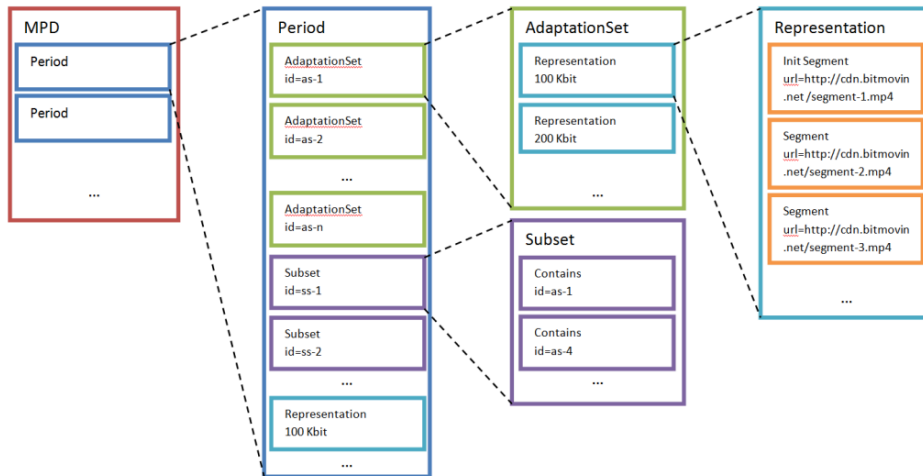
### MPEG Dynamic Adaptive Streaming over HTTP

The same companies that created their own adaptive streaming HTTP protocols, and others that one way or another depended on those solutions, started to realize that the market was too fragmented and convergence was the future for market growth, benefiting everyone. The MPEG Dynamic Adaptive Streaming over HTTP (MPEG-DASH) standard was created in 2012 with participating companies like Microsoft, Apple, Netflix, Qualcomm, Ericsson, Samsung, and others [16]. In 2014 it was revised [17] by the already then built DASH-Industry Forum (DASH-IF) foundation. The DASH-IF was established shortly after the first standard approval [16] and counts with over 67 members, like Microsoft, Adobe, Samsung, Qualcomm, Google, Akamai, Ericsson, LG, and others. The DASH-IF main focus is interoperability to avoid ecosystem fragmentation. That strategy worked, and DASH became fastly adopted, resulting in the integration of DASH on HTML5, Media Source Extensions, allowing DASH playback via HTML5 audio and video tag. Also, the 3rd Generation Partnership Project (3GPP) added support for DASH on its Release 10 [17]. Figure 2.4 shows the industry evolution of adaptive streaming protocols.



**Figure 2.4:** Evolution of Adaptive Streaming Protocols and Standards [17]

DASH introduced a new type of descriptor called MPD. This file is an Extensible Markup Language (XML) file that comprises the various qualities of video and audio content, pointing to the right HTTP Uniform Resource Locator (URL). It has a hierarchical organization, describing time *Periods* (start time and duration) that each can contain different *Adaptation Sets*. The time periods resolve one of the most challenging problems in the industry, dynamic ad-insertion. The adaptation sets group different configurations that can include different codecs, audio, subtitles, and others. Figure 2.5 shows the organization of a Media Presentation Description (MPD).





















**Figure 2.5:** Media Presentation Description (MPD) [17]

Due to the number of companies contributing to the DASH project, features were added from previous technologies and new features that create an extensive list that DASH holds. Here are some highlights:

- independent standard, not owned by any company;
- support for different ABR Strategies;
- multi-video and audio tracks to give the full DVD/Blu-Ray experience;
- common encryption, allowing multiple DRM technologies to coexist, even in different adaptation sets;

- seamless ad insertion, through the use of periods;
- ease of integration, DASH works on any HTTP server and media servers, not requiring any vendor-specific ecosystem;
- codec agnostic;
- support for quality metrics, the client can report them.

With all of the contributions and options designed in the standard specification, the DASH project struggled with its own flexibility and its large number of features, and the will to be codec agnostic, creating a problem of what technologies makes a DASH player. Therefore, the DASH-IF foundation started to create guidelines that worked as 'profiles', creating a baseline configuration with restrictions and linking a codec, so that the interoperability would actually be achieved in a fast way, like a standard inside a standard. The main profile created, "DASH-AVC/264", as the previous adaptive streaming technologies, used the H.264 video codec except for the audio, using the new High-Efficiency Advanced Audio Coding (HE-AAC) v2. In the process of finishing new profiles, like the "DASH-IF Ultra High Definition (UHD) HEVC 4K" or "DASH-IF VP9 UHD", it shows just how malleable and future-proof DASH is. Two major video distributors, YouTube and Netflix, that account for almost 50% of all video traffic generated globally, are already using DASH making a big push of this standard [18]. The only requirement that Dash.js from DASH-IF has, is that the web browser must support HTML5 Media Source Extensions (MSE) [19] [20]. Figure 2.6 shows which browser is compliant with MSE. Noticeable, Safari from iOS does not support MSE, but Safari from MacOS does. Internet Explorer has some support issues [21], and currently, MSE is disabled by default on Samsung Internet [22].

														
		 Chrome	 Edge	 Firefox	 Internet Explorer	 Opera	 Safari	 Android webview	 Chrome for Android	 Edge Mobile	 Firefox for Android	 Opera for Android	 Safari on iOS	 Samsung Internet
Basic support		31	Yes	42	11 ★	15	8	4.4.3	33	Yes	41	30	No	Yes
MediaSource		31	Yes	42	11 ★	15	8	4.4.3	33	Yes	41	30	No	Yes
sourceBuffers		31	12	42	11 ★	15	8	4.4.3	33	Yes	41	30	No	Yes

**Figure 2.6:** MSE Browser compatibility [21]

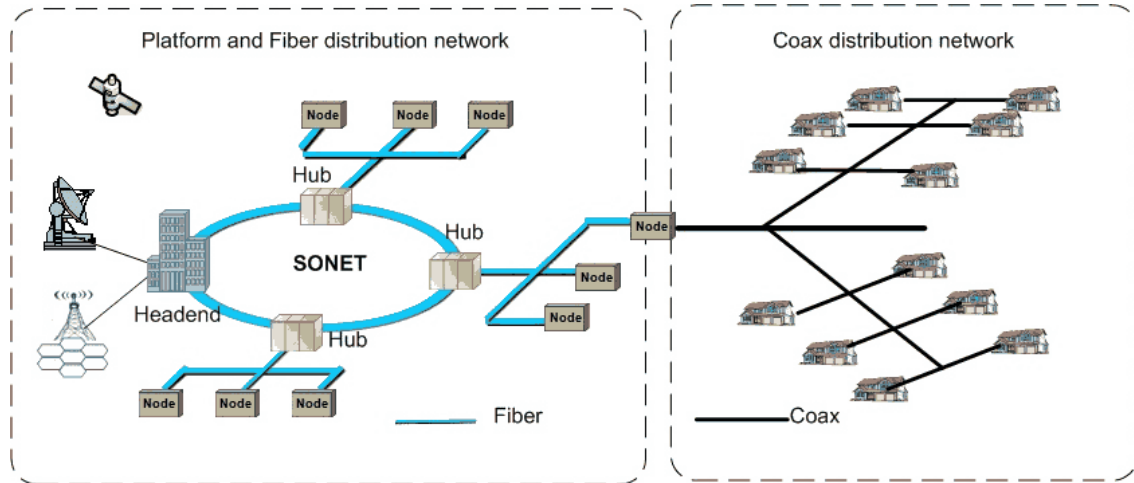
At the moment, DASH is still competing with HLS to become the *de facto* standard, but it has a broader interoperability offer and a bigger DRM ecosystem for vendors compared to HLS.

## 2.2 Multimedia Infrastructures

This section will describe the most common infrastructures used in multimedia content delivery and its most important components.

### 2.2.1 TV Cable Networks

The last significant evolution of the Cable TV was in mid-nineties when digital Television was standardized for CATV networks, the already discussed DVB-C. By that time, CATV operators were investing considerable capital to improve CATV networks to reach more customers and offer more channels [23], moving from all coaxial topologies to hybrid ones with fiber. This push happened because of the type of architecture that Cable TV has and shares with Digital Terrestrial Television (DTT) and Satellite TV: the broadcasting of all the channels from their headend to a single transport medium. Moreover, the TV channels encoding used at the time was MPEG-2, meaning that each channel would take at least five Megabits per second (Mbps) and multiplying with almost one hundred channels offered by the operator. The HFC networks were then developed to accommodate these requirements. Figure 2.7 depicts a HFC network, where a major fiber ring exists throughout a metropolitan area, having then fiber nodes that feed the coaxial distribution network, the client's point of access. The client would then need a Set-top Box (STB) that tunes to the right frequency to view a channel.

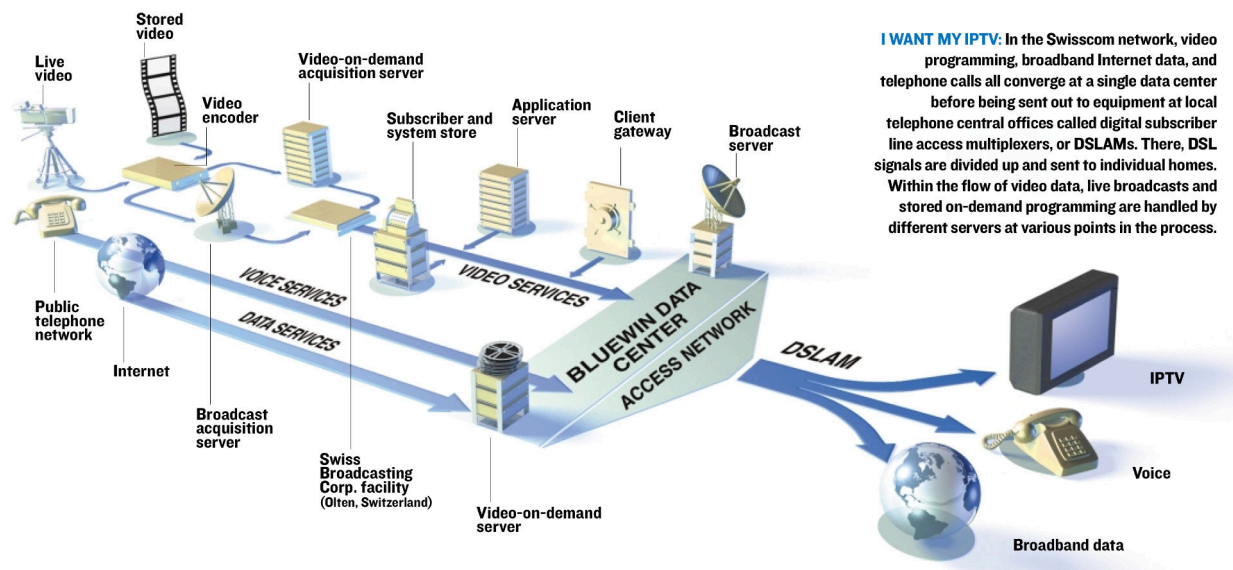


**Figure 2.7:** Hybrid Fiber-Coaxial example (HFC) [24]

With this improvement, Cable operators were in a unique position to offer Personal Communication Services as it was called at that time; today we merely refer to it as Internet access. With the advent of VoIP, CATV operators started to provide the *triple play* service, that combines television, telephone, and Internet access [25]. Consequently, Cable TV operators were no longer offering just Television and were creating a new market that would profoundly affect Telecommunications Operators (Telcos).

## 2.2.2 IPTV Networks

IP Television (IPTV) is a different take on TV-channels broadcast: a combination of technologies that make video content delivery on the realm of Internet Protocol (IP) networks a reality. Instead of delivering all of the channels at once, it delivers only the TV-channel requested by the client, reducing the network infrastructure requirements drastically. In fact, in this environment, although a set of minimum requirements exist, it is virtually possible to have an unlimited number of channels available to the client because the limitation does not come from the client side but in how many servers the operator has to fulfill the client's request [26]. Therefore, IPTV changed the distribution paradigm; the client interacts with the service, not like CATV in which the client only tunes-in the desired TV-channel, opening a world of new features. Microsoft started to push IPTV technology around 2004 [27] due to the Telcos necessity to compete with CATV operators ( and as a way to Microsoft diversify its products) to create a new market. Realizing that IPTV technology would require an interactive STB, essentially a media center computer, that Microsoft started to bring to market some years before. However, IPTV faced some critical challenges due to the client's medium of access limitations, the Telcos network: a pair of thin copper wires. DSL innovations combined with the new codec compression from Microsoft, reduced the 5 Mbps of MPEG-2 per TV-channel to 2 Mbps, making IPTV look achievable. Figure 2.8 depicts all of the network components allied with the IPTV components of the first European company to deploy IPTV.



**Figure 2.8:** Telco voice services integrated with video services, IPTV [27]

Besides the low bandwidth access, one more problem needed fixing, the start-up time of a TV-channel, that was slow in comparison with CATV that had all channels transmitted to the STB, due to the cost of changing the multicast group to another with the requested TV-channel. Microsoft overcame this technical difficulty with a special buffer: each time the client changes the channel, a burst of UDP packets are sent, fast starting the channel stream, later changing to the multicast stream when ready without the client noticing the

transition. Further, this technology enabled on-demand content due to the IPTV paradigm shift, radically changing how consumers watch Television, like Catch-Up TV. Later the European Telecommunications Standards Institute (ETSI) standardized the core elements of the IPTV technology, and CATV operators embraced this technology.

### 2.2.3 CDNs

CDNs are an essential element of the Web landscape since 1998 when first introduced [28]. With the exponential growth of Internet access and its services, web pages started to become more elaborate and with more content, and web servers were not up to the task. Therefore, CDNs aim to reduce the load on the servers at the same time satisfying the clients as fast as possible. The core principle is replicating the content from the source, usually called the origin server, to other servers called surrogate or replica servers. These types of networks are tailored for the kind of content they serve, that can be static web pages, video or other web services. The replica servers are strategically placed considering the geographical demand, that improves response times but at the same time reduces backbone traffic on the network. There are two approaches to implement CDNs: *overlay* and *network* (hybrid solutions with the two exist as well). The overlay approach uses specific servers or caches that handle the content distribution, creating virtual interconnections between the replicas. No core network infrastructure like switches and routers play an active role in content delivery, providing only basic connectivity and complying with QoS agreements for the CDN nodes. The network approach, as the name points out, depends on network equipment like routers and load balancers that directly participate in routing requests throughout the CDN network. Comparing the two, the overlay approach is more flexible, and for that reason, the most widely deployed. For the actual communication between replica servers, one or more protocols can be used like Cache Array Routing Protocol or the Hypertext Caching Protocol, but some vendors implement their own.

### Load Balancing

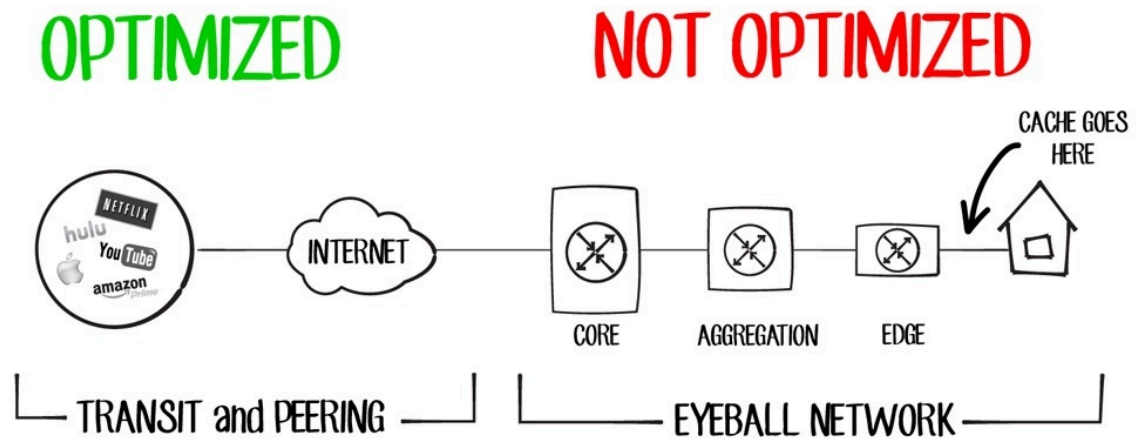
Load Balancing is one of the core foundations of CDNs, aiming for equal distribution of load among the replica servers while pointing the user request to the nearest one [29]. Nowadays, load balancers can exist in the form of hardware appliances or virtual appliances optimized for the specified platform and content. The primary tasks of a load balancer are to choose, based on application-specific methods of distributing network traffic, the best server for both the client and the infrastructure, fast and efficiently, and to constantly monitor the performance of the network servers to make better future decisions. Several techniques and algorithms exist to make the appropriate decisions depending on the type of content and the level of load. Here are some of the most commonly used:

- *Round-Robin*: one of the simplest and most used, Round-Robin, distributes requests in a rotating sequential manner, worrying only about even load distribution;
- *Weight Round-Robin*: builds on the idea of the previous method, but each server has a static numerical weight that rates the servers that can process more or fewer requests;

- *Least-Connection*: this method considers server load, the requests are routed to the server that has the least active sessions at the current time;
- *Source IP Hash*: a unique hash key is generated for a particular server with information of the source and destination IP address of the client request. Therefore, if the client connection drops, this method assures that the client is redirected to the same;
- *Choice-of Two*: this algorithm takes two servers randomly and chooses the one with the smallest load. This method keeps the simplicity of the Round-Robin while taking load into account.

#### 2.2.4 OTT Networks

With the advent of IPTV technologies and the growing Internet bandwidth access, a set of new technologies of content delivery emerged, OTT multimedia networks, defined by the unmanaged nature of the delivery. Contrary to the IPTV networks, which is managed (closed) to guarantee QoS metrics, OTT networks delivery works without the interference of an ISP, in a best-effort style on the last mile. While IPTV operators build their dedicated networks needing client-side specific hardware (a multicast-enabled device like a STB), OTT operators use any infrastructure available for free, delivering content in a multi-platform approach, where the client can use a number of devices like smartphones, tablets, smart TVs, explaining the exponential growth and popularity of OTT type delivery [25] [30] [9]. However, without controlling the network, it becomes harder to guarantee user perceived QoE resulting in various challenges that OTT operators must overcome. The components that make an OTT network must then be scalable, reliable and adaptable in real-time to account for client environment changes and others; they must provide low buffering times avoiding video halts, adequate video-resolution regarding the client's device, and small delay if the content is live. Consequently, OTT services pose at the same time a menace and an opportunity to IPTV operators: a menace because OTT services can reach more clients due to the multi-platform approach and are more adaptable to the client's preferences and habits, eating up the IPTV market; however, at the same time, it can be an opportunity for IPTV operators to take advantage of the network's control and transition fully or partially to a OTT content delivery style or offer a separated OTT service. Also, offering enhanced access to major OTT actors, by signing Service Level Agreement (SLA) to work together, both companies can benefit, for instance, Netflix establishes protocols to install hardware on the ISP networks, improving their service. Figure 2.9 depicts the problem of the unmanaged network and the appropriate place to deploy OTT hardware on a ISP network (eyeball network/access network).



**Figure 2.9:** The unmanaged network problem, at the right-side the ISP network [31]

In the past, IPTV brought a new interacting TV experience to the client, but with the arrival of OTT media content delivery, the client has a more active role on watching Television, on-demand content and the real promise of content anywhere-anytime.

## 2.3 Multimedia Services

The multimedia content delivery industry evolved in such a way that today clients have a multitude of different ways to consume Television and on-demand content. As we have seen before, Telcos have pushed major technology innovations to dominate the Pay-TV market against CATV. However, with globalization, technological giants like Google (YouTube), Facebook, and Netflix, are pushing OTT services aggressively, disturbing the market with great competition and putting stress on ISP networks, due to unicast increase traffic. Unicast traffic demands a dedicated connection from the server to the client, one-to-one transmission. Though, Telcos still play a significant part in the content delivery realm because they own the infrastructure that connects clients to the Internet and are in the perfect position to shape OTT services, pushing their OTT solutions. Nowadays, Telcos offer OTT services complementing their Pay-TV offer, giving clients the opportunity to watch IPTV channels in different screens and locations. Next, on this section, the most significant types of watching Television and on-demand content are described.

### Linear TV

The oldest and still the most common way to watch Television, Linear TV refers to regular TV broadcasting, and it was renamed to Linear TV to differ from the new ways of TV content distribution. This type of TV follows a programming schedule, broadcasting sequentially, without the user interaction. For many decades, it was the only means of watching TV.

### Video-on-Demand

The need to capitalize on selling special content gave way to VoD, where the client pays for a usually non-broadcasted content:

- Transaction VoD, Transaction VoD (T-VoD), was the most common and similar to the old video rentals stores, where the customer pays an amount to watch the desired content with limited time, watching as many times as wanted in that period.
- Electronic Sell-Through VoD, Electronic Sell-Through VoD (EST-VoD), similar to the previously described but instead of renting, the customer makes a one-time purchase and owns a digital copy of the requested content, much like Apple iTunes or Google Play.
- Subscription VoD, Subscription VoD (S-VoD), is the most common and used by OTT companies like Netflix, Hulu and Amazon Instant Video. The client pays a monthly bill to access all of the video content catalog, watching whenever, wherever the client wants. Many IPTV operators are using as well this business model for their OTT services.

### Time-shift TV

Time-shift TV is Linear-TV with client's interaction. The client can watch content that has already been broadcasted, from seconds to weeks, having several approaches:

- *Personal Video Recording (PVR)* is a STB feature that some possess, that records a TV program ordered by the user. Here the user has a proactive role in choosing and scheduling which programs to record;
- *Pause TV* is mainly a STB feature, where the client can stop the broadcasting and resume whenever wanted, respecting the STB memory. From the moment that the client executes a pause command, the STB begins to save the transmission, and later, the client can skip portions, go further or backward (on the cached content) or go to linear broadcast;
- *Start-Over TV* is a feature that allows the client to restart a TV program that is already broadcasting, or in some other approaches, a program that aired one to two days before. This feature comes from the infrastructure rather than the STB;
- *Catch-up TV* builds on Start-Over TV, by offering the possibility of going back to a program aired in the last week and some other services a full month. Some TV-channels or programs can have this feature disabled due to copyright or business constraints, causing some fragmentation on the user perception of the service. Giving this type of flexibility to the client demands more infrastructure from the provider, mostly because it is transmitted in unicast.

With the ramping of features offered by Telcos, as the resource-intensive Catch-up TV described in this section, their own OTT services, and major companies distributing as well their services for free in the Telcos infrastructure adds to a significant overall increase in unicast traffic. Further, Cisco predicts that all of consumer Internet traffic generated in 2021, 82% will be video [1], showing a growing trend, putting Telcos in a delicate position. In order to avoid the *dumb pipe*<sup>1</sup> trap where they are merely an infrastructure that others monetize, Telcos must perform a thorough analysis of what technologies to develop and implement on their infrastructure, to lower the overall OPEX and Capital Expenditure (CAPEX) to be competitive, bringing at the same time new features to stay as key players on the market [32].

---

<sup>1</sup>Dumb pipe trap: no value added, just carrying bytes

## 2.4 SDN

We live in a connected world as a result of the development of the Internet, and all of the systems created that use it. Mobile communications, social networks, Internet-of-Things (IoT), and others, are all bound together with this massive infrastructure that is the Internet, inherently becoming extremely complex to manage and configure. With the increasing access to this infrastructure, dynamic management is critical to deal with faults and changes. However, this infrastructure was built on a limited resources and high-efficiency philosophy due to the hardware technology available on the early days, making networks vertically integrated, where control and data plans are bounded, making management hard. To apply high-level network configurations, each network device must be configured individually. Making things worst, network devices from different vendors have different systems, thereby different configuration commands, creating a management nightmare. IP networks by nature do not have mechanisms of autoconfiguration to respond dynamically to faults and loads [33]. All of these shortcomings reduce network flexibility and make innovation hard, stagnating network infrastructure development. For instance, a protocol transition like IPv4 to IPv6 began more than a decade, and it is still incomplete proving the inertia of existing IP networks. Consequently, on traditional IP networks, the only way to introduce new features is through specialized, costly and complex to manage hardware called middleboxes (load balancers, firewalls, and others). In the wake of these challenges, Software-Defined Networking (SDN) is an emerging proposal to separate the control plane from the data plane, allowing network programming and centralized control of those network applications. The core principle is to have an SDN controller that has a global view of the network, configures forwarding devices, SDN switches, with the policies desired which translate to packet forwarding rules known as flow tables, affecting the traversed packets on that forwarding device. The policies applied can, in fact, transform a SDN switch into a Network Address Translation (NAT), firewall, router and so on. Splitting the data and control plane creates enough abstraction to build the foundation for real network infrastructure innovation [25].

### 2.4.1 Defining SDN

OpenFlow standard, originally developed at Stanford University, California [34], due to the researcher's frustration on network equipment with closed software that would difficult the testing of experimental protocols, gave way to the idea of *software-defined networking*. However, it was not the first attempt of splitting the data plane from the control plane: AT&T created technology in the 1990s that separated the two planes, the Network Control Points (NCPs), to improve the administration and control of its telephone network. Other initiatives, for instance, Tempest (Asynchronous Transfer Mode (ATM)) or ForCES (Ethernet) [33], tried to separate the two planes to improve management.

The concept of a Network Operating System (NOS) is not new; the most widely deployed is the famous Cisco IOS that date back to the 1990s; although with the SDN concept, it breaded a new life into the category, generating NOS like NOX (the first SDN controller),

ONOS, RYU, and others [35]. The software-defined approach also brought benefits to the network virtualization realm and network slicing, key technologies for future 5G networks.

Such a broad concept can have sometimes different meanings concerning architecture. Therefore, it is essential to establish the core components of an SDN:

- Data and control planes are split. The network devices are merely forwarding devices that follow the control management;
- That control is based on flows rather than destinations. A flow is a stream of packets between the source and the destination. A flow table is created by the controller that matches incoming packets with criteria, applying a set of actions on the packets regarding that flow entry;
- The control is an external entity that can be called either SDN controller or NOS, and it is the platform that provides software abstraction to deploy network applications that produces flows on the packet forwarding devices;
- The network is programmable, giving way to network applications that connect with the data plane.

Allowing the control separation from the data plane brought numerous advantages: an external control provides a higher abstraction from the low-level configuration languages, reducing the number of misconfigurations; the control can automatically acknowledge changes in the network, maintaining high-level procedures in place; thanks to the global network view, more complex network applications can be developed. Moreover, forward abstraction hides the low-level details of the network devices, being OpenFlow one of the first to provide that type of abstraction, working like a device driver. Figure 2.10 shows a layered view of an SDN networking functionality.

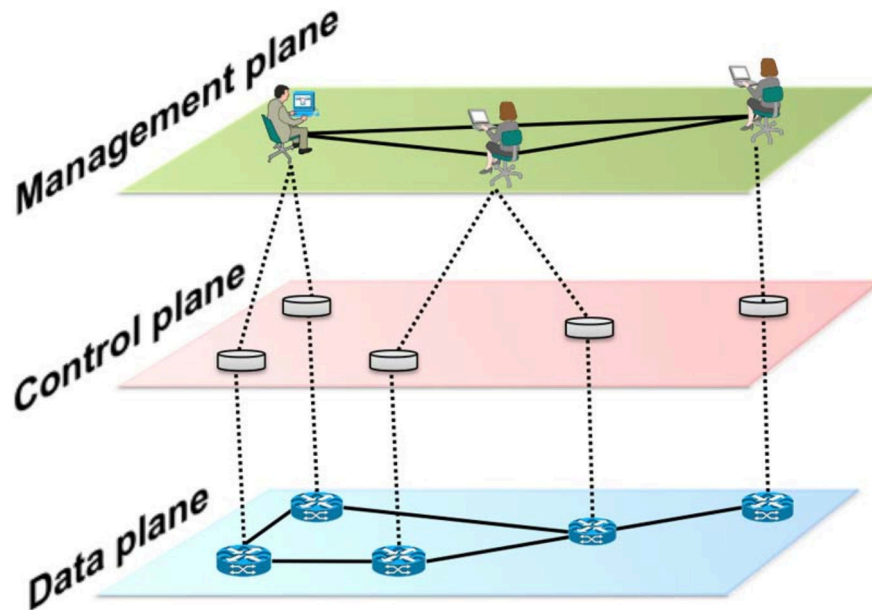


Figure 2.10: Layered view of an SDN [33]

### 2.4.2 Terminology

This subsection provides the most common nomenclature used in Software-Defined Networks:

- Forwarding Devices (FD) can be hardware or software-based networking devices that perform a set of instructions. The FD performs operations based on the Flow Table that is created by the control plane with the help of southbound protocols, through the southbound interface. For instance, a flow has matches and actions like dropping packets, modify headers, send to the controller to perform more elaborate operations and others;
- Data Plane (DP) refers to all of the Forwarding Devices interconnected by wired cables or wireless, essentially the lower network infrastructure;
- Southbound Interface (SBI) enables communication between the data plane and the control plane through a well-defined Application Programming Interface (API). The communication is done by the southbound protocol, commonly OpenFlow that has become the *de facto* standard [33];
- Control Plane (CP) is all of the elements that make an SDN controller or a NOS. This component connects the management/application layer with the data layer, using the northbound interface and the southbound interface;
- Northbound Interface (NBI) provides another level of abstraction, offering an API that abstracts the network low-level instructions (used by the SBI) and provide means to network control and management. Still, there is not any standard on the NBI;
- Application Layer (APL) or management plane is at the highest level on the SDN architecture. It is where applications are deployed to exploit the services provided by the CP, that will translate to actions on the DP. They can be applications related to routing, load balancing, monitoring, firewall and so on.

Figure 2.11 compares the traditional network layer structure versus the SDN structure.

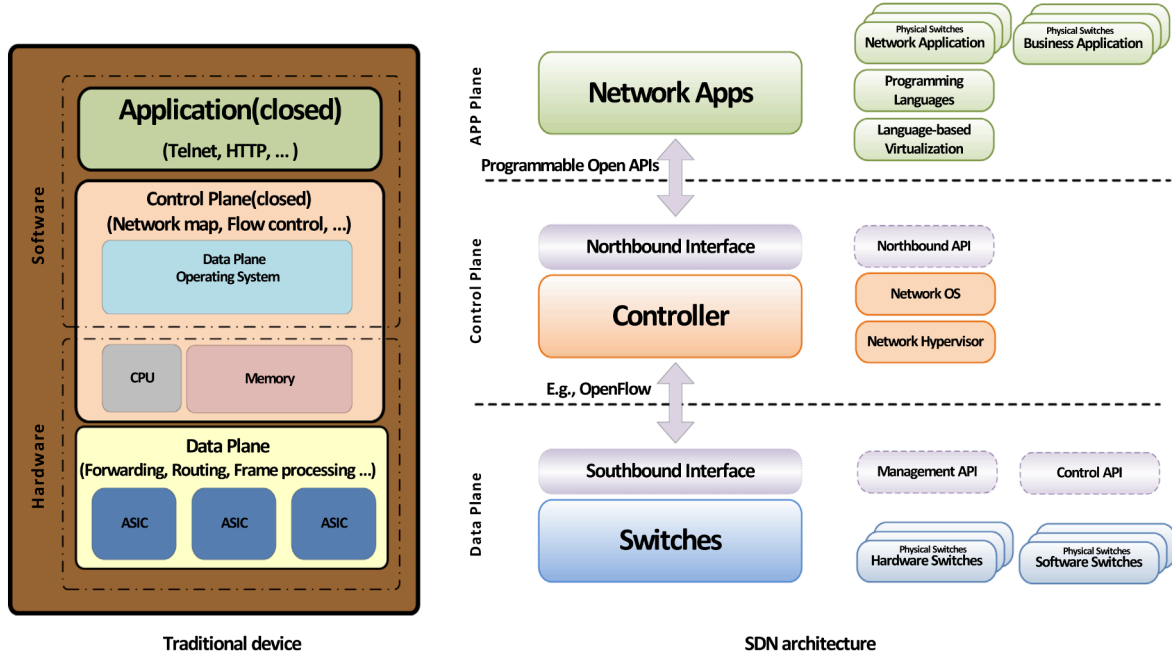


Figure 2.11: Traditional structure versus SDN structure [35]

### 2.4.3 SDN Architecture

The keystone of the SDN architecture is the abstraction, for example, the abstraction from forwarding. The developer or researcher needs to know which forwarding features are available at the NOS to apply the desired forwarding behavior without having to implement those low-level actions or worrying if the switch supports that feature, difficulties that happen on vendor-specific hardware. This section presents the core architectural design of an SDN.

#### Data Plane

The core of the DP is packet forwarding, having network devices without embedded control that can make decisions on its own like traditional network devices. The programmability of the network allows a more flexible and transparent way to process packets. For instance, a new protocol deployment depends on software rather than vendor-specific hardware. That happens due to the abstraction of network operations and centralized control. Network devices only have to support a southbound protocol like OpenFlow, to make possible the communication between the CP and the network device. Thus, OpenFlow is a protocol to implement low-level network actions, and to also provide management control over the network devices or, in this case, over OpenFlow enabled switches. Three main components are present at these switches: a flow table that is used to lookup a packet for a match, applying the correct action; a secure channel between the controller and the switch (Transport Layer Security (TLS) or Secure Sockets Layer (SSL)); and the OpenFlow protocol for communication. Flow entries make up the flow table, and each entry comprises six different parts [36] as figure 2.12 depicts. Several flow tables can exist, making a path for a packet defining its behavior. Packets in the perspective of the switch include the ingress port and packet headers. The lookup process

starts when a new packet enters the switch, looking for a match on the first table and so on until a match is found or a miss happens (producing a packet drop). Usually, a default rule (with low priority) is defined at the start to make the switch send all packets to the controller to produce the right flow rule for that type of packet in the future, having a high priority than the default flow rule. Different types of matching fields can be used on a flow rule, increasing or diminishing the matching granularity. On the last OpenFlow specification, v1.5.1, the mandatory number of total matching fields are 44 [37]. Therefore, a packet can suffer a number of actions: forward to one of the switch outgoing ports; drop the packet; forward it to the controller; send it to the processing pipeline to apply the desired alteration; send it to the next flow table. Also, in a flow rule, the Cookie field serve as a way to produce statistics and ease the job of deleting specific rules; the Priority allows precedence between rules that can affect the same set of packets; the Timeout is a way to introduce rule expiration by an absolute time value or an idle time value; a Counters field keeps track of packets that hit that rule [37].

Over the last specifications, OpenFlow has matured and supports now a wide range of matching fields for key protocols including IPv4/IPv6, Ethernet, Multi Protocol Label Switching (MPLS), TCP, UDP, Address Resolution Protocol (ARP), and others. A matching field is no more than bit masks combined to process a packet header correctly, and since version 1.2, that task was made more accessible through the introduction of OpenFlow Extensible Match (OXM). For instance, an OXM module enables features like deep packet inspection. The last OpenFlow version is 1.4, although the most widely supported version on vendor hardware is version 1.3.

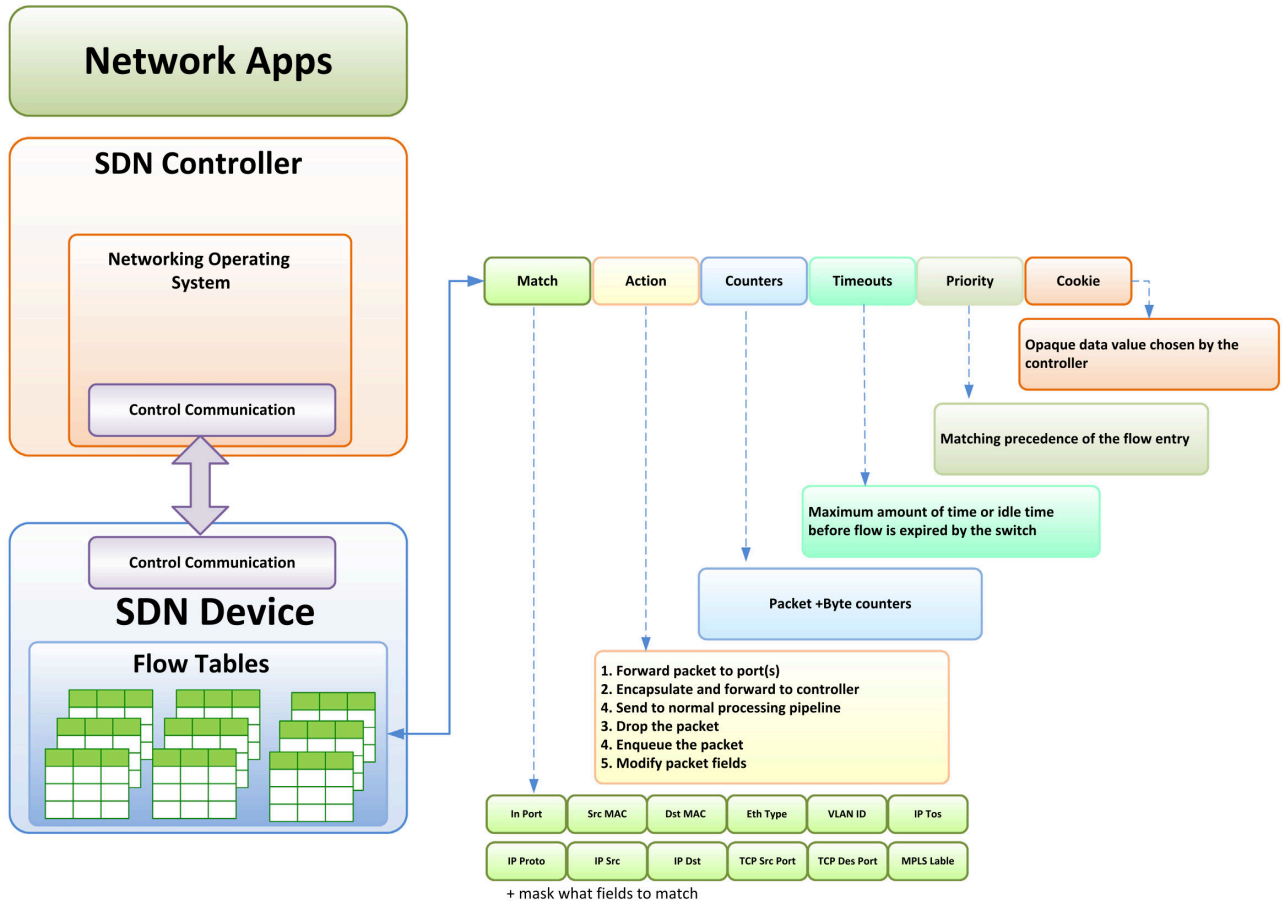


Figure 2.12: OpenFlow enable devices [35]

## Control Plane

The SDN controller or NOS is the brain of SDN. With traditional operating systems that provide abstractions for low-level devices, security mechanisms, and resource management throughout competing processes; with those features comes productivity, applications development, translating in innovation. Comparing to the network realm, operating systems for networks were inexistent or vendor-specific with complexity that comes with low-level management (Cisco IOS and JunOS) making innovation hopeless. With the SDN approach, NOS were able to offer abstractions and APIs to developers. The NOS is the foundation where applications from the APL that generate network configurations lies on. Therefore, the NOS must be designed carefully to deliver a comparable (or higher) network performance to traditional devices:

- SDN controllers can be centralized or distributed, and state consistency must be considered to have a stable forwarding policy across switches;
- With an extensive network infrastructure comes scalability issues. For effective control, scalability must be a core architectural design for SDN controllers;
- Modularity: for a distributed approach, even load distribution among controllers is essential;
- Failure recovery (links, switches) must come as a natural element for SDN controllers;

- With the decoupling of the control plane, there comes security challenges with communication on the SBI. Also, third-party applications that run on the NOS can have abusive practices that can endanger the network, and the controller design must account for these security issues.

## Application Layer

Network applications are the last piece on the SDN architecture, sitting at the top layer. Figure 2.13 puts in perspective the difference between the traditional approach versus the SDN one. These applications take advantage of all the abstraction built so far, by the controller and southbound protocol, to implement network services in a flexible and agile manner. Frequently, SDN controllers offer out-of-the-box network applications ready to deploy or at least to serve as a foundation to a specific service. One of the first uses of a network application in the SDN realm was load balancing. Whenever a new server is placed at the network infrastructure, a load balancing application can be deployed onto FD that can seamlessly activate load balancing configurations. Further, load balancing actions can precede routing actions, making way to new improved load distribution techniques. Network applications are very diverse, but almost everyone falls into one of these categories: mobility, monitoring, traffic engineering, security, and data center networking.

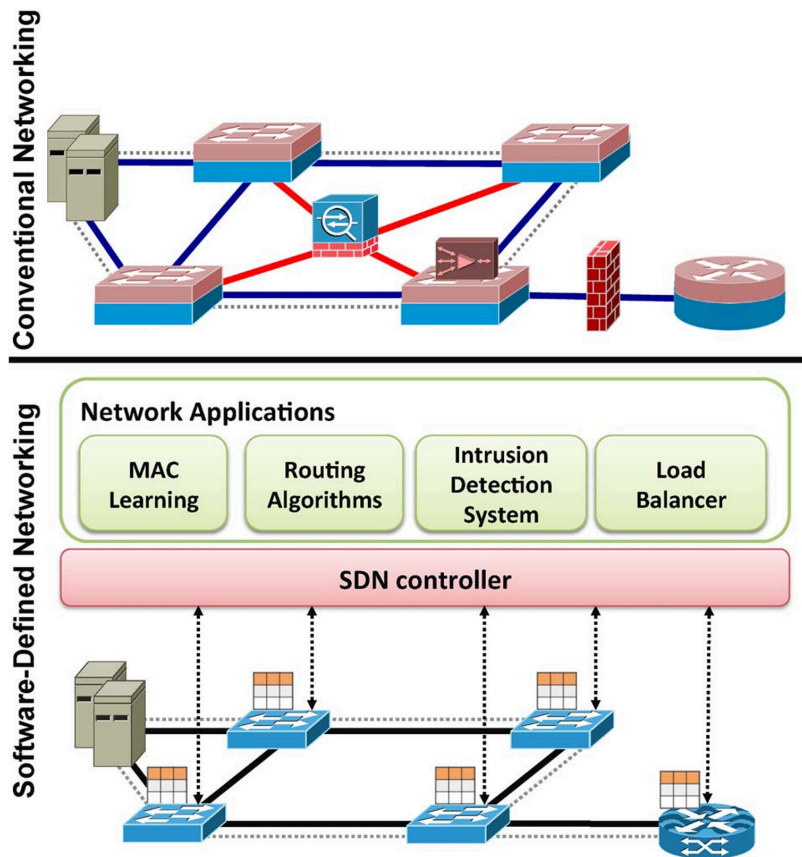


Figure 2.13: SDN versus traditional networking [33]

## Simulation

Virtualization has been one of the SDN goals, and in the way benefiting simulation efforts. One of the earliest struggles was vendor support to OpenFlow and simulation arose as a natural solution. Mininet established as the *de facto* simulator for SDN networks [38] [35]. Mininet takes advantage of OS-level Linux virtualization on both process-based virtualization and network namespaces, making it possible to create a network with hundreds of nodes in a lightweight way. A Mininet host or node has its own network and computational resources. Switches are software-based, Mininet can use Open vSwitch (OVS) or the OpenFlow reference switch. Links on the virtualized network live on the Linux kernel, that later are connected to the virtual switches providing host connectivity. The last piece is the controller that can run outside of the virtualized network, needing only to be connected to the virtual switch. The most significant advantage of working with Mininet is that any application that works on Mininet works on a real network with little adaptation, enabling faster prototyping as well as providing a learning environment on the subject.

### 2.4.4 Evaluation of SDN Controllers

In [39], the most popular SDN controllers are compared feature- and performance-wise. On the performance side, the *PACKET\_IN*, which happens when there is no match on the switch and a default rule forwards the packet to the controller, is evaluated taking in consideration two metrics:

- Latency: how fast a controller takes care of a *PACKET\_IN* event;
- Throughput: per second, how much *PACKET\_IN* messages a controller can process.

The authors use Cbench, a performance measurement tool for OpenFlow compatible controllers, like many others that made controllers performance analyses used [40] [41] [42].

On the featured based comparison, the controllers were compared using these criteria:

- SBI: OpenDayLight has the most support for southbound protocols, followed by Ryu.
- NBI: Representational State Transfer (REST) API is the most supported throughout the controllers;
- Controller efficiency: distributed schemes aim to overcome performance issues on vast scale network. ONOS and OpenDayLight are the only ones that support distributed control over the compared controllers;
- Partnership: in an open source world, having partnerships with large companies ensures continuous development and quality innovations. OpenDayLight has the most partnerships, like Cisco and IBM, followed by ONOS.

Therefore, a featured based comparison shows that OpenDayLight and ONOS are the most feature rich, and it makes sense because both aim to solve challenges on large commercial networks. Figure 2.14 shows a detailed view of features that controllers possess.

Throughout the performance comparisons, ONOS had the best results on scalability and throughput tests; Ryu followed by OpenDayLight had the best results on latency. The authors conclude that ONOS is appropriate for large network deployments, while OpenDayLight

presents a great overall choice due to the vendor support and features. Ryu is suitable for smaller networks and delay sensitive applications.

	Ryu	Floodlight	OpenDayLight	ONOS
<b>Southbound Interfaces</b>	OF 1.0, 1.2, 1.3, 1.4, NETCONF, OFCONFIG, OVSDDB	OF1.0,1.1,1.2, 1.3, 1.4,1.5	OF1.0, 1.3,1.4,1.5 NETCONF/YANG,OVSDDB, PCEP,BGP/LS, LISP, SNMP, OFCONFIG	OF1.0, 1.3,1.4, 1.5 NETCONF
<b>REST API</b>	Yes (For SB only)	Yes	Yes	Yes
<b>GUI</b>	Yes (Initial phase)	Web/ Java- based	Web-based	Web-based
<b>Modularity</b>	Medium	Medium	High	High
<b>Orchestrator Support</b>	Yes	Yes	Yes	No
<b>OS Support</b>	Most supported on Linux	Linux, Windows , and MAC	Linux, Windows , and MAC	Linux, Windows , and MAC
<b>Partner</b>	Nippo Telegraph And Telephone Corporation (NTT)	Big Switch Networks	Linux Foundation With Memberships Covering Over 40 Companies, like Cisco, IBM,	ON.LAB, Sk Telecom, Cisco, Ericsson, Fujitsu, Huawei, Intel, AT&T, Nec, Ciena, Nsf.Ntt Communication
<b>Documentation</b>	Medium	Good	Very good	Good
<b>Programming Language</b>	Python	Java	Java	Java
<b>Multi-threading support</b>	Yes	Yes	Yes	Yes
<b>TLS Support</b>	Yes	Yes	Yes	Yes
<b>Virtualization</b>	Mininet and OVS	Mininet and OVS	Mininet and OVS	Mininet and OVS
<b>Application Domain</b>	Campus	Campus	Data center and Transport-SDN WAN	Data center and Transport-SDN WAN
<b>Distributed/Centralized</b>	Centralized	Centralized	Distributed	Distributed

**Figure 2.14:** SDN Controllers feature comparison [39]

## 2.5 QoE

In the realm of engineers and technicians, they tend to care more about what new technology goes into a product, judging by its raw performance compared to the old technology. However, in the eyes of an average user, they want a product that solves a problem. The user does not care if Firefox has a new super advanced engine, they *expect* Firefox to open a webpage within an acceptable time. Different products with the same functionalities can present the user with an entirely different experience. Sometimes more technologically advanced products do not necessarily translate into success due to user experience problems. Therefore, QoE is a growing trend, especially in multimedia services where there are a lot of visual elements, to evaluate the overall performance of a product. In the past, the trend was to improve QoS presuming that would lead to QoE improvements, which to some degree is true, but QoE deals with a broader scope and with subjectivity rather than objectivity like QoS. Another way to separate QoS and QoE is that the first deals with machines and the second with humans, and humans are far more complex.

The International Telegraph Union Telecommunication Standardization Sector (ITU-T) [43] defines QoE as:

*The overall acceptability of an application or service, as perceived subjectively by the end-user.*

- *NOTE 1 – Quality of experience includes the complete end-to-end system effects (client, terminal, network, services infrastructure, etc.)*
- *NOTE 2 – Overall acceptability may be influenced by user expectations and context*

Experience is subjective, the user's background and past experiences influence the perceived quality of a product. Therefore, how can QoE be measured in order to be enhanced? Taking insight from social sciences and marketing, fields that have already studied how to quantify people's preferences [44], QoE assessment can be divided into objective and subjective. In the subjective category, surveys are performed to evaluate a determined product. The surveys' main goal is to collect users' opinions under different scenarios in a controlled environment. However, this type of assessment does not work for on-the-fly QoE estimations for a service, that is why the objective approach builds upon the subjective approach findings and tries to infer which objective values influenced the opinions of the surveys. Thus, training objective assessment algorithms with data from subjective surveys give way to QoE evaluation models.

In the multimedia services realm, quality video assessment is subjective; thus it is natural to perform several subjective studies to evaluate a service. The key metric to do so is called Mean Opinion Score (MOS), where a user rates the experience in a scale of 5, with five being the best experience. Then, objective information is added, like video resolution, frame rate, buffer stalls to connect to the user experience, building a QoE model. The DASH-IF is trying to standardize metrics to assess video QoE, because different players can calculate metrics differently, like calculating buffer stalls in a slightly different way, making a comparison between players inaccurate [45]. Studies performed [46] [47] indicate that humans are time sensitive, meaning that buffer stalls have a very negative QoE impact, though the initial delay

to buffer a new video does not have a substantial impact. Thus, video playback plays a more critical role than video resolution variation on user experience.

In conclusion, for HTTP adaptive video streaming technologies, where the user is given a great deal of control because they are in the best position to assess their environment, QoE metrics are excellent to help the user take decisions to improve the overall experience.

## **2.6 Related Work**

This section presents the related work of the two main subjects developed in this dissertation.

### **2.6.1 Bandwidth Forecast**

To Telcos it is essential to anticipate user demand, in order to develop new features that may impact their infrastructure. Nowadays, Telcos services usually divide into Live-TV and on-demand content. The first uses a multicast type of transmission, and the second one a unicast type of transmission. Comparing unicast to multicast, the latter is more complicated to manage and have hardware compatibility issues [48]. Also, the rise in unicast traffic [1] versus multicast traffic poses a question of how the bandwidth will increase due to video services like Live-TV and on-demand, and which measures must be taken infrastructure-wise. To do that, it is important to understand which type of transmission is more expensive and how future demand will behave. Annual reports on global network indicators like Cisco [1] or video consuming habits like Ericsson [2] are a vital help to predict future demand but are not sufficient. Specific data on video services bandwidth usage from a Telco is needed to have a more significant comprehension on how the infrastructure must change.

Further, it is important to assert the difference in operational costs between unicast and multicast transmission. The complexity lies in how much a multicast network costs and specifically, how a multicast tree grows and its efficiency. Several works try to determine how to price multicast routing compared to unicast [32][49][50][51][52][53]; thus, due to the complexity of the problem, none of them reaches a consensus. In conclusion, it would be interesting for a Telecommunication Operator (Telco) the conciliation of a bandwidth forecast with an infrastructure estimation cost.

### **2.6.2 SDN Load Balancing**

Load balancing is a critical job in the age of traffic overload on web servers; it can add flexibility, reduce delay and improve service availability. The coupling of load balancing with SDN, although recent, was a natural evolution due to the requirements of flexibility and others [54]. Several works have proved the benefits of these two concepts working together. The work on [55] proposes an SDN load balancing mechanism for machine-to-machine (M2M) heavy bursts of traffic based on QoS metrics. The mechanism proposed monitors the network and dynamically re-routes traffic if certain QoS criteria are met. On [56] SDN load balancing is applied in a data center network scenario. It aims to optimize link utilization in before-

mentioned networks with dynamic load management: when congestion occurs, the path is replaced by the next best (minimum link cost and lower traffic flow). Over time, the proposed method showed improvements over traditional load balancing. The work on [57] is similar to the previously discussed with a data center scenario and a re-routing method based on QoS; however, it introduced a congestion control mechanism and developed a new load balancing algorithm. The authors in [58] compare two different algorithms of load balancing that possess a bound function for client session management in a web server environment. Again, QoS metrics were used to improve the solution. The work on [59] tackles the problem when flow tables become complex, degrading the SDN load balancing approach. The proposed method divides the flow table into two: single flow table that monitors the traffic of each client, and a group flow table that groups clients in categories — the results show a higher performance in traffic scheduling. The work in [60] proposes a FlowQoS mechanism where flows are rated with QoS metrics, creating virtual queues to improve network data flows. Results also show improvement over the tested scenarios.

In conclusion, most of the SDN load balancing proposed solutions are based on QoS metrics and tailored for web-server/data center scenarios.

## 2.7 Summary

This chapter started by describing multimedia technologies, to then approach multimedia infrastructures that support such technologies. To close the multimedia content delivery subject, multimedia services described the major features available for content consumption. Moreover, SDN networks and QoE subjects were described in detail because of the influence in future multimedia infrastructure and this work. Lastly, related work section contextualizes further the two subjects developed throughout this work.



# Forecasting OTT Bandwidth Consumption

*This chapter discusses the dilemma of multicast transmissions versus unicast transmissions. For internal ISP delivery, native IP-Multicast is the most efficient way to deliver video in a group-like manner but poses challenges such as additional configurations, security issues, and hardware compatibility along the whole network path. Unicast transmissions, on the other hand, do not scale well for large requests of video content, but are the most straightforward way of delivering Over-the-Top (OTT) content. This chapter proposes a forecast model for unicast and multicast transmissions.*

## 3.1 The Proposed Forecasting Model

The forecasting model focuses on Live-TV and On-Demand services. These two must have different approaches due to their different characteristics. The On-Demand category contains Video-on-Demand (VoD) and Catch-up TV services. It is possible to cache these contents, having only a slice of the total catalog on Service Router (SR) appliances to have a significant decrease in Core Router (CR) requests. For Live-TV, it is not possible to cache, but it is possible to add a small delay by storing the streams on a buffer. With this approach, the network stress would pass onto the appliance, saving critical SR uplink bandwidth. The primary goal of this work is to achieve a forecasting model that can provide useful insight over several options to evaluate feasible migrations from multicast IPTV. First, for the model foundation, a set of critical variables are chosen:

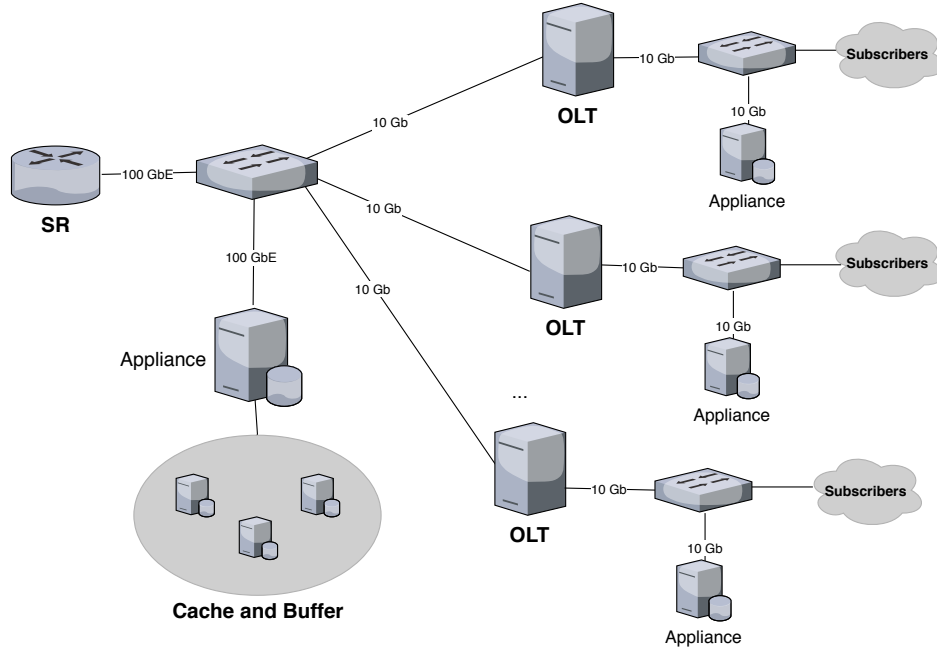
- number of subscribers;
- peak streams;
- average and peak bandwidth Internet usage;
- likelihood of a subscriber watching a stream;
- client and Headend channel mix percent;

- cache appliances hit ratio;
- average on-demand bandwidth;
- Optical Line Termination (OLT) maximum subscribers;
- percentage of Live-TV and On-demand services;
- Baseline/Unicast cost ratio.

Most of these variables are self-explanatory; however, some need further explanation. Peak streams, an essential parameter, represents the maximum request ever made at the same point in time, making it a more meaningful metric than STBs per subscriber. The client's channels mix is the percentage of each type of TV channels that subscribers watch comprising Standard Definition (Standard Definition (SD)), High Definition (High Definition (HD)) and Ultra High Definition (4K) (ultra high definition). On the other hand, the Headend channels mix percent represent what kind of TV channels are available for the subscriber. The Baseline\Unicast cost ratio represents how much more expensive is a multicast path compared to a unicast path.

### 3.1.1 Multicast vs Unicast Model Envisioned Architecture

The content delivery infrastructure on this work is envisioned to have four levels: subscriber, OLT, SR and Headend. The subscriber can have one or more STB; the OLT is the point of access to the network that aggregates small groups of clients; the SR aggregates the OLTs, providing all ISP services; the Headend is responsible for broadcasting TV channels. The appliances, on the suggested architecture, consist of specialized hardware for cache and buffer possessing switching capabilities. The CRs are not included in the infrastructure because the total SRs' uplink bandwidth is the same bandwidth that one or more CRs must provide for downlink bandwidth. The logical places for the appliances would be at the SRs and the OLTs. From a theoretical perspective, the more distributed (cache and buffer) is the solution, the better. However, due to the technological challenges of installing appliances at the OLTs on the current architecture, and also the cost of having such a distributed approach for an ISP, the only place considered to install them is at the SRs. Figure 3.1 presents the envisioned architecture with appliances on the SRs.



**Figure 3.1:** Vision for cache and buffer implementation

### 3.1.2 Approaches

The services provided by the envisioned infrastructure are Live-TV, Catch-Up TV, and VoD. The baseline infrastructure considered, as numerous IPTV operators run, uses multicast for Live-TV, and unicast for On-Demand (Catch-Up TV and VoD). Therefore, regarding On-Demand services, it is possible to cache that content having only a slice of the total catalog on cache appliances distributed throughout the network. For Live-TV services, it is not plausible to cache it, but it is viable to use a buffer even if it adds a small delay. With this approach, part of the network stress would pass to the appliances, saving significant bandwidth resources and improving response times. Several methods were considered to create a model to forecast bandwidth demand and cost. The most obvious path was to send everything in unicast to establish a raw baseline comparison with the ISP actual infrastructure, and then implement appliances to mitigate bandwidth spending. Another approach, besides caching and buffering, is to leverage the multicast Live-TV efficiency on the most requested TV channels, leaving the remaining in unicast transmission for Live-TV. Therefore, beyond the current multicast delivery baseline, two approaches emerge:

- **Unicast only:** to simplify the whole delivery infrastructure, every transmission, Live-TV, and On-Demand, is performed in unicast from the Headend to the subscriber. With this approach, the stream delivered is the same for both STB (Set-top Box) and mobile terminals, reducing operational costs compared to multicast. However, it is of the utmost importance to add appliances to make the delivery infrastructure efficient and conceivable, splitting the approach into unicast and unicast with appliances.
- **Hybrid:** in some scenarios, multicast channels can be more expensive than unicast channels when the viewers are small-scale for such TV channels. Since reliable data

about the most requested channels are scarce, the Pareto Principle is a valid option to estimate them [61]. The fundamental idea is to use multicast on the most demanded TV channels and the remaining in unicast. The hybrid method takes the most of multicast Live-TV while diminishing entropy due to multicast channels scalability problems. The Pareto Principle applies as follows: 20% of the TV channels available are responsible for 80% of the subscriber's requests, making multicast the ideal method for delivering them. Consequently, the remaining 80% of the TV channel's requests, are transmitted in unicast, which are responsible for 20% of the channel's demand. This approach also divides into hybrid without and with appliances, the latter adding On-Demand cache.

In the end, the challenge was to find a way to represent the appliances bandwidth gain, the overall benefit for the Hybrid model, and above all, to estimate an OPEX for multicast and the other approaches. Although spending less bandwidth is a good indicator, the OPEX estimation unifies all data into a cost-like estimate. These two approaches are alternatives for the current implementation: Live-TV in Multicast and the other services in Unicast (On-Demand). In this work, the current implementation will have the name Baseline. It is important to mention that the Baseline calculations follow Altice Labs specification, like average bandwidth spent at the OLT or how much requests are from Live-TV compared to other services.

### 3.1.3 Implementation

Reliable data is the cornerstone for a prediction. In this work, the values of the critical variables aforementioned, all except the cache appliances hit ratio from [62], comes from Altice Labs. The values used on the scenarios built, like average Internet usage or which types of channels are more requested (SD, HD, 4K), were of the utmost importance to forecast bandwidth requirements as well as the OPEX cost-metric. Figure 3.2 displays a table from Altice Labs with the values for Baseline on the Headend and the SR.

Headend Multicast		Service Router Multicast	
<b>Downlink</b>		<b>Uplink</b>	
Number of TV Channels	195	Uplink Mcast	610 Mbps
Average Channel Bitrate	3.3 Mbps	Uplink Unicast IPTV	7500 Mbps
Total Channels' Bitrate	610 Mbps	Uplink Unicast Internet	8320 Mbps
Number of Service Routers	60	<b>Total Uplink Bandwidth</b>	<b>16430 Mbps</b>
<b>Total Downlink Bandwidth</b>	<b>610 Mbps</b>	<b>Aggregated Downlink</b>	
		Number of OLTs	10
		Downlink Mcast	4508 Mbps
		Downlink Unicast IPTV	7500 Mbps
		Downlink Unicast Internet	8320 Mbps
		<b>Total Downlink Bandwidth</b>	<b>20328 Mbps</b>

**Figure 3.2:** Altice Headend and SR values

As a result of the trouble of quantifying how much more expensive Baseline is than unicast, the Baseline\Unicast ratio parameter allows for an adjustable value, with a theoretical default value of 1.3. Linear regression was made from data on [62] to predict the cache size needed

to achieve a desired hit ratio. With this data, it is possible to calculate CAPEX for cache storage, where the price of the gigabyte can be established on the model. The buffer for Live-TV was considered to have 30 seconds of each channel stream, adding the size needed to the CAPEX for buffer storage calculation. The model also allows CAPEX calculation for appliances that will be at the SR or at the OLT. It takes uplink, price and then calculates how much units would be necessary to match the total Service Router uplink bandwidth calculated on Unicast-only or Hybrid. Still, the most challenging part of the model was the estimation of multicast operational costs. One of the fundamental aspects of quantifying how much multicast transmissions costs is the multicast power law [49]. One of the first researches of its type, Chuang and Sirbu [49], propose a cost-based approach to determine multicast pricing with unicast traffic as a guideline for comparison. In this work, multicast packets were assumed to traverse along a shortest-path tree from source to multiple destinations. The shortest-path tree applies Dijkstra's algorithm to simulate a multicast tree, and the cost metric is hop-based. Tree saturation effect shows that the cost of a multicast tree grows at a 0.8 power regarding the multicast group size. When it is saturated or fully grown, additional members can enter the multicast group with virtually no added cost. With an estimated value for the cost of a multicast tree inferred from the multicast group size, the research introduces an equation for multicast price over unicast. The authors suggest an increasing price until the saturation happens, applying a price ceiling when the multicast tree becomes fully grown. The authors of [50] go in-depth on [49] to investigate the power law further, which had been adopted by other works [32] [61] [53]. The study encounters difficulty to find validity from observing graphs empirically, as done by the original work, and proves that for some scenarios the power law does not hold, and for some other scenarios it holds. However, the scope of that work was about giving a more accurate method to measure multicast trees that would work for numerous scenarios, not to develop a new way to calculate multicast tree costs. The study conducted in [51] reminds that cost pricing for multicast was extensively studied by multidisciplinary teams with Economists, Computer Scientists, and Mathematicians. The work's motivation lies in reviewing issues on cost allocation algorithms for shared multicast networks among ISPs. Without an explicit standard pricing scheme, business agreements can be troublesome. The authors had to slightly modify the problem, assuming some parameters due to the very challenging problem of finding an optimal multicast tree, presenting a solution for that modified multicast approach.

### 3.1.4 OPEX Estimation

Considering the research work evaluated, it is clear the significant challenges of pricing multicast communications and the high costs of operating such networks compared to unicast. In general, profit models for shared multicast networks [32] and other models developed follow the multicast power law by [49]. For a generic and standalone multicast network, the power law is suitable, and the proposed model adopts a more straightforward power law, considering that the multicast tree is always growing at  $n^{0.8}$ , where  $n$  is the average number of links on a unicast path (sender to receiver). Thus, in this work, OPEX is quantified regarding

network resources needed to attend the estimated delivery approach demand, according to the law mentioned above, meaning that it is not calculated taking into account how much money is necessary to operate particular network delivery approaches, since these data are entirely confidential to each network operator. However, since a resource cost-oriented value is provided, each operator can estimate its own actual money OPEX value. For unicast-only without appliances, OPEX is calculated according to Eq. 3.1 and corresponds to the sum of peak stream plus the total bandwidth divided by 4, to reflect the lower weight on the final cost, where  $p$  is the peak streams and  $b$  is the total unicast bandwidth.

$$C_1 = p + \frac{b}{4} \quad (3.1)$$

OPEX for Unicast-only with appliances is calculated according to Eq. 3.2, where  $s$  is the required number of SRs,  $o$  is the required number of OLTs, and  $m$  is the maximum number of subscribers per OLT. It reflects the number of paths reduction between SR and OLT.

$$C_2 = s + \frac{(o \times m) + b}{4} \quad (3.2)$$

Paths between OLTs and subscribers are divided by 4 to account for the lower cost, since core network resources are more expensive compared to the network edges.

Eq. 3.3 shows how OPEX is calculated for the multicast delivery. There,  $r$  is the multicast cost ratio and  $p$  represents the number of peak streams. The formula tries to quantify how a multicast tree grows regarding the number of subscribers and the bandwidth cost compared to the unicast.

$$C_3 = p^{0.8} \times r + \frac{b}{4} \quad (3.3)$$

Eq. 3.4 shows the cost for the hybrid forecasting model. It follows the Pareto principle, where 20% of costs comes from multicast transmissions and 80% from unicast transmissions. The multicast cost ratio is divided by 2 to reflect the entropy decrease for TV channels reduction, where variables  $p$  and  $r$  have the same meaning as in aforementioned cost equations,  $b$  is the multicast bandwidth, and  $u$  is the unicast bandwidth.

$$C_4 = (p^{0.8} \times \frac{r}{2} + \frac{b}{4}) \times 0.2 + u \times 0.8 \quad (3.4)$$

These equations estimate OPEX for the three main delivery approaches compared, using the forecasted bandwidth requirements. The forecasting framework developed was built in R<sup>1</sup> using RStudio<sup>2</sup>, and the user interface was developed with Shiny, a web application framework for R.

### 3.1.5 Platform

This subsection describes the platform of the proposed forecasting model in a use case style. There are three types of forecast provided: 2017 Forecast, 2021 Forecast, and Custom as figure

<sup>1</sup>R, Available: <https://www.r-project.org/>

<sup>2</sup>RStudio Inc. and Shiny, Available: <https://www.rstudio.com/>

3.3 illustrates. In each type, different options may be available. The total subscribers, hit ratio for both SR and OLT and Baseline/Unicast are options that are always on display, to give a glimpse of which default values are being used. The first is self-explanatory, the second is the ratio expected for the cache appliances that will affect bandwidth values and OPEX values for Baseline, Unicast-only, and Hybrid. The third parameter indicates how much more expensive multicast Live-TV (Baseline) is, comparing with unicast TV (unicast-only and hybrid). The three checkboxes allow other options to be displayed. The most straightforward use case is to forecast for 2017. The user opens the web page, clicks on the "Apply" button and the result data is shown. Every requested forecast considers the three forecasts done simultaneously: Baseline, Unicast-only and Hybrid. Some parameters do not affect the Baseline like the hit ratio on the SRs and OLTs.

**Input Data**

**Type of Forecast:**  
Forecast 2017 ▼

**Total Subscribers**  
1200000

**Hit Ratio Appliance**  
0.7

**Today(as is)/Unicast ratio**  
1.3

☐ Growth options  
☐ More options  
☐ Opex curve

**Apply**

**Figure 3.3:** Initial Platform Menu

If a user wants to activate the growth checkbox, it must do it only on the "Forecast 2017", on the other types of forecast it is not allowed. Therefore, the user must switch to "Forecast 2017" if necessary, before clicking on "Growth Options". Then, new options will become available as in figure 3.4. For every one of the sliders (input values), it will impact the values year by year until 2021. If subscribers grow 20%, they grow 20% each year until 2021. In the end, the user must click on the "Apply" button to forecast with new values.



**Figure 3.4:** Extended options for growth functionality

If a user wants to control how much throughput an SR has or to tweak the appliances for CAPEX (experimental) estimates, the user must click on the "More options" checkbox, like in figure 3.5, modify the parameters for the specified scenario and hit "Apply." Here the user can set the bandwidth surplus, the maximum uplink for the SR or for the OLT (considered to be symmetrical) and all the values needed to compute the CAPEX for appliances.

**Figure 3.5:** Extended options for more flexibility

The last use case is the Custom forecast. The user wants to build a forecast that meets specific characteristics. For that, the user must select the "Custom" on the "Type of Forecast." Then, new options will surface as illustrated in figure 3.6. Limiting the number of subscribers that an OLT support is possible. Also, the mix channels percentage is divided into Headend

and client consumption. The Headend is the percentage of channels available in the catalog to the subscribers, and the client percentage is the mix percentage of which channels subscribers watch. The "TCP usability" parameter represents how much of the link is usable on a TCP connection, affecting the Unicast-only and the Hybrid transmissions (both SR and OLT are affected). The custom forecast has default values that come from the 2021 scenario. This way, it is easier to build upon the 2021 scenario or to correct some parameters for the 2021 forecast. The user can also set new values for average Internet bandwidth and on-demand average bandwidth. Lastly, four new parameters are introduced. For a company that has its OPEX values, they can insert them here. The cost for a Live-TV multicast team and how much that team can handle regarding streams/subscribers is used to calculate the overall population of subscribers. The same happens for the unicast team, the cost and how much they can handle. Then, when the user hits the "Apply" button, the new parameters will be used to calculate the custom scenario.

<b>Max subscribers per OLT</b> <input type="text" value="2000"/>	<b>Custom average internet</b> <input type="text" value="0.832"/>
<b>SD percentage headend</b> <input type="text" value="0.4"/>	<b>Custom average iptv</b> <input type="text" value="0.75"/>
<b>SD percentage client</b> <input type="text" value="0.55"/>	<b>Custom peak streams</b> <input type="text" value="900000"/>
<b>HD percentage headend</b> <input type="text" value="0.5"/>	<b>Team cost multicast</b> <input type="text" value="100"/>
<b>HD percentage client</b> <input type="text" value="0.4"/>	<b>Team max (boxes/streams) multicast</b> <input type="text" value="500"/>
<b>4K percentage headend</b> <input type="text" value="0.1"/>	<b>Team cost unicast</b> <input type="text" value="50"/>
<b>4K percentage client</b> <input type="text" value="0.05"/>	<b>Team max (boxes/streams) unicast</b> <input type="text" value="550"/>
<b>TCP usability</b> <input type="text" value="0.6"/>	<input type="button" value="Apply"/>

**Figure 3.6:** Extended options for Custom forecast

All of the parameters available on the forecast model platform are described in the Appendix 7.

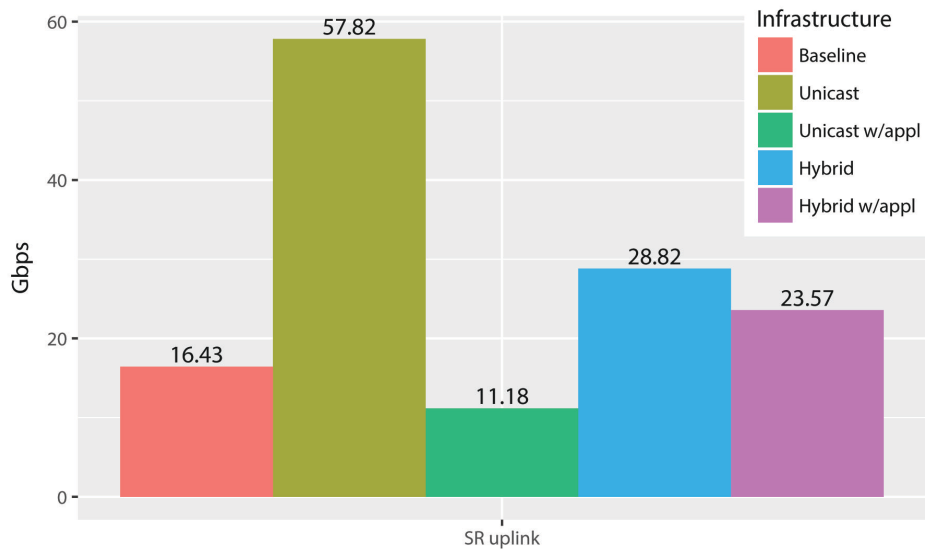
## 3.2 Experiments

One of the most significant concerns of a unicast stream, besides the bandwidth spending, is the delay. In some cases, like sports events, the increased delay is less tolerable by the subscribers. On the other hand, if every channel/stream is on unicast, no additional effort on the server side is needed to watch a stream on a mobile terminal or an STB. On the Hybrid model, besides the most popular channels, all the sports events can be on multicast, avoiding the natural increase of unicast delay. Additionally, the 4K channels can also be added to the multicast group for bandwidth saving, which needs six times more bandwidth than an SD channel.

To evaluate the above-mentioned forecasting approaches, the proposed model processes data from two scenarios. To accomplish that, the scenarios' data comes from Altice Labs. They comprise a Portuguese operator and a North American operator which portrait subscriber's behaviour as well as key infrastructure data. The values used on these scenarios, like average Internet usage or which types of channels are more requested (SD, HD, 4K), were of the utmost importance to forecast bandwidth requirements as well as CAPEX and OPEX. The forecast is divided into three categories: Forecast for 2017, Forecast for 2020 and Custom forecast. With the latter, it is possible to adjust the values for another country. On each category, a forecast for Baseline, Unicast-only, and Hybrid are made. The results presented include appliances at the SR and at the OLT, to show the best outcome possible.

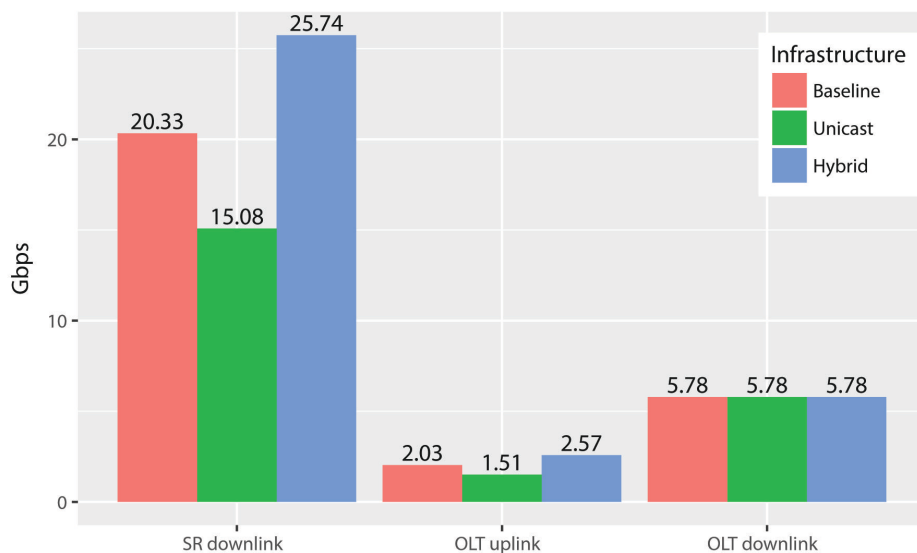
### 3.2.1 Portuguese Scenario

Starting with the Portuguese scenario for 2017, Figure 3.7 reveals the forecasted SR uplink bandwidth between baseline (multicast Live-TV and On-Demand unicast), unicast without, and with appliances and Hybrid without and with appliances. The appliances on the Hybrid approach are only for cache, for On-Demand services, presenting the gain of the hit ratio achieved on such services. Plain Unicast needs 57 Gbps on each SR. However, when appliances are in place, it becomes better than the Baseline configuration, which uses multicast for Live-TV. With cache appliances, the Hybrid approach still spends more bandwidth than the other two (Unicast-only and Baseline). Particularly noteworthy, the Hybrid method needs  $\approx 43\%$  more bandwidth than the Baseline.



**Figure 3.7:** PT SR bandwidth uplink forecast 2017

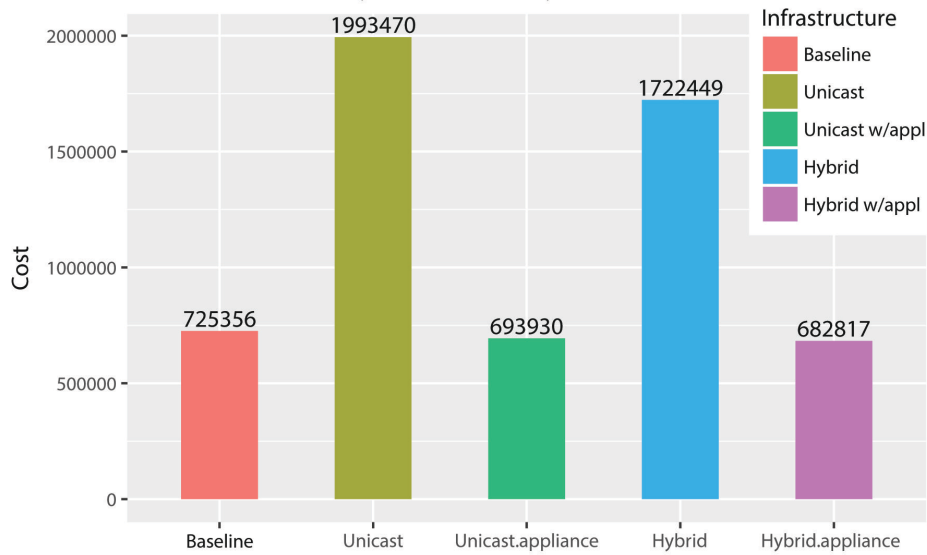
Figure 3.8 depicts the remaining 2017 bandwidth consumption forecast comprising the other network elements on the assumed architecture including SR downlink and OLTs uplink and downlink. This forecast only compares Unicast-only and Hybrid with appliances, and shows that Unicast-only with appliances still spends less bandwidth. The number of OLTs and SRs are the same for both methods. The OLT downlink is certainly the same because the subscribers use the same services. At the OLT, Unicast with appliances spends  $\approx 34\%$  less than the Baseline.



**Figure 3.8:** PT SR and OLT forecast 2017

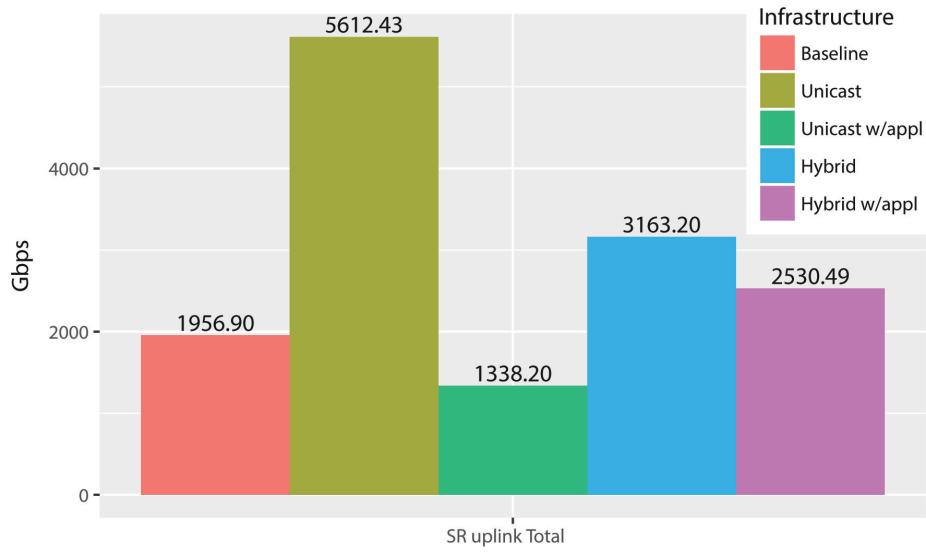
Although spending less bandwidth is a good indicator, OPEX is the one that unifies all

data into a cost estimate. Figure 3.9 shows a close comparison between the Baseline and the other two approaches. For this scenario, the Hybrid approach with appliances at the SR and OLT has a lower OPEX, that is  $\approx 6\%$  of improvement compared to the Baseline. Again, a multicast path is considered to cost 3 times more than a unicast path; it is a theoretical value and could be higher.



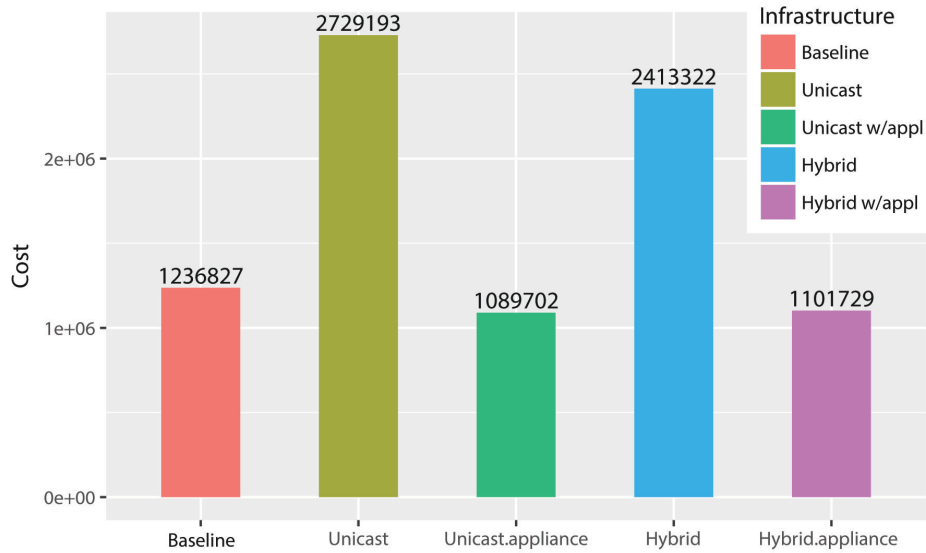
**Figure 3.9:** PT OPEX forecast 2017

For the Portuguese 2020 scenario, the Unicast-only and the Hybrid need 10% more infrastructure (SR and OLT) than the baseline approach. Therefore, to get a correct overview, Figure 3.10 shows the total SR uplink bandwidth needed. Once more, Unicast-only with appliances shows less bandwidth consumption, beating the Baseline and Hybrid approaches.



**Figure 3.10:** PT Total SR uplink bandwidth 2020

With the increase of unicast traffic, especially of On-Demand content, and no increase in TV channels, Unicast-only surpasses the Hybrid approach and becomes the one less expensive, deepening the gap between the Baseline cost estimation and unicast delivery, as Figure 3.11 shows.

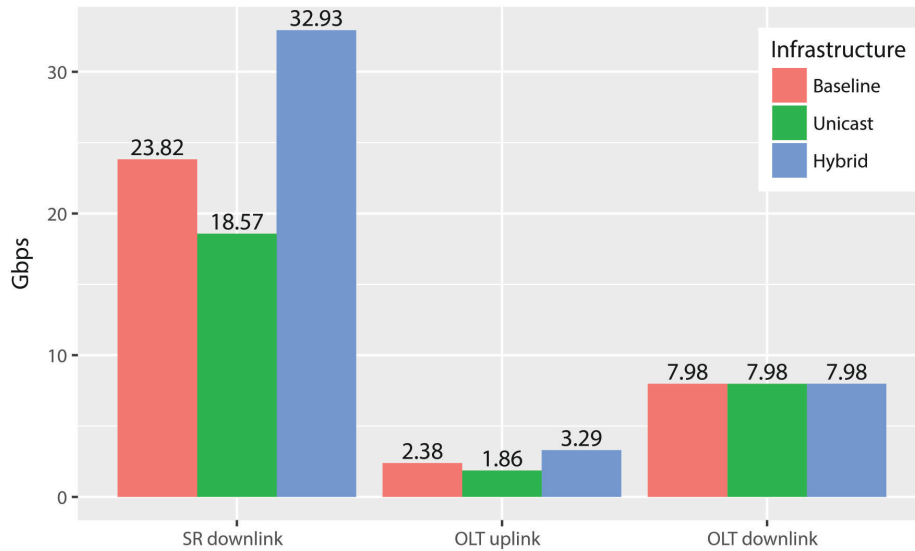


**Figure 3.11:** PT OPEX forecast 2020

### 3.2.2 North American Scenario

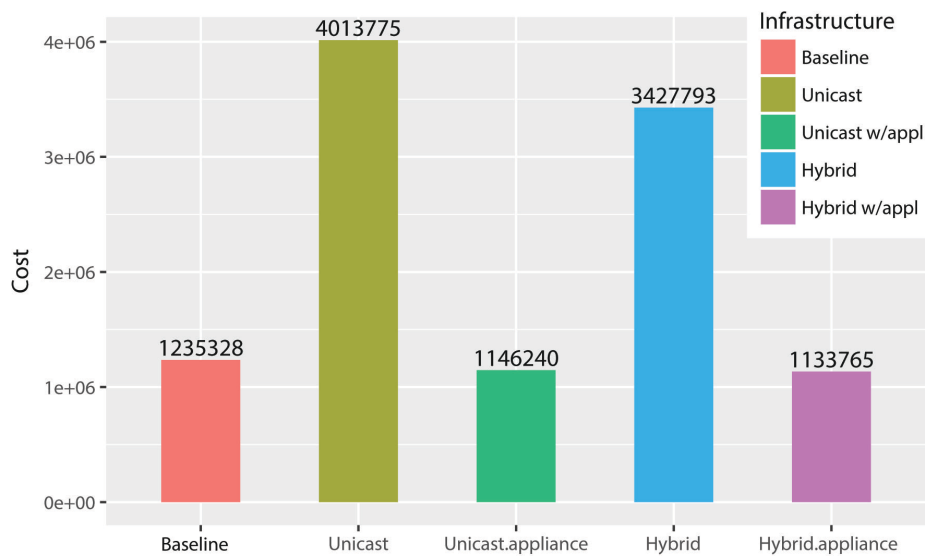
For the North American 2017 scenario, the higher number of TV channels and the higher average Internet consumption in comparison with the Portuguese scenario gives a different set

of estimated values. For the 2017 forecast, the approaches need the same number of OLTs and SRs. Figure 3.12 shows that Unicast-only with appliances approach spends less bandwidth at both SRs and OLTs. However, drawing a comparison with Portuguese 2017 forecast, the gap between the Baseline and Unicast-only is smaller.



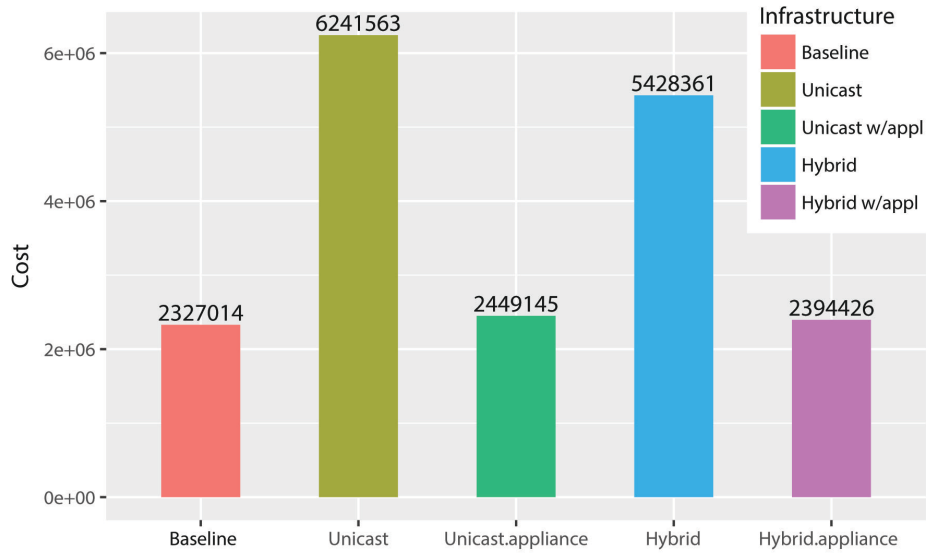
**Figure 3.12:** USA SR and OLT forecast 2017

For the OPEX estimations on the 2017 forecast depicted in Figure 3.13, the Hybrid approach takes the lead as the least expensive, although all three are very close. For this scenario, the large number of TV channels plays a key role in the cost estimation, giving the edge to the Hybrid approach, where this method leverages the efficiency of multicast delivery.



**Figure 3.13:** USA OPEX forecast 2017

On the 2020 forecast, Unicast-only and Hybrid approaches need about more SRs and OLTs to provide the same service as the Baseline method, later resulting in a better cost performance for the Baseline. In the overall, Unicast-only performs better in bandwidth spending, but the necessary infrastructure increases, giving the edge to the Baseline on OPEX estimations, i.e., less than Hybrid, as Figure 3.14 shows.



**Figure 3.14:** USA OPEX forecast 2020

### 3.3 Summary

This chapter provided a forecast approach for unicast and multicast content delivery. For the Portuguese scenario, Unicast-only delivery proves to be the cheapest one on the weight-based OPEX estimates, mostly because of the average number of TV channels and number of subscribers. In the North American scenario for the 2017 forecast, the Hybrid OPEX is the smallest, but the other two methods present very close results. However, for the 2020 forecast, the Baseline spends less, but the Hybrid strategy comes very close, making a viable solution for the future. Finally, the numbers for an architecture with appliances only at the SR make the Baseline approach the least expensive for both scenarios, though the Hybrid one comes close. The Hybrid approach could be improved further, using multicast to broadcast more demanding TV channels like 4K and 1080p60 channels as mentioned before. Finally, for an architecture with appliances only at the SR level, multicast wins over the other two approaches, which proves that distributing network resources through the entire network is a must in the future.



# Proposed Content Delivery Method and Architecture

*As discussed in the last chapter, Telcos must improve the network infrastructure to support future demand for video content while diminishing the overall costs of the infrastructure to stay competitive. A meaningful way to do that is to distribute hardware appliances throughout the network, improving flexibility and capacity while introducing new ways of content delivery. In this chapter it is discussed a new content delivery approach using SDN to improve on-demand content distribution.*

## 4.1 Proposed Content Delivery Method

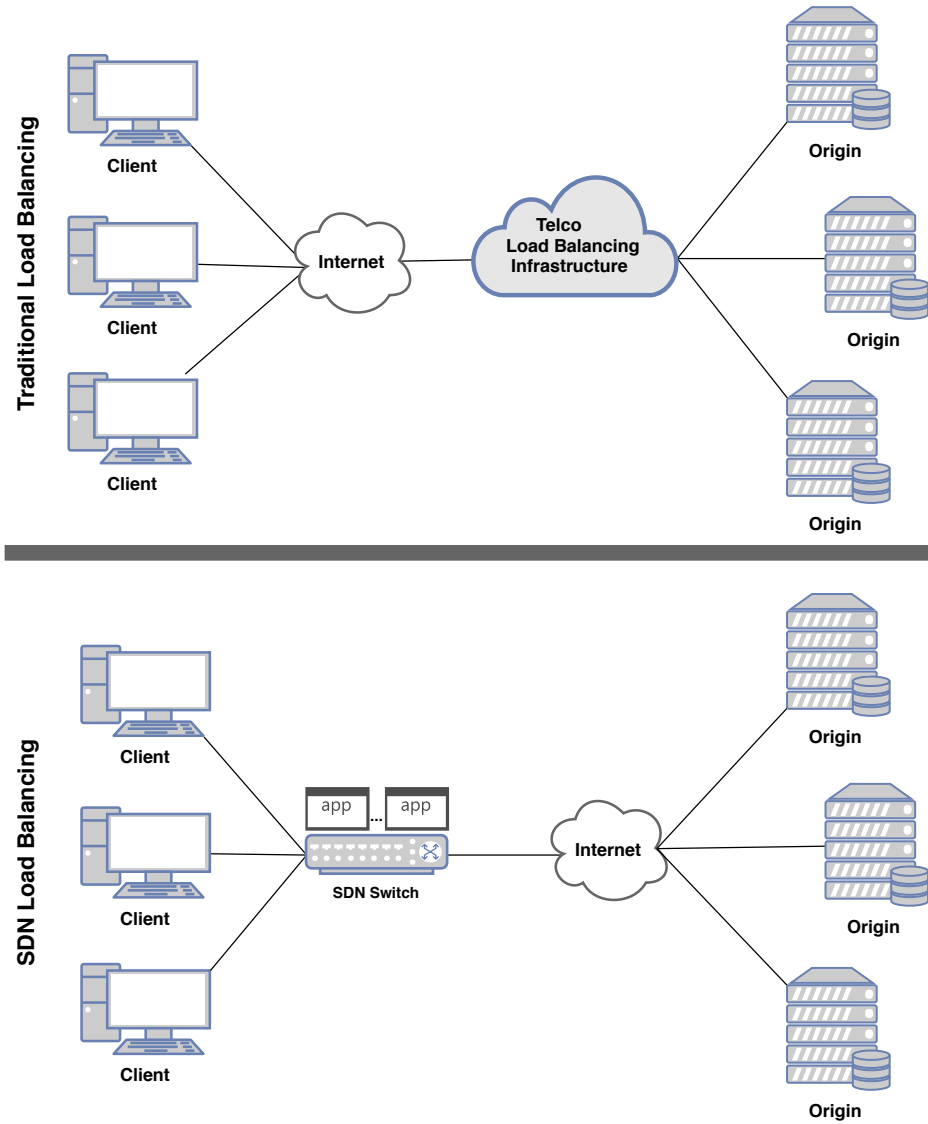
A starting point for Telcos in the content delivery market is to improve the OTT services that they already offer. A good approach for this improvement is to reduce the complexity to manage middleboxes scattered throughout the network infrastructure. In traditional on-demand content distribution with HTTP adaptive streaming, when clients request video content, a load balancer is present at the Telco side, distributing load among a pool of servers usually called the origins. The proposed content delivery method uses an SDN approach to distribute the load at the network edge. Rather than having a central load balancer infrastructure that would process all of the clients' requests, this approach allows for a distributed load decision right at the clients' point of access. For instance, Telcos could deploy this type of hardware and software on their wireless hotspots infrastructure or at the cellular network on the cell towers.

Besides the common origin infrastructure that needs no modification, the core philosophy behind the proposed method relies on two essential components:

- the client player;
- the SDN load balancing application.

Figure 4.1 depicts a high-level overview of the traditional load balancing approach versus the SDN load balancing method proposed.

To have a load balancing decision on the edge of the network that is beneficial to the clients and the origins, the SDN application that runs on top of the SDN controller must collect information about the status of the client session and have pre-installed information about the origins where the load is to be distributed. The following sections will explain in detail these two components: the client's player and the application for SDN load balancing.



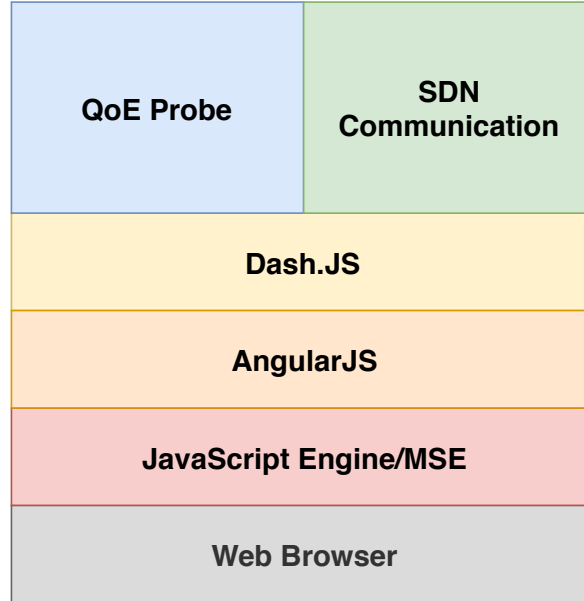
**Figure 4.1:** Traditional Load Balancing versus SDN proposed method

## 4.2 Client Player

The client's player is a critical element because it is the element to which the client interacts with and consumes the content, so it bundles all of the technologies across the infrastructure to provide the video experience. For future-proofing, the player must be codec and browser agnostic, enabling multi-platform compatibility. As discussed on the state of the art chapter, Dash.js developed by DASH-IF (based on the MPEG-DASH standard) provides the before-mentioned features. The only requirement is that the web browser must support HTML5 MSE [19] [20]. Through MSE, video and audio elements can dynamically change without the use of plugins, enabling in this context, the adaptation of the media streaming. For the proposed content delivery method, two components were integrated with Dash.js:

- QoE Probe;
- SDN communication.

Figure 4.2 gives an overview of the client's player elements.



**Figure 4.2:** SDN Load Balancing Overview

### 4.2.1 QoE Probe

The QoE analysis is relegated to the client side because it is in the best position to make a QoE assessment. An established QoE probe is used from [63] and was improved to be more accurate with the help of the metrics provided by Dash.js. The next chapter will describe the thorough work done to make the QoE Probe useful in the environment of this work. The probe was built using both subjective and objective tests and simulates human video scoring behavior taking into account several metrics:

- Frames Per Second (FPS): it is an important metric because sometimes frames are skipped frames, affecting video playback, due to the lack of processing power or network constraints that lead to a lower frame rate video selection;

- For a given codec, Bitrate is a indicator of the video quality that the client can play; the higher the bitrate, the higher the quality (when comparing the videos encoded with the same codec);
- Screen Ratio: the client's device resolution is taken into account to calculate a ratio with the video resolution. The client can have a FullHD screen watching SD content, which is bad for the client, but if a 4K stream is rolling on a FullHD screen, it is bad for the server-side infrastructure;
- Buffer Stalls: when the client's player runs out of buffered chunks, the video playback stops. This is one of the most important metrics because, as seen before, this metric causes the most negative impact.

Moreover, a human memory filter technique is applied to take into account past experiences in the current evaluation, because a buffer stall is different from two consecutive buffer stalls, the last having a more negative impact. In conclusion, the probe calculates QoE for each chunk received, ranging from two seconds to ten seconds. When there is a buffer/play stall, the calculation halts, so when it resumes, the duration of the stall will affect the QoE calculation.

#### 4.2.2 SDN Communication

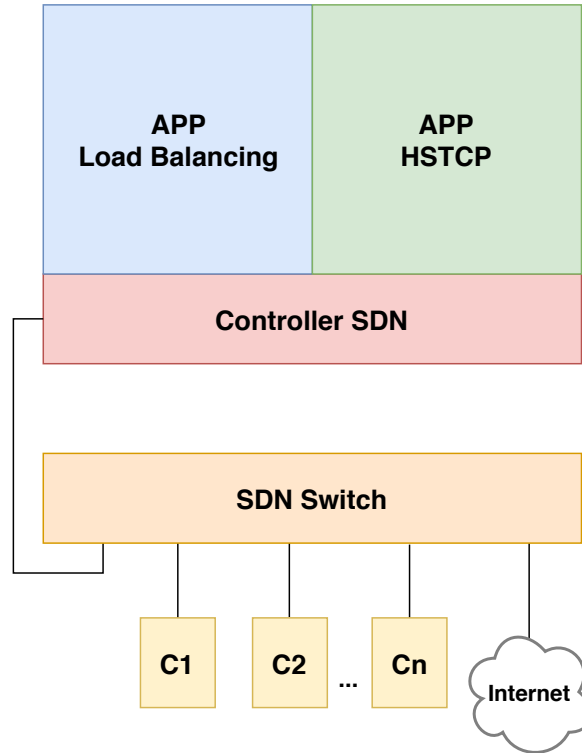
For the QoE calculation to add value to the infrastructure, the player must send it in the most effective way to the SDN controller. However, the QoE value must pass through a normalization process to give the controller the best insight. This is key because a client can have limitations hardware-wise preventing it from reaching the maximum MOS value. If a client has a SD screen resolution, the client is limited, and a plain QoE information to the controller would be misleading, making the controller to try to improve something that is hopeless, impossible. Therefore, the client's hardware plays a role in normalizing the QoE value, and the player sends a *relative* QoE value. A special packet is created to send the relative QoE value that only the controller can process.

## 4.3 SDN Load Balancing

The SDN approach to load balancing allows the distribution of load decision, removing the single location of load balancing on traditional networks. Furthermore, the decision takes insight from the clients' perceived QoE to improve load distribution and client experience. At the same time it eases the process of deploying new policies and configurations across the network, improving flexibility when compared to middleboxes administration.

Figure 4.3 gives a high-level overview of SDN load balancing, where  $C$  stands for clients that are connected to an SDN switch. The switch has a dedicated port for the controller and another one for outside traffic, for Internet access. The controller has two main applications, load balancing and Hot Swap TCP (HSTCP). Therefore, the proposed method can be divided into three categories:

- Processing packets;
- Registering clients;
- Managing clients.



**Figure 4.3:** SDN load balancing overview

### 4.3.1 Processing Packets

Processing packets in a centralized control is the whole point of the SDN approach. At the start, the controller and the switch establish communication through a specific port, and it is when the controller installs the default flows:

- First flow: send everything to the controller, with a priority of zero. This rule allows the initial learning of the controller, to capture packets to apply a specific rule or just enabling basic network connectivity like a switch;
- Second flow: send to the controller all of the packets that have the 8000 TCP destination port, with the priority of 1000. The origins provide video content through port 8000, hence if a client is requesting a TCP connection to this port, it is possible that they are using the service proposed. This port can be dynamically configured;
- Third flow: every packet that matches the special packet that transports the relative QoE is sent to the controller, and the flow has a priority of 1000.

After the switch flow table initialization, when a match occurs, a *PACKET\_IN* event is produced, sending the packet headers and ingress port to the controller of the respective packet. The controller receives the *PACKET\_IN* and applies the policies suited for that packet, resulting in a flow installation on the switch and a *PACKET\_OUT* event happens. A *PACKET\_OUT* occurs when the controller learns from the processed packet and injects it back to the network, avoiding a dropped packet everytime a packet goes to the controller. Also, the controller has pre-installed information about the origins and the point of entry IP to the infrastructure, but it must learn which switch ports lead to those networks. Thus, the controller also processes ARP packets to learn which port to forward the packets to reach the requested network.

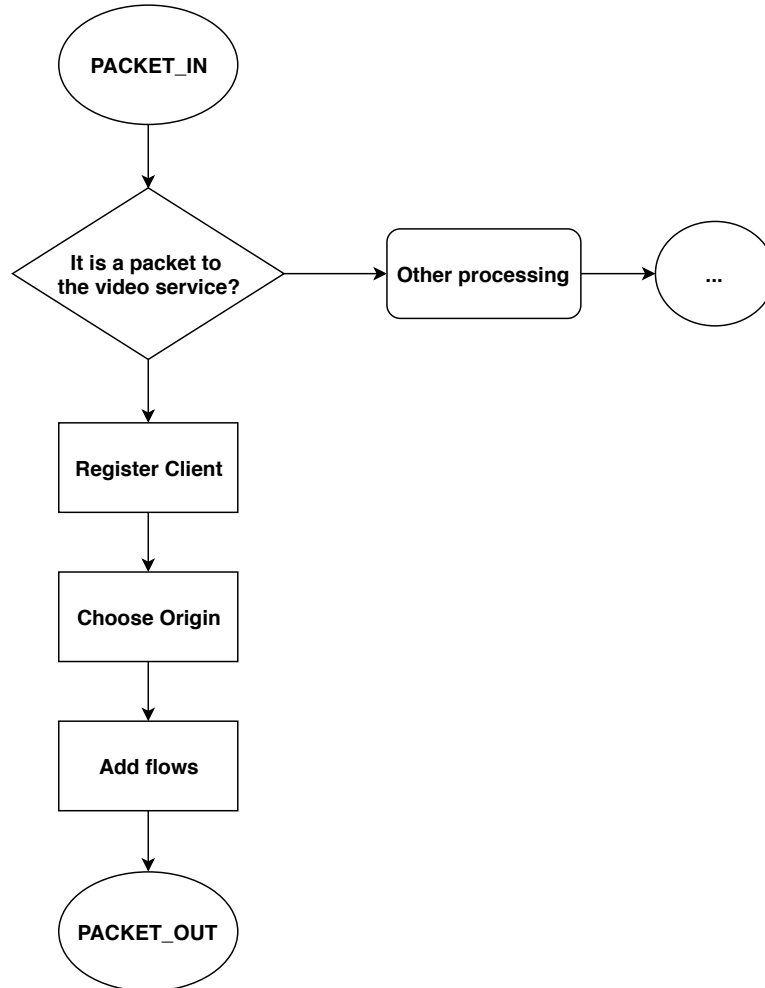
### 4.3.2 Registering Clients

To improve clients' QoE, the controller must keep information about each client session to act upon. When a client starts watching a video on this infrastructure, the pre-installed second flow generates a match, sending the first package of the TCP session to the controller. Then, the package is processed: if the IP destination matches the point of entry to the origins, a client registration proceeds. The controller has a client table, which records the Ethernet address, the IP address and later the origin to be attributed. The clients' table has other fields that will be described in the next subsection. Then, an origin is chosen to serve this particular client. The controller chooses the origin with fewer clients, using a round-robin approach, adding that information to the client's table entry and updating the origin table entry to account for one more client added. Following the client's registration, two flows are installed on the switch:

- Client to origin:
  - Match by the client's port, Ethernet destination of the point of entry IP, Ethernet of the client, TCP protocol and port destination (8000);
  - Actions: replace the IP and Ethernet destinations with the origin chosen, and update the port to match the origin's port on the switch.
- Origin to the client:
  - Match by the port, Ethernet and IP address of the chosen origin, and TCP protocol and port source (8000);

- Actions: replace the Ethernet and IP addresses to the point of entry IP and send it to the client’s port.

When the flows are installed, the packet is then returned to the network through a *PACKET\_OUT*. Figure 4.4 gives a simplified overview of the client registration process.



**Figure 4.4:** Client registration high-level view

### 4.3.3 Managing Clients

Every component described so far is integrated to allow client session management and intelligent load balancing. Every ten seconds the client sends a relative QoE to the network, producing a match on the switch flow table, the third pre-installed flow: forward the special packet carrying the QoE value to the controller for further processing. If the QoE falls within a threshold of 2.3 or less, the controller can change the client to a new origin seamlessly. This value is chosen because the MOS scale goes from zero to five (the higher, the better), but in practice, the average of human video evaluations never reaches a score of zero or five; thus, 2.3 is the *middle* value of the scale. In the proposed content delivery method, this feature is

called HSTCP and refers to the capability of changing a client TCP session to a new origin without the client noticing it. Therefore, managing a client can be divided into:

- assessing if the client needs a new origin and it is allowed to change;
- electing a new origin;
- update origin's and client's table;
- delete old flows and add new ones;
- terminate the old TCP session.

As stated before, the controller manages information from the clients and the origins. Figure 4.5 illustrates the fields that each entry possess, on the client's table and the origin's table respectively. The SDN controller saves seven fields on the client part:

- ether: Ethernet address;
- ip: IP address;
- changes: how many times has a client change origin;
- next: a timestamp value that indicates when a client can change origin;
- zeros: to count how many times the client has sent a QoE of zero, meaning playback stoppage;
- origin: is the client's current origin;
- old: is the last origin that the client changed.

On the origin part, the controller saves five fields:

- ether: Ethernet address;
- ip: IP address;
- port sdn: is the port where the switch has access to the origin;
- score: for every time a client changes from an origin, this value is incremented;
- number of clients: is the current number of clients on a given origin.

#### Client

ether	ip	changes	next	zeros	origin	old
-------	----	---------	------	-------	--------	-----

#### Origin

ether	ip	port sdn	score	number of clients
-------	----	----------	-------	-------------------

**Figure 4.5:** Client and Origin table fields

The following pseudo-algorithm describes in a simplified manner how the controller processes the relative QoE value sent by the client.

```

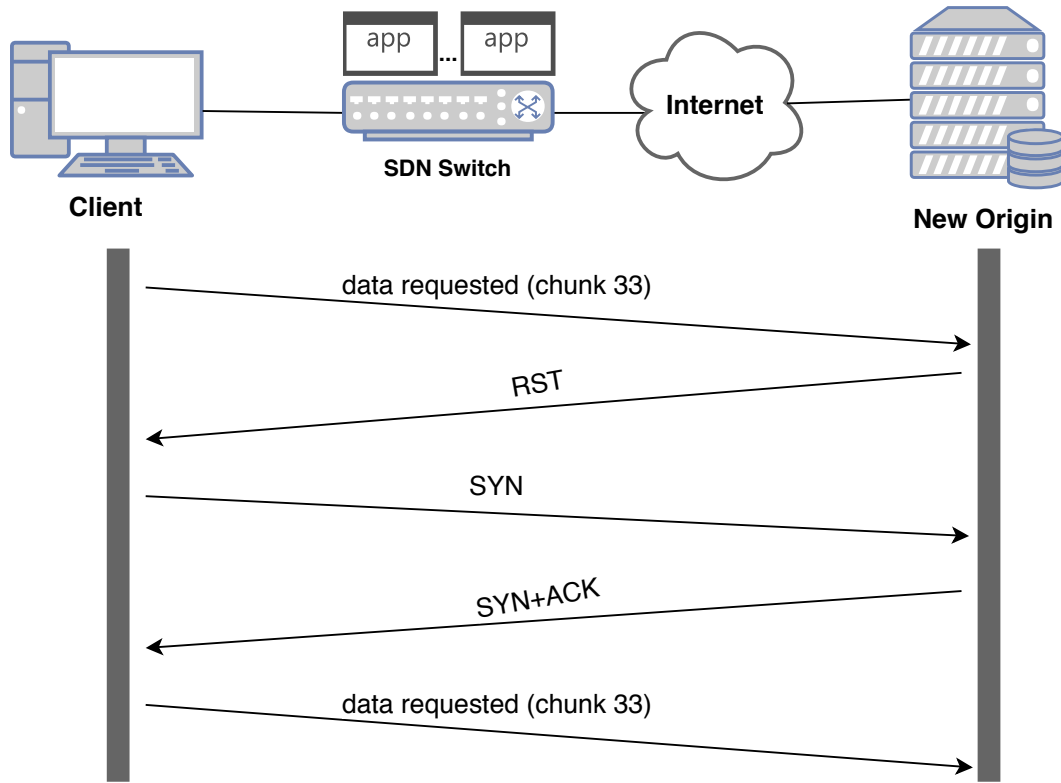
packet_in_handler(){
    if (the packet is a special packet with QoE?){
        if (relative QoE is < 2.4 & != 0){
            if (client is allowed to change?){
                change_client()
            }
        } else if (relative QoE == 0){
            if (client has three consecutive zeros){
                change_client()
            }
        }
    } else if (known client changing Origin?){
        add new flows
        terminate old TCP connection
        packet_out
    }
}

change_client(){
    elect new origin
    update client and origin tables
    delete corresponding flows
}

```

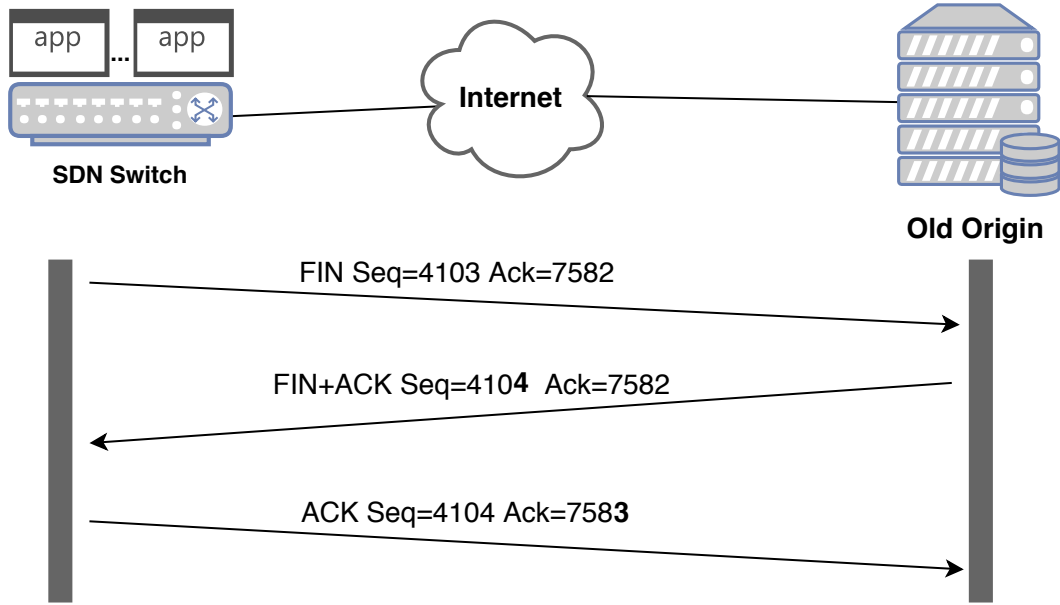
For a client to change origin, various requirements must be met. If a client sends a QoE between 2.3 and 0.1 (middle of the MOS scale, the higher, the better), the process to change the origin starts. A timestamp is created and compared to the timestamp of the *next* field to check if the client is allowed to change the origin. For each time a client changes an origin, the *next* value is 25 seconds times the number present on the *changes* field. Thus, a client that changes an origin has some time to stabilize before it changes again (the QoE value is sent in a ten seconds cycle). At the same time, it saves infrastructure resources because a client can have unfavorable conditions that will not allow it to go higher than 2.3, avoiding certain clients to jump all the time. Next, if the client is allowed to change, the controller will choose another origin, excluding the actual, on the same round-robin approach. The fields from the client's table are updated: the *changes* is incremented one value, the *next* changes accordingly, the *origin* is updated to reflect the controller choice and the old origin is saved on the *old* field. On the origins table, the new origin increments one in the *number of clients* field and the old origin decrements one, and adds one to the *score* field. Though, some challenges arise in moving the TCP session seamlessly. The controller deletes the two flows related to the client and the *old* origin, and installs a new flow with a timeout to block communication from the old origin to the client because if the client becomes absent, the old origin will try to re-establish the TCP session. Due to the previous flows deletion, the next packet of a client moving to a new origin will end up in the controller because of the second pre-installed flow. Now, the controller recognizes the client and uses the field *origin*, that is the new origin

given to the client, and initiates the second part of the Hot Swap TCP. New flows are added between the client and the new origin, and the packet is injected onto the network. The new origin will not recognize the client's session and will send a *RST* packet, forcing the client to send a *SYN* to start (restart) the TCP session, followed by a *SYN+ACK* packet from the new origin, as the figure 4.6 depicts. Now the client keeps on requesting the chunks that it needs.



**Figure 4.6:** Client restarting the TCP session with the new Origin

However, the capture of this packet, provoked by the flows' deletion, is critical for the TCP session termination with the old origin. The controller will use another application developed for this purpose to terminate the TCP session gracefully. With the fields *sequence* and *acknowledgment* extracted from the packet, the controller is able to forge two new packets, simulating the client wish to terminate the session: *FIN* and *ACK*. The first informs the old origin that the client wants to terminate; the origin then sends a *FIN+ACK* that never reaches the client due to the flow with timeout added before, and waits for the client to send a *ACK*, the second packet created. The controller can produce the two packets without the response from the old origin, so it sends the first packet, waits two milliseconds and then sends the *ACK* packet. Figure 4.7 depicts an example of how the controller gracefully ends the TCP connection with the old origin on behalf of the client.



**Figure 4.7:** Controller terminating the TCP session with the old origin

Lastly, to avoid origins overloaded with other services or origins that for some reason turned into *zombies*, the controller has a failback process. If a client's playback stop, where no QoE calculation happens, the client sends a QoE of zero to signal the controller that the video playback is frozen. However, most of the times the problem comes from the client, so it needs to send three consecutive zeros to change the origin. Usually, if the conditions are not good on the client-side, the client will not be able to send the three packets, but if it is from the origin side, the controller will move the client to another origin.

In the end, the controller can indirectly evaluate the origins' performance through the client's QoE and move live TCP sessions to other origins without them noticing it, giving way to a more intelligent and flexible load balancing service.

## 4.4 Summary

This chapter described the components that make up the proposed content delivery approach: the clients' player where Dash.JS was integrated with a QoE probe and SDN communication feature to send the MOS value to the controller; and the load balancing application that runs on top of the controller and makes use of the MOS value for load balancing decision.



# Implementation

*Behind the concepts and methodologies introduced in the last chapter, the implementation chapter describes the technologies behind each entity that leads to the integration of the proposed content delivery method. Naturally, this chapter goes into technological detail on three major sections: the clients, the origins and the SDN controller.*

## 5.1 Clients

One of the goals of the proposed content delivery method is to improve the clients' perceived QoE. As discussed in chapter 4, the client must support real-time QoE estimation and communication with the SDN controller, on top of the adaptive streaming technology.

Major companies like Google or Netflix use the MPEG-DASH standard in their video players, and as stated in chapter 2, the DASH-IF, which those companies contribute to, produces the Dash.JS player which:

- is multi-platform: the browser only has to support HTML5 MSE;
- has a free license for commercial use;
- implements the MPEG-DASH best practices;
- has several adaptation algorithms;
- is written in AngularJS.

AngularJS is a framework for dynamic web applications that extend the HTML syntax and reduces the amount of JavaScript coding. It offers a two-way data binding between the model (Javascript objects) and the view (user interface) and does it automatically. Also, it enables techniques like dependency injection and inversion of control.

The QoE probe from [63] was written on C#; hence, a process of code refactoring was necessary to translate to the AngularJS environment. However, other challenges arose, since the probe was developed to process metrics from Microsoft Smooth Streaming that differ slightly in some cases from the Dash.JS metrics. For instance, buffer events had to be re-engineered to preserve the probe behavior. Also, they suffered from a mutual exclusion

problem, leading in some specific cases to a wrong calculation of the length of the buffer stall. Another problem was the maximum resolution supported by the probe, HD (1280x720p), that nowadays does not reflect the maximum resolution on HTTP adaptive streaming services that can go up to 4K resolution [64]. Accordingly, bitrate metrics and screen ratio can reach higher values and have different meanings. Thus, the probe's formulae were modified to sustain this change, preserving the overall behavior.

Equation 5.1 illustrates the main formulae of the original QoE probe versus the new one 5.2. Maintaining the naming from the original code, variables *tmp1* and *tmp4* suffered changes. The first one is associated with Bitrate calculation while the second is related to screen ratio calculation. To account for higher bitrate streams, *tmp1* suffered a normalization: where the 5000000 value is the maximum bitrate of the original probe (an HD stream) and the 15000000 is the maximum bitrate considered for a 4K stream. The change was not enough due to the *tmp1* further calculation, the arc-tangent. For Full HD and 4K Bitrates, the arc-tangent behavior with the *tmp1* value was different from the expected one; thus, a different normalization is applied in these cases.

The QoE Probe did not account for the situation where a higher resolution stream is running on a lower resolution screen; in the original probe that would increase the estimated QoE value incorrectly. Thus, for infrastructure resource optimization and probe integrity, it is important to deal with this situation. It was performed an empirical analysis, testing different resolutions, creating a normalization of the *tmp4* variable that would produce the right behavior on the further logarithmic calculation. Then, several tests were performed to full proof the new QoE Probe, resulting in minor changes.

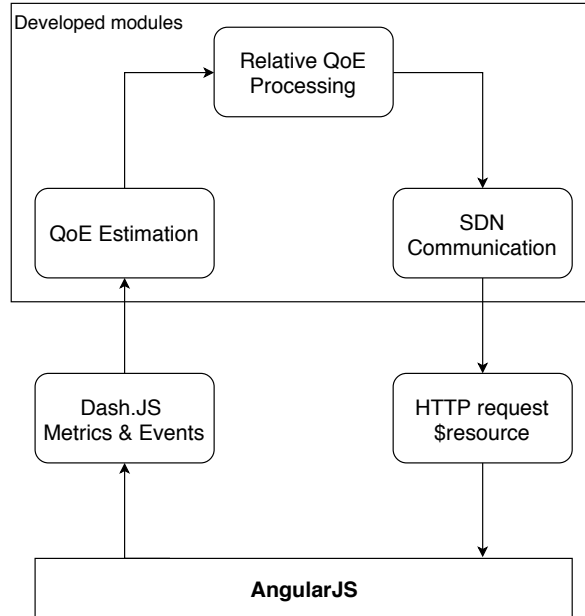
$$\begin{aligned}
tmp1 &= 0.00000142 \times bitrate \\
tmp2 &= 0.3032 \times fps \\
tmp3 &= 3.064 \times delayold\_value + 0.5407 \\
tmp4 &= 0.05652 \times screen\_ratio + 1.756 \\
VideoScoreUP &= 2.38 + 1.027 \times \arctan(tmp1) \times \log(tmp2) - \log(tmp3) - \log(tmp4)
\end{aligned} \tag{5.1}$$

$$\begin{aligned}
tmp1 &= (0.00000142 \times (5000000/15000000)) \times bitrate \\
tmp2 &= 0.3032 \times fps \\
tmp3 &= 3.064 \times delayold\_value + 0.5407 \\
tmp4 &= \begin{cases} 0.05652 \times (1 - screen\_ratio) + 3 & , screen\_ratio < 1 \\ 0.05652 \times screen\_ratio + 1.756 & , screen\_ratio \geq 1 \end{cases} \\
VideoScoreUP &= 2.38 + 1.027 \times \arctan(tmp1) \times \log(tmp2) - \log(tmp3) - \log(tmp4)
\end{aligned} \tag{5.2}$$

The final component added to the modified Dash.JS player was the *SDN* communication. As previously discussed in chapter 4, the client sends a packet with a relative QoE value to give the best information to the controller. However, creating a packet on a web browser to later match in a flow table on a forwarding device created two challenges:

- creating and sending a packet on a web browser;
- creating an OXM.

To solve the first challenge, the most obvious way was to generate and send a UDP packet, where the data payload would transport the QoE value. That would solve the second challenge immediately because UDP is a low-level protocol from the transport layer; a layer where the controller can process information because OpenFlow modules are already built to deal with these protocols. Web browsers were built on top of HTTP which runs on top of TCP to provide reliability, to send and receive packets in the same orderly sequence [65]. There are methods to overcome this difficulty and use UDP packets. Protocols like Quick UDP Internet Connections (QUIC) offer a world where web browsers can use UDP as the transport protocol, but it adds complexity to the client's player and creates the second challenge. If a controller needs to process HTTP data, it needs to perform deep packet inspection, which is not desirable, and an OXM to perform it. Also, for every modified network protocol like QUIC, a new OpenFlow module must exist to process the packet. Although the OXM feature introduced in OpenFlow 1.2 makes that task more accessible, it is still a complex and time consuming. Hence, the most straightforward way to inform the controller about the client's QoE is to use TCP. So, the player generates an HTTP request to a unique IP that the controller is expecting. Then, the port's number to each client that makes the request, has the QoE encoded within the last three numbers of the port: 50378 means a 3,78 value for example. This is done with the AngularJS *\$resource* directive, that creates a request with a timeout to specifically create only one packet. In summary, the controller installs a flow on to the switch that filters any TCP connection to the unique IP and forwards it to the controller. The controller then processes the packet, and with the source IP, it can correlate the QoE value to a client. Figure 5.1 illustrates the sequence for QoE calculation and dispatch.



**Figure 5.1:** Client's Player components integration

## 5.2 Origins

The content that the client requests resides on the Origins infrastructure. For that content to be provided to the client, the content must be encoded respecting the MPEG-DASH standard, and the Origin must have web HTTP server capabilities adjusted to HTTP adaptive video streaming. Therefore, this section naturally divides into two segments: content preparation and content delivery.

### 5.2.1 Content Preparation

As discussed in chapter 2, HTTP adaptive video streaming is based on video fragmentation, to provide more flexibility in video delivery and quality shifting. The content is encoded in several qualities for audio and video in a fragmented style, and an MPD is created to describe the media segments and other attributes. A DASH player then uses the MPD to calculate the URLs for each segment to perform the requests, downloading the chunks. The tools used to prepare the content for the origins were *FFmpeg* and *Bento4* tools. *FFmpeg* is a widely used tool for video transcoding, video scaling among others. Here, it is used to generate mp4 files with a predefined Group of Pictures (GOP) setting in different qualities. The GOP setting forces the encoding process to add an intra-frame (independent of the frames that precede or follows) at the beginning of each group, making possible the seek and playback at the start of a group of pictures. The value chosen correlates to the desired duration of a chunk and the original video frame rate. Then, two tools from Bento4 are used:

- *mp4fragment*: fragment the video with the desired chunk length, the value used to compute the GOP value;
- *mp4-dash*: takes the fragmented videos with different qualities and creates the DASH presentation, organizes the media files and creates the MPD.

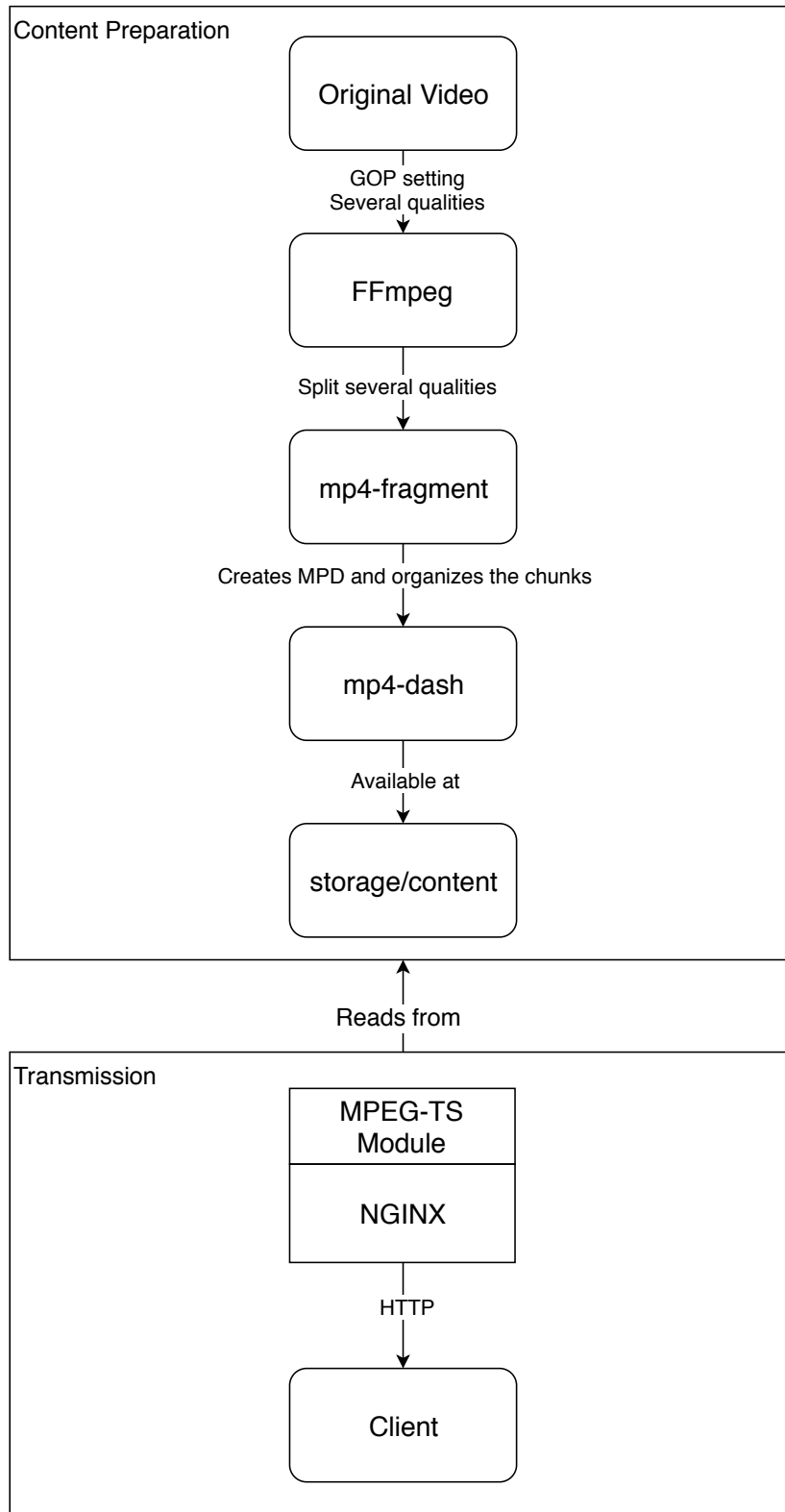
### 5.2.2 Content Transmission

Several companies offer web server products like Microsoft, Apache or NGINX. Except for the web server from Microsoft, the other two are open-source into some degree, making it ideal for deployment. Thus, various performance analysis [66] [67] indicate that NGINX performs better than Apache in response time, CPU utilization and memory usage, making NGINX a more future-proof option [68]. Therefore, NGINX was chosen as the web server to provide DASH content, but NGINX alone is not capable of distributing on-demand DASH content, only live because it just has one stream quality. To provide that functionality, NGINX is compiled with the *NGINX MPEG-TS Live Module*<sup>1</sup>. Some minor issues manifested, like the Cross-Origin Resource Sharing (CORS) problem that lead the browser to block HTTP requests, disabling video reproduction at the client. Additional NGINX configuration overcame these challenges.

Figure 5.2 gives an overall view of the two segments previously discussed.

---

<sup>1</sup>NGINX MPEG-TS Live Module, Available: <https://github.com/arut/nginx-ts-module>



**Figure 5.2:** Content preparation and transmission overview

## 5.3 SDN Controller

The piece that glues the proposed content delivery method is the SDN controller. With it, the client participates on the load balancing decision, bringing more knowledge to optimize the client's QoE, and also to improve load efficiency with new metrics. Therefore, this section describes the chosen controller and the deployment of the same at the edge of the network.

### 5.3.1 The Controller

The choice of the SDN controller must take into account the requirements of HTTP adaptive streaming, which is delay-sensitive, and the constraints of the proposed work for rapid prototyping. Two controllers were tested: Floodlight [69] and Ryu [70].

The Floodlight controller was chosen to run some experiments due to being easy to develop Java language and for the overall OpenFlow support. However, simple tasks were complex to develop when compared to Ryu, and documentation was scarce. Also, as discussed in chapter 2, Ryu had the best delay results compared to other controllers and a broader OpenFlow compatibility. The Ryu controller uses a component-based approach for software-defined networks, it is developed in Python, which enables faster developing, it is ideal for small-scale scenarios, and the project is open-source with an Apache 2.0 license.

The component-based approach means that the Ryu API enables multiple applications on top of the controller. Ryu applications are single-threaded, asynchronous and use events to communicate between them. Each one has a queue to process events, FIFO (first in, first out), that preserves the order of events. Figure 5.3 provides a component examination of the Ryu controller.

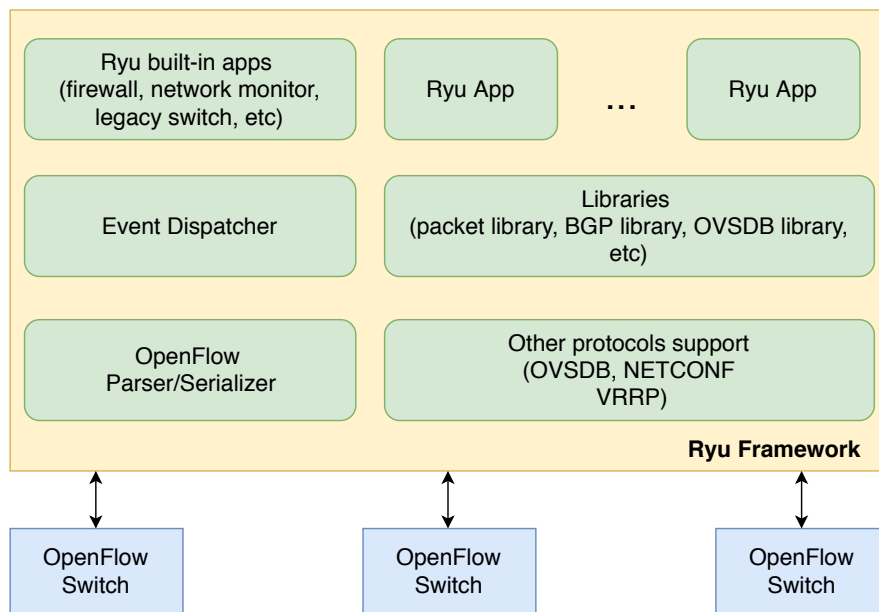


Figure 5.3: Ryu framework overview

### 5.3.2 Deployment

One of the setbacks of transitioning to an SDN environment is the lack of hardware variety and high cost. SDN controllers are easy to choose because they are a piece of software, yet they need special network hardware to implement the controller's actions. Switches compatible with OpenFlow are much still in the enterprise realm, reaching thousands of euros per equipment easily. Northbound Networks <sup>2</sup> offer a small SDN switch targeted at researchers and SDN learning enthusiasts, with four 10/100 Fast Ethernet ports. It supposedly supports OpenFlow 1.0, 1.3 and 1.4 but it was observed that many OpenFlow functionalities were still being developed. Thus, another path was taken, network emulation.

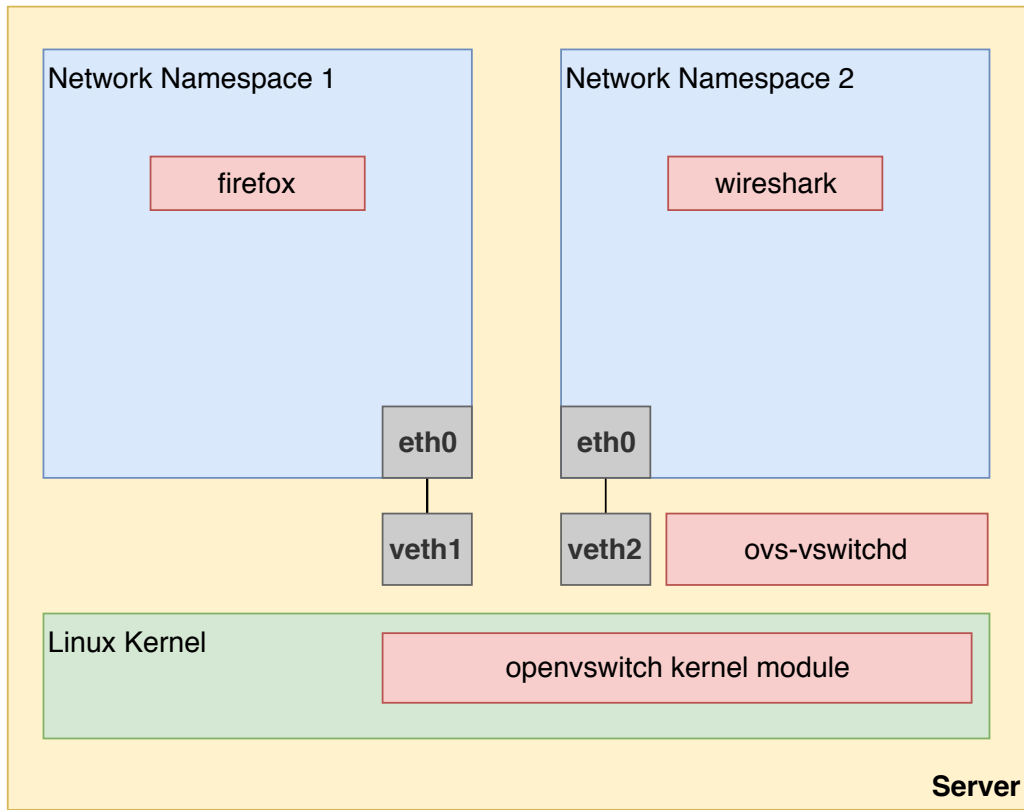
Mininet <sup>3</sup> is a network simulator that creates a virtual network with hosts, connecting to virtual switches, controllers (internal or external) and links. It runs on Linux and takes advantage of the flexibility that Linux network stack offers. Therefore, Mininet positions itself as a network testbed for developing OpenFlow applications. It offers a Python API to create more advanced topologies, configurations, and other features; it offers a Command Line Interface (CLI) to interact in real time with the hosts, links, and switches, creating an overall fast prototyping environment that can build complex network applications. Mininet uses process-based virtualization rather than process abstraction, enabling a lightweight approach to full topology virtualization. With Linux network namespaces, individual processes can have independent network interfaces, ARP tables, routing tables, and applications. To run a GUI(Graphical User Interface)/X11 application on a host, Mininet sets up an X11 tunnel to the host over the CLI *xterm*. Therefore, Mininet can virtualize a network with fewer resources when compared to full virtualized networks, while using a real network stack, connecting hosts to switches through virtual ethernet pairs. To add more flexibility to the control plane, the switch implementation from the Open vSwitch project is used rather than the default Mininet switch. Open vSwitch is a project that offers a multilayer virtual switch that usually bridges traffic between virtual machines and the outside world, and among others, it supports the OpenFlow protocol [71], making it the ideal virtual OpenFlow switch to work with the chosen controller. In conclusion, Mininet offers a testbed to emulate a network (with real code) that, through virtual ethernet pairs, connects the hosts to the root network namespace, where the Open vSwitch and the controller reside.

Figure 5.4 depicts the Mininet environment with the Open vSwitch integration. The *ovs-vsitchd* represent the virtual switch to whom the virtual hosts are connected over virtual ethernet pairs (veth). Also, it is the point where the links for the controller and external network are connected.

---

<sup>2</sup>Northbound Networks, Available: <https://northboundnetworks.com/products/zodiac-fx>

<sup>3</sup>Mininet, Available: <http://mininet.org/overview/>



**Figure 5.4:** Mininet and Open vSwitch integration

## 5.4 Summary

This chapter described the challenges of implementing the proposed architecture. Sending the QoE value from the clients' browser to the controller, proved to be one of the greatest difficulties on the client-side of implementation. Further, the process of preparing and serving the content on an Origin with NGINX was described in detail. On the controller side, for the reasons discussed, hardware was not an option to run the controller and associated components; thus, simulation was used to implement the chosen controller, Ryu.

# Proof of Concept and Evaluation

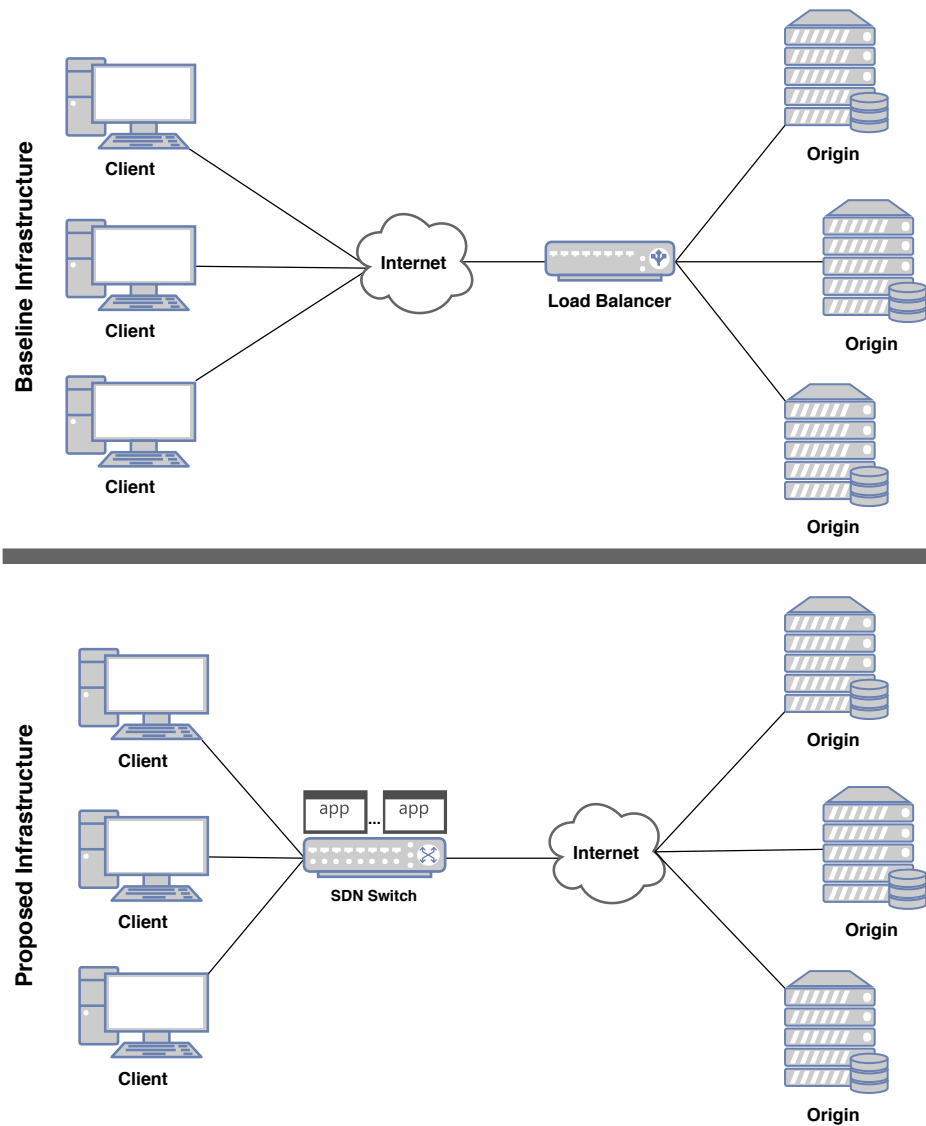
*All of the subjects discussed in the previous chapters lead to the evaluation of the proposed content delivery method. For a performance comparison between the baseline and the proposed approaches, the proof of concept scenario considers a traditional HTTP adaptive streaming infrastructure, running the same client, the same content, and the same Origins. Accordingly, this chapter is divided into three sections: scenarios, testbed and evaluation.*

## 6.1 Scenarios

To evaluate the proposed content delivery approach, a traditional infrastructure must be defined. Nowadays, Telcos have complex load balancing infrastructures, with multiple load balancers that can communicate among themselves and even with the Origins. Due to the complexity of replicating such infrastructure, a simpler approach is taken. Also, the proposed method does not need control communication with the Origins; hence the traditional infrastructure, which will be called baseline infrastructure from this point forward, does not implement such communication. Figure 6.1 illustrates the baseline infrastructure elements and the proposed infrastructure elements.

To compare the baseline versus the proposed approach, the need for multiple scenarios arise. The scenarios contemplate client side and server infrastructure throttling. On the first one, a set of metrics are used to represent a client-side scenario:

- Bandwidth: maximum available bandwidth (Mbps);
- Delay: packet delay (milliseconds);
- Jitter: packet delay variation (milliseconds);
- Loss: packet loss (percentage).



**Figure 6.1:** Baseline vs Proposed Infrastructure

The metrics' purpose are two-fold: they are used to represent a technology scenario limitation and to throttle that scenario further. The Firefox developer tools<sup>1</sup> inspire as a base for the four technology scenarios created. Nine sub-scenarios are created to throttle further the technology scenarios, first considering normal values for the several technologies, and then considering a larger variation in the network metrics, as is depicted in tables 6.1 and 6.2.

---

<sup>1</sup>Firefox developer tools, Available: [https://developer.mozilla.org/en-US/docs/Tools/Responsive\\_Design\\_Mode](https://developer.mozilla.org/en-US/docs/Tools/Responsive_Design_Mode)

**Table 6.1:** Technology Scenarios

Selection	Bandwidth (Mbps)	Delay (ms)
Regular 4G	32	20
ADSL	24	5
Wi-Fi	50	2
Wired	100	1

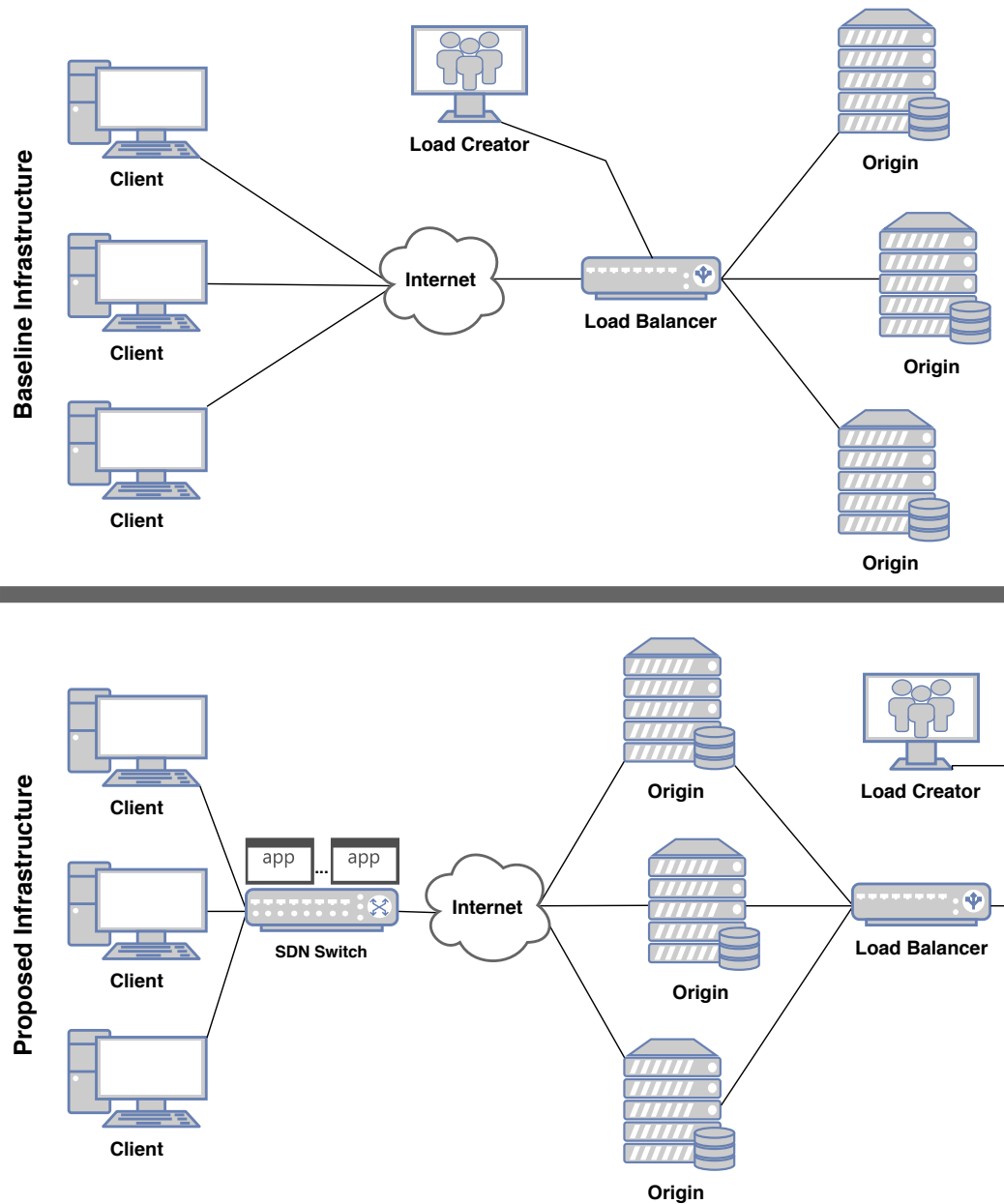
**Table 6.2:** Sub-Scenarios Throttle

Type	Delay (ms)	Packet Loss(%)	Jitter (ms)
A	0	0	0
B	25	0	25
C	25	2.5	25
D	100	0	100
E	100	5	100
F	175	0	175
G	175	7.5	175
H	600	0	600
I	600	2.5	600

The server infrastructure throttling is divided into three parts:

- Part 1: no throttle;
- Part 2: overall infrastructure throttle reaching almost full capacity;
- Part 3: one of the Origins is throttled to almost full capacity.

To provide the throttle on one Origin, the Load Creator performs requests at one Origin directly. However, the load on the overall infrastructure is seamlessly divided: the Load Creator stresses the Load Balancer with requests, and for the proposed content delivery approach, it provides an uniform load on all Origins. Figure 6.2 illustrates the placement of the Load Creator on the overall infrastructure, for the baseline and the proposed infrastructure.



**Figure 6.2:** Baseline vs Proposed Infrastructure with Load Creator

In conclusion, the client-side throttling will consider in the test the Dash.JS and the QoE calculation, creating various situations where the baseline and the proposed infrastructure will react differently. Furthermore, the proposed content delivery approach aims to improve user perceived QoE, also taking into consideration the server-side circumstances.

Therefore, the part 2 of the throttling simulates an infrastructure with a high load, while part 3 simulates an Origin that is overloading with another service or an Origin that stops working correctly, to verify again how the two infrastructures react to these various conditions. Because the client-side and server-side throttling are combined, part 1 accounts to no server throttle.

## 6.2 Testbed

This section describes the hardware and software necessary to execute the discussed scenarios, the way data is retrieved, and some challenges that shaped the overall testbed.

### 6.2.1 Hardware

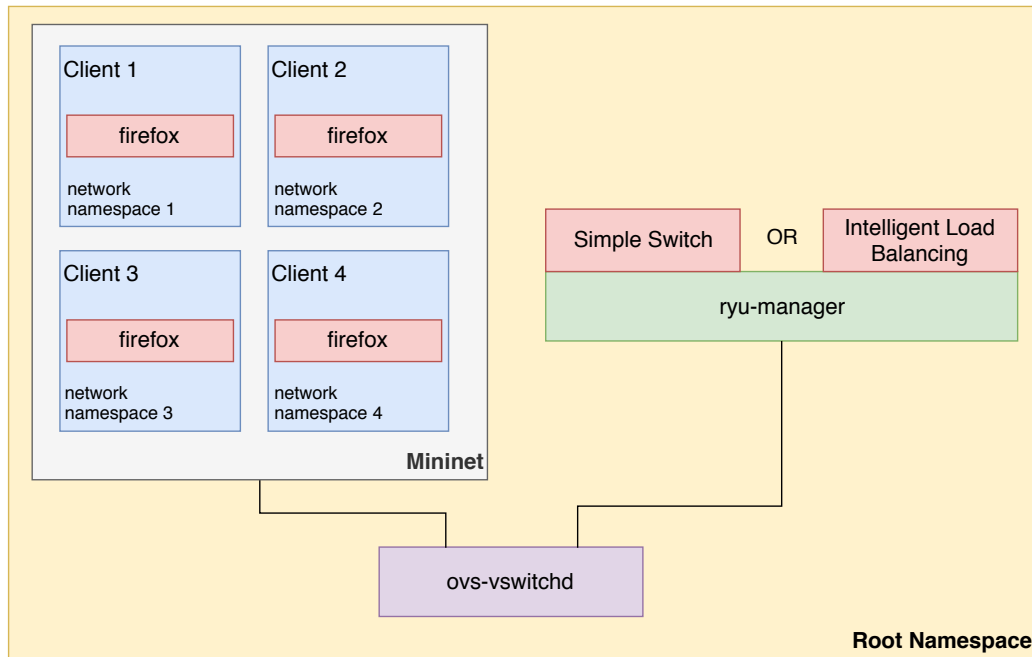
The overall testbed divides into server-side virtual machines and into a machine where 4 clients operate. On the server-side, 5 machines were created: 3 Origins, 1 Load Balancer, and 1 Load Creator that performs high volume requests to stress the infrastructure. Specifications for the virtual machines and clients' desktop are listed on table 6.3.

**Table 6.3:** Testbed machines specification

	Origin	Load Balancer	Load Creator	Clients' Desktop
Memory	2 GB	4 GB	16 GB	32 GB
Processor	2 Cores	4 Cores	8 Cores	8 Cores
Disk	32 GB	20 GB	32GB	512GB NVME

### 6.2.2 Software & Challenges

To meet the scenarios and infrastructure requirements, the network is partially emulated with Mininet. As discussed in previous sections (2 and 5), Mininet has the capability of virtualizing hosts with process-based virtualization and network namespaces. To build a seamless client testbed, Mininet virtualizes the hosts and others for both Baseline and proposed infrastructure with a key difference: the Ryu application. Ryu provides several out-of-the-box applications, among them a legacy switch. A legacy switch in SDN is a straightforward application: simple match fields with a port forwarding action associated. Thus, for the Baseline method, a Ryu legacy switch is used, allowing connectivity to the outside network where the Origins are. Figure 6.3 illustrates the clients' desktop elements: the controller, the virtual switch and the clients, all running in the same machine.



**Figure 6.3:** Clients' Desktop

To stress the server-side infrastructure, an HTTP load testing tool is a viable option. Apache bench and Vegeta were used to produce load on the infrastructure, but the latter proved to be more flexible with more options for specifying loads. Also, Vegeta helped to uncover a performance issue related to the Load Balancer setup later discussed. For load balancing, NGINX and HAProxy<sup>2</sup> were considered, but the latter was chosen because:

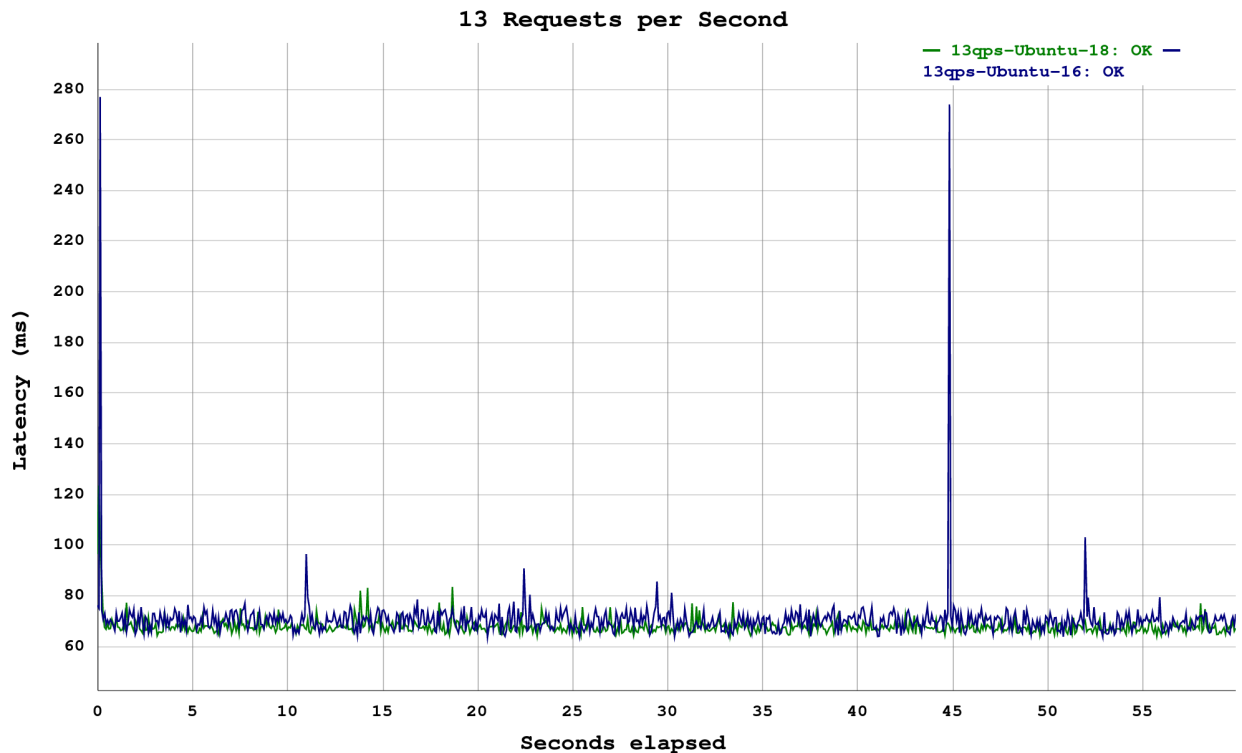
- it was designed from the start to be a high-performance load balancer, outperforming NGINX in this category;
- it is genuinely open source, while NGINX for some options like statistics on load balancing is paid;
- it is a mature solution used by several major companies [72].

Because Ubuntu had released its new version of the Long Term Support Operating System (18.04 LTS) a few months before starting the testbed, it was wise to future proof the testbed using this version instead of the two-year-old version that would receive just one more year of support (16.04 LTS). At the Origins virtual machines, no performance issues were detected with NGINX serving the content, when compared to the Ubuntu 16.04 version, but the HAProxy strangely was not matching the speed of the Gigabit connection. In practical terms, the maximum throughput is never achieved due to several factors, but with Vegeta load creation, the HAProxy on Ubuntu 18.04 was farther from the theoretical maximum. Vegeta was set to request a 7.2MB chunk, meaning for a Gigabit connection, 125MB/s, the theoretical maximum requests per second was 17, but 13 requests were the maximum achieved without performance degradation. At the same server, with the same hypervisor (Proxmox), same virtual machine resources, same HAProxy configurations but with Ubuntu 16.04.5, the Load

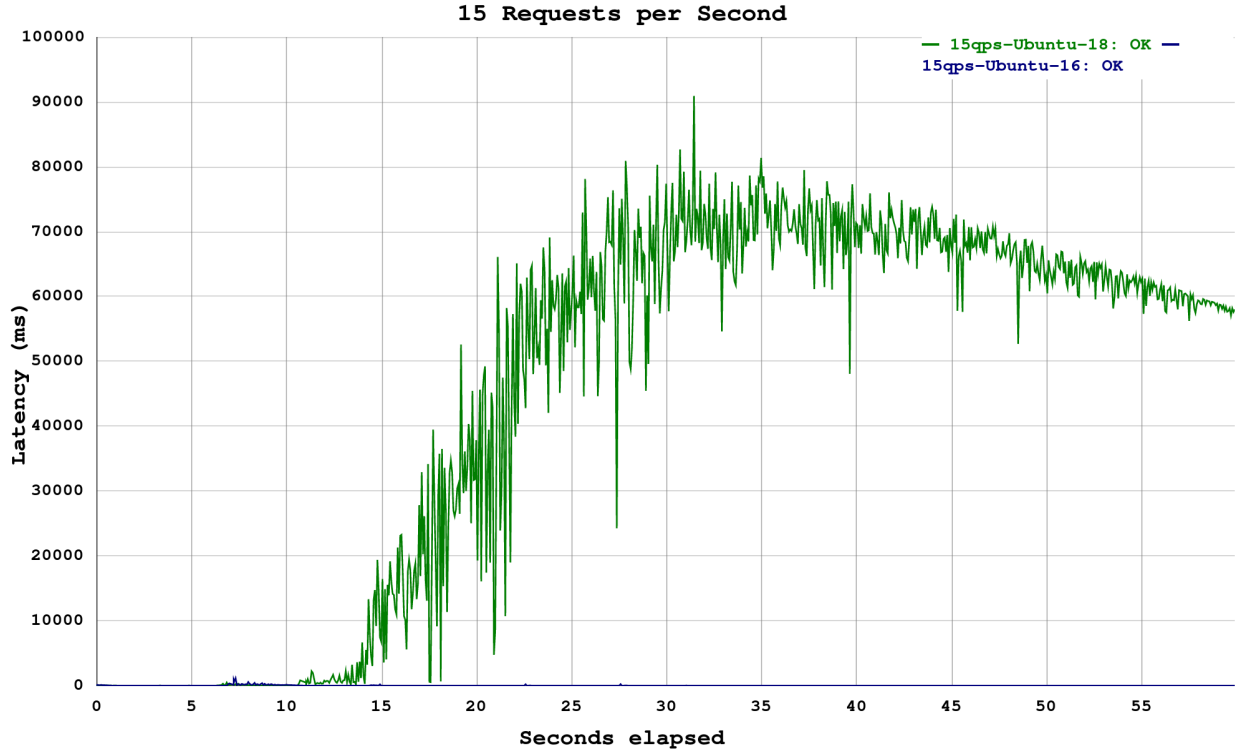
<sup>2</sup>HAProxy, Available: <http://www.haproxy.org/>

Balancer reached 15 requests per second without degradation. Two more requests translate to a 15% increase of available bandwidth compared to the other Load Balancer, which is quite significant in a load balancing environment.

Figure 6.4 presents a comparison between HAProxy on Ubuntu 18.04 and Ubuntu 16.04 with 13 requests per second of a 7.2MB chunk, showing that neither of the Load Balancers throttles. The plot on figure 6.5 shows a different behavior: when Vegeta makes 15 requests per second of the same chunk, the Load Balancer on Ubuntu 18.04 falls apart, reaching a peak of 100 seconds to fulfill the requests. On this plot, it is a little difficult to see the Ubuntu 16.04 Load Balancer corresponding line because the maximum delay is around 200 ms and the other Load Balancer is 90000 ms.



**Figure 6.4:** Vegeta 13 requests comparison, on HAProxy with Ubuntu 16 and 18



**Figure 6.5:** Vegeta 15 requests comparison, on HAProxy with Ubuntu 16 and 18

On the subject of bandwidth limitations, sometimes virtual machines can display network performance above the Virtual Ethernet Adapter specification due to hypervisor techniques to improve throughput between virtual machines on the same server. Thus, an *iPerf*<sup>3</sup> test was performed throughout the machines to test if the Proxmox hypervisor was respecting the virtual machines network limits, which it was shown to be true.

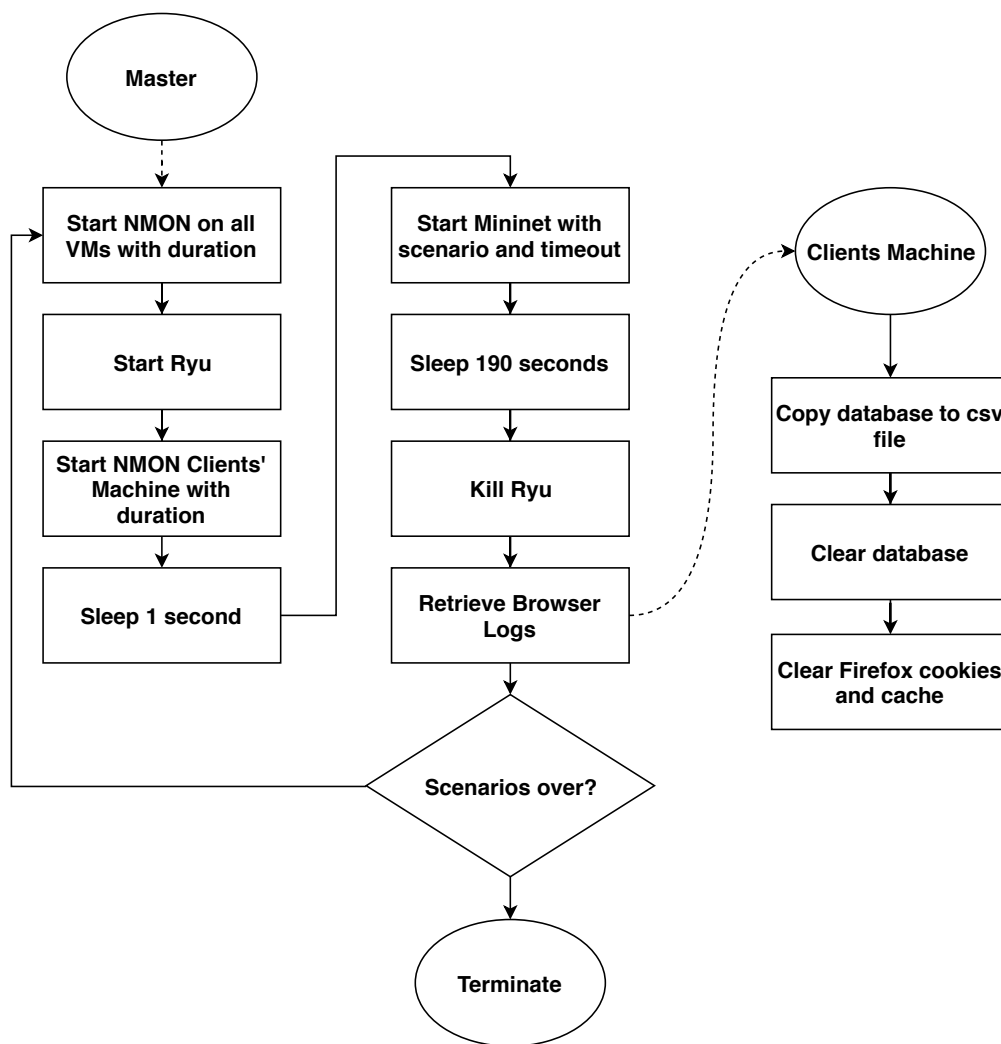
With everything set up, the missing piece is how to retrieve data to compare the approaches. As already discussed throughout this work, the most critical metric is the user perceived QoE, and here it is no exception. To compare the baseline infrastructure performance against the proposed infrastructure, the clients' QoE is analyzed. Firefox was chosen as the clients' browser because it comes installed on most of the Linux distributions, working with Mininet out-of-the-box, and offers greater flexibility in developer tools when compared to Google Chrome. Every client has its own network namespace and its own Firefox profile, creating a perfect sandbox experience, allowing multiple clients to run in a single machine perfectly isolated. The only issue was to run clients in private mode to discard any cached content that would influence subsequent testing.

The recording of clients' QoE brought several challenges. The most straightforward way to record this information was to run a REST application, where the clients would send their QoE values. Nevertheless, this method introduces unnecessary network overhead. Therefore, the approach was to each clients' Firefox to save the data. The only correct way for a browser to store data on a client is through the *localStorage* property [73] that writes to a special

<sup>3</sup>iPerf, Available: <https://iperf.fr/iperf-doc.php>

database. However, to guarantee independent testing, the clients' Firefox was running in private mode to disable any caching, also disabling any writing to the localStorage.

Several Firefox options were tested to run in normal mode without caching any content resulting in failed attempts. For instance, disabling cache alone would not work because cookies influenced the Dash.JS requested quality at the start. Cookies and localStorage are stored in different databases, but disabling cookies for some reason disables all Firefox database writing. Thus, a script deletes the right files, cookies, and cache avoiding breaking a Firefox profile, retrieves the content of the *webappsstore.sqlite* (the localStorage database), cleaning at the end. To improve the insight into the clients' environment, other values besides QoE were recorded too: FPS, bitrate, video buffer size, client resolution, and video resolution. To record metrics other than video related, the testbed uses NMON in every machine. IBM originally developed the tool for its proprietary operating system, later converting to Linux operating systems. NMON records information about CPU utilization, memory usage, disk read and write transfers, and network usage. To perform the scenarios proposed, a separate computer orchestrates all of the necessary instructions in a life-cycle illustrated on figure 6.6.



**Figure 6.6:** Scenarios Orchestration Overview

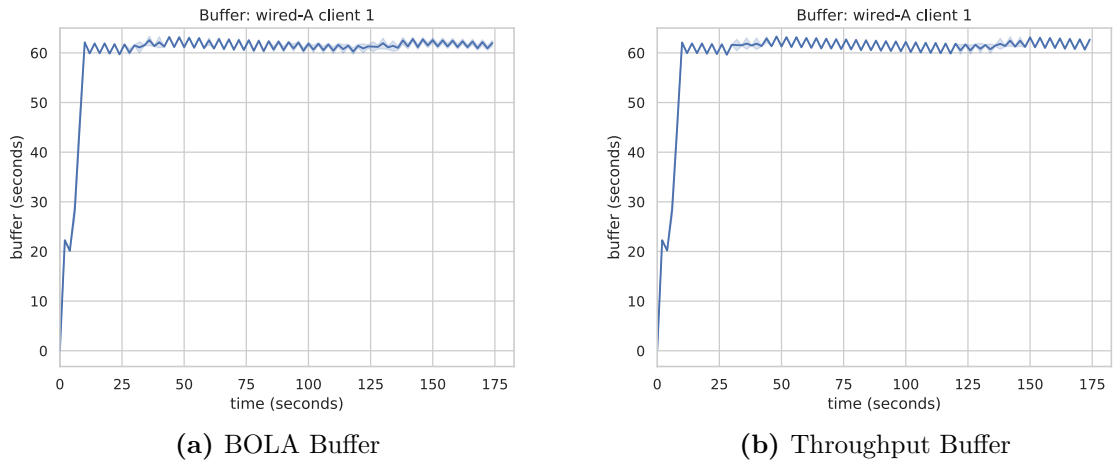
## 6.3 Evaluation

The last section of this chapter will focus on the performance evaluation of the proposed content delivery approach, comparing to the baseline. Still, before the main evaluation, it is important to evaluate the ABR algorithms offered by Dash.JS, because of the QoE impact that it may have.

### 6.3.1 ABR Algorithms Evaluation

This type of algorithms are critical for the success of an HTTP adaptive streaming technology. They are the decision makers of what chunks to request depending on the clients' environment, like display size, CPU loads or network conditions, to give the best quality possible avoiding re-buffering. The ABR algorithms continuously monitor bandwidth and CPU load to adjust to any client changes to request the most adequate stream. Therefore, these algorithms aim to stream at the highest bitrate while minimizing re-rebuffering events and excessive bitrate switches. Dash.JS comes with three ABR options: Throughput, BOLA, and Dynamic. The first strategy is the most simple, chooses bitrate on recent throughput. BOLA is a more advanced strategy [74], choosing higher-bitrate segment as the buffer grows, and the opposite, choosing lower-bitrate segments as buffer occupation depletes. The Dynamic strategy is a combination of the first two. Therefore, a performance test between the three was done with the proposed scenarios to throttle clients.

The comparison will center on BOLA and Throughput to check for any major differences. The Big Buck Bunny <sup>4</sup> video runs for 3 minutes, 10 times for each test. On a best-case scenario like the Wired scenario (bw=100, d=1) and sub-type A (no further throttle), the buffer occupancy throughout the 3-minute testing is identical, as figure 6.7 shows.

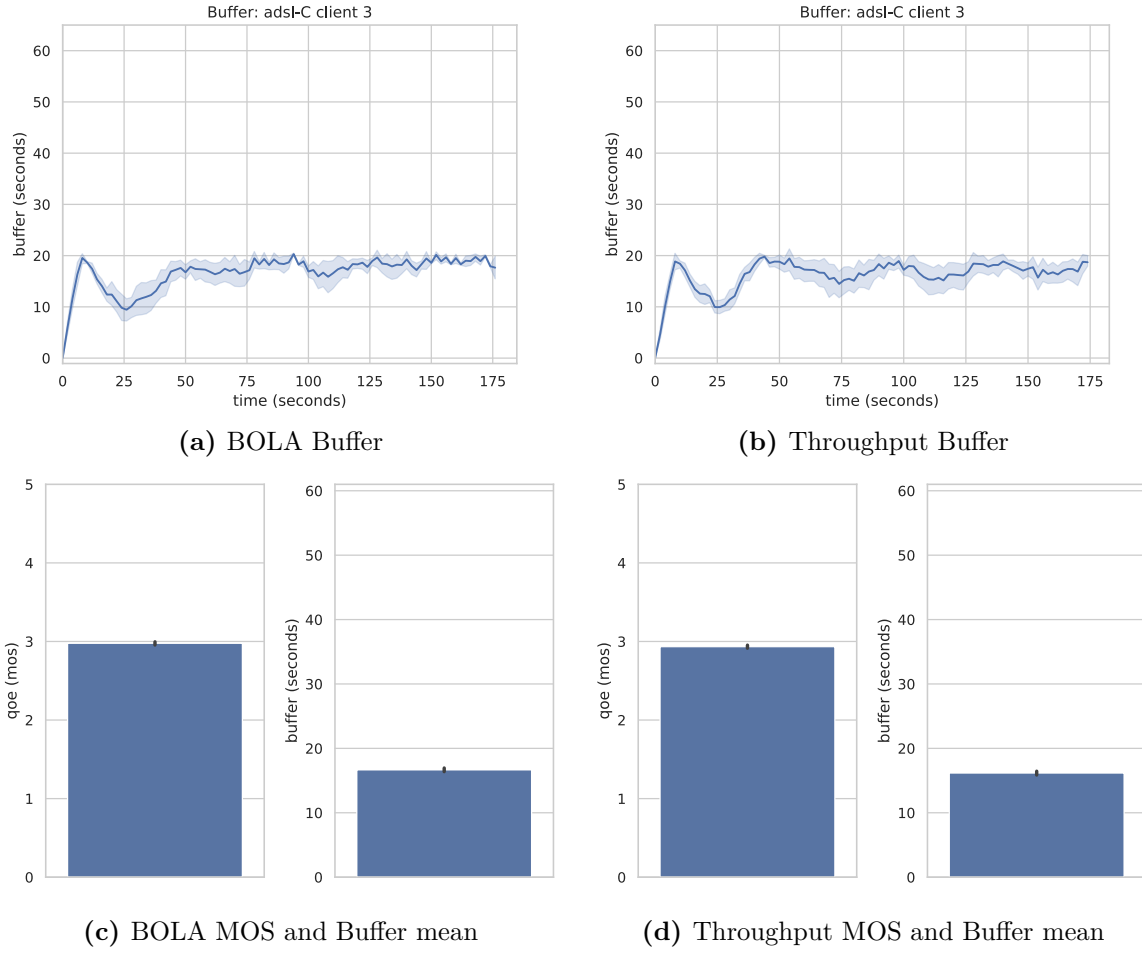


**Figure 6.7:** Wired scenario sub-type A between BOLA and Throughput (ci=95)

In other scenarios with packet loss, the results are similar. Figure 6.8 illustrates an ADSL scenario (table 6.1), sub-type C (table 6.2) with buffer occupancy through the 3-minute testing

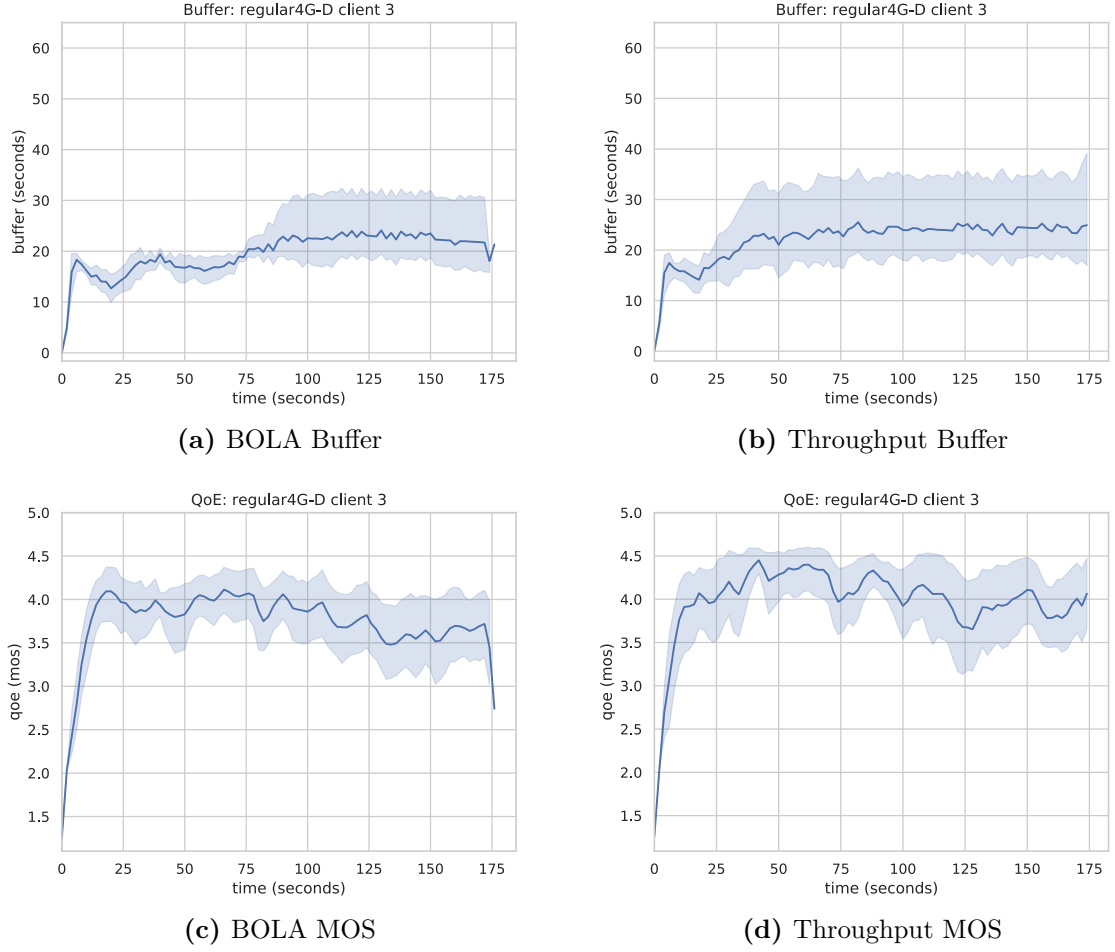
<sup>4</sup>Big Buck Bunny, Available: <https://peach.blender.org/>

as well as the mean for QoE and buffer, to give a broader overview.



**Figure 6.8:** ADSL scenario sub-type C between BOLA and Throughput (ci=95)

In scenarios with more packet delay and no packet loss, Throughput strategy is more aggressive on buffer occupation, translating into slightly higher QoE values. This trend is noticeable in several scenarios like the one in figure 6.9 that illustrates a Regular 4G scenario (table 6.1), sub-type D (table 6.2), where QoE is always slightly better.



**Figure 6.9:** 4G scenario sub-type D between BOLA and Throughput (ci=95)

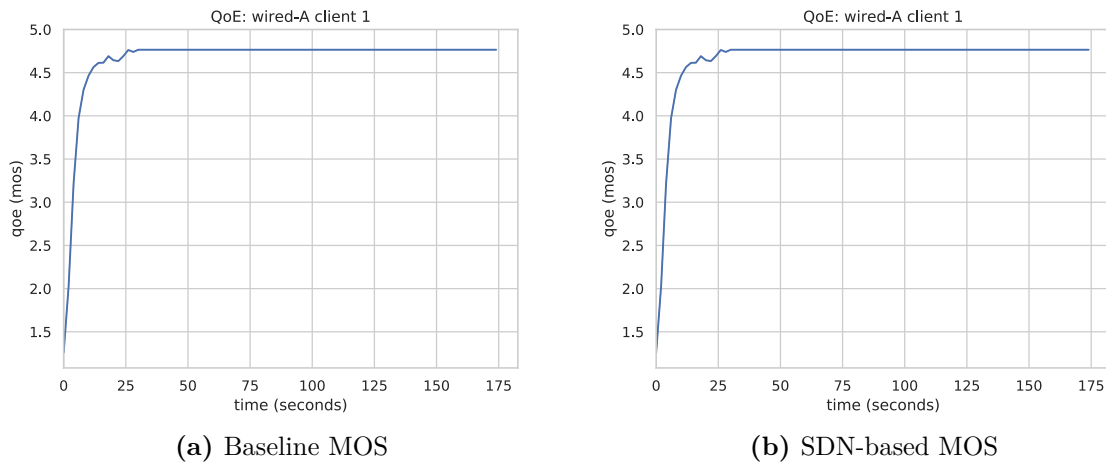
Through empirical analysis, it is shown that the Dynamic strategy takes the best of the two approaches: Throughput strategy is a little bit more aggressive at the start, where there is no past data to learn from, but as time goes, the BOLA strategy can produce a more weighted decision on what bitrate to choose to increase the video experience. Thus, for the following performing tests comparing the baseline and the proposed approach, Dash.JS ABR strategy is set to Dynamic.

### 6.3.2 Evaluation of the Proposed Content Delivery Approach

The performance tests follow the scenarios with sub-types and infrastructure throttling (divided into three parts) as previously discussed. 10 tests are performed on every sub-type of a scenario, for confidence interval purposes. Each test is a 3-minute run of the Big Buck Bunny, a standard for video testing. The maximum quality of the stream is Full HD (FHD), running on a screen with the same resolution. The maximum resolution chosen was not 4K, due to performance issues of running more than two clients. Mininet virtualizes four hosts, so the three Origins are always unbalanced. From this point forward, the section will be divided into the different parts of infrastructure throttling.

## Part 1

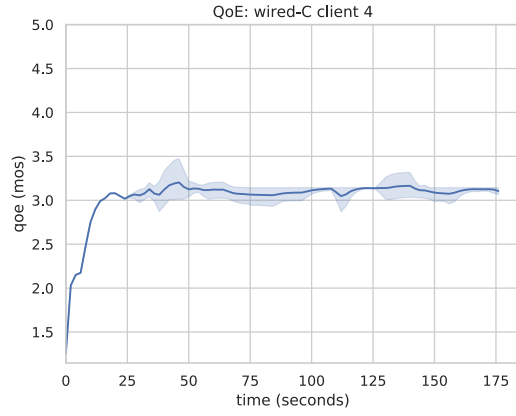
In part 1, only the clients suffer from throttling, like a best-case scenario for the infrastructure. For the first comparison, the Wired scenario with sub-type A is chosen to present the best-case scenario of all. Figure 6.10 shows a QoE comparison of the 3-minute run. As expected, both perform at the maximum, staying at a 4.7 MOS value, the probe's maximum value for a FHD stream in a screen of the same resolution. Between the 20 seconds and the 30-seconds mark, there is a small hiccup produced by the QoE probe, more specifically by the human memory filter technique. In the probe, the last 30 seconds of events are weighted and combined into the final MOS value, but when a video starts, there is no past information. That is why the plot has a small disturbance right before the 30 seconds mark, and after that it stabilizes.



**Figure 6.10:** Wired scenario sub-type A between Baseline and SDN-based infrastructure, part 1 (ci=95)

However, for every scenario with further throttling, sub-type B to I, the proposed method is always a little behind on MOS score. Figure 6.11 depicts a test with Wired scenario and sub-type C (26 ms, 100 Mbps, 25 ms jitter, 2.5% loss), with a QoE and buffer comparison.

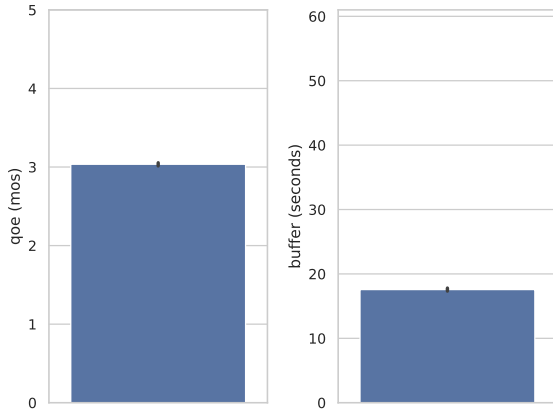
Comparing with a more demanding scenario, ADSL with sub-type D (120 ms delay, 24 Mbps, 100 ms jitter and 0% loss), the trend repeats as illustrated in figure 6.12 with MOS and buffer average comparison. It is also interesting to observe the sub-type C performance with a Wired scenario when comparing with the sub-type D with ADSL scenario, where it is noticeable that packet loss has a much greater impact than delay.



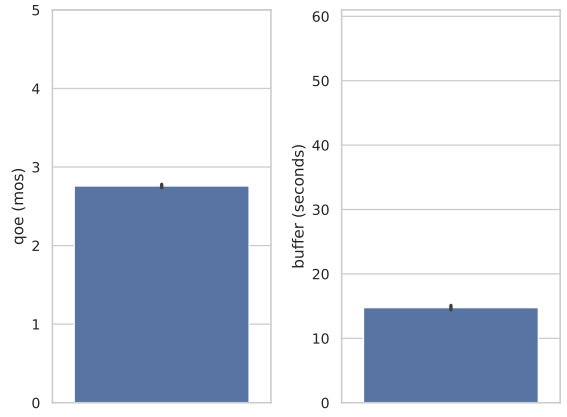
(a) Baseline MOS



(b) SDN-based MOS

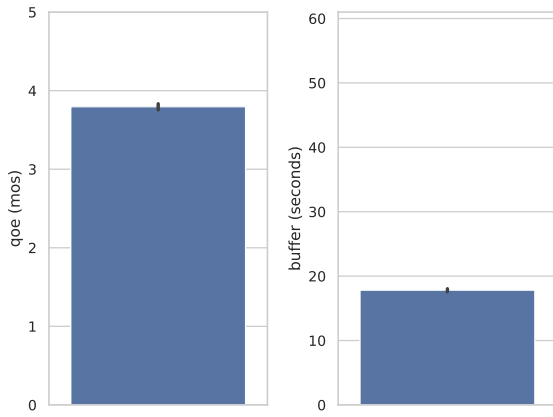


(c) Baseline MOS and Buffer mean

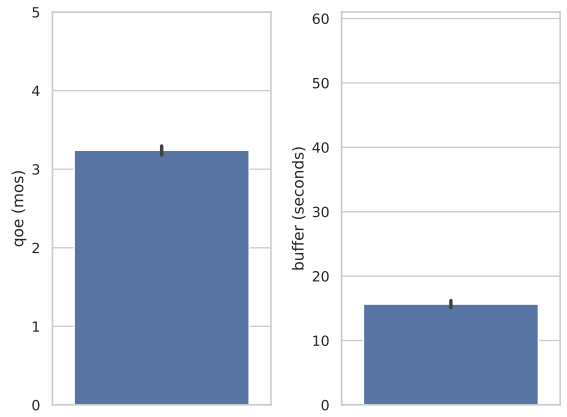


(d) SDN-based MOS and Buffer mean

**Figure 6.11:** Wired scenario sub-type C between Baseline and SDN-based infrastructure, part 1 (ci=95)



(a) Baseline average MOS and Buffer

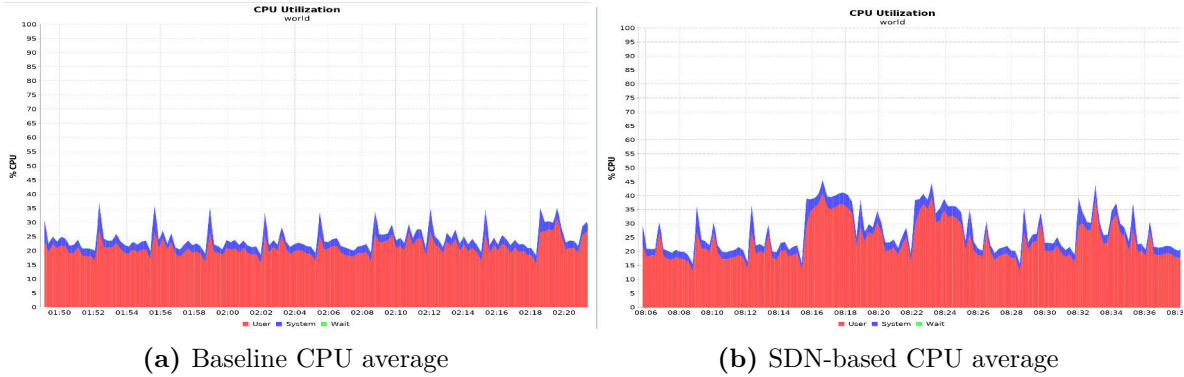


(b) SDN-based average MOS and Buffer

**Figure 6.12:** ADSL scenario sub-type D between Baseline and SDN-based infrastructure, part 1 (ci=95)

To further analyze why the proposed method is not performing similarly to the Baseline,

figure 6.13 shows a comparison of CPU utilization throughout the 10 runs on the Wired-A scenario, a scenario in which the same bitrate streams are processed to produce a fair comparison. The proposed approach has a little higher average on CPU utilization (3% more), not enough to assume anything. Other statistics produced by NMON are inconclusive as well. However, the most plausible cause is the operation of packet header modification done by the virtual switch, that adds just enough delay to affect the clients' player.

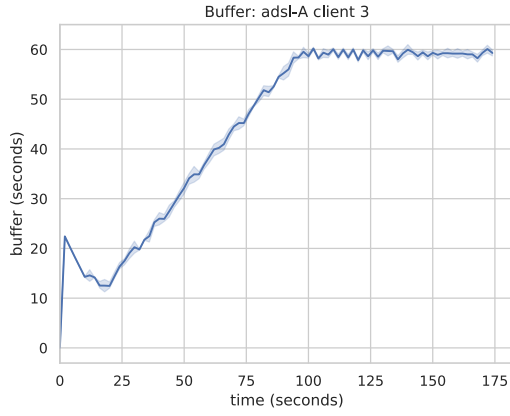


**Figure 6.13:** Average CPU of the Wired scenario, sub-type A, part 1

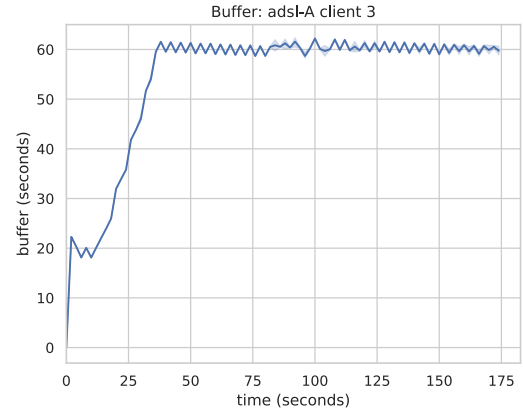
## Part 2

For part 2, the overall infrastructure is throttled with Vegeta, running 13 requests (7.2MB chunk) per second on the Load Balancer. As already discussed, on the SDN infrastructure, Vegeta uses the Load Balancer from the Baseline to stress the Origins equally. With 13 requests, the infrastructure is almost at full capacity.

Comparing ADSL-A scenario (24 Mbps and 5 ms delay) in figure 6.14, it is obvious the impact on buffer occupancy, although QoE is not affected. Looking to a Wired-B scenario (100 Mbps, 26 ms delay, and 25 ms jitter), the low buffer size starts to influence the MOS value. Even with more bandwidth, the added delay on this scenario takes a toll on the Baseline infrastructure, as illustrated in figure 6.15, and the overall QoE is higher on the SDN approach.

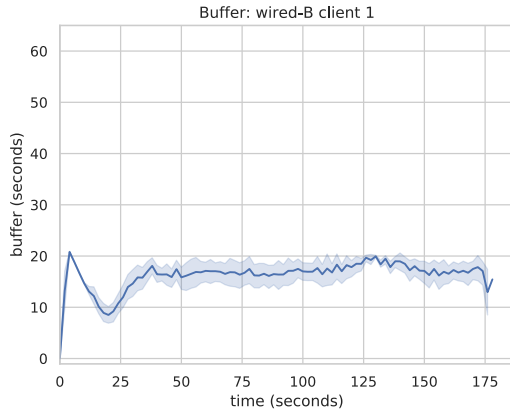


(a) Baseline Buffer

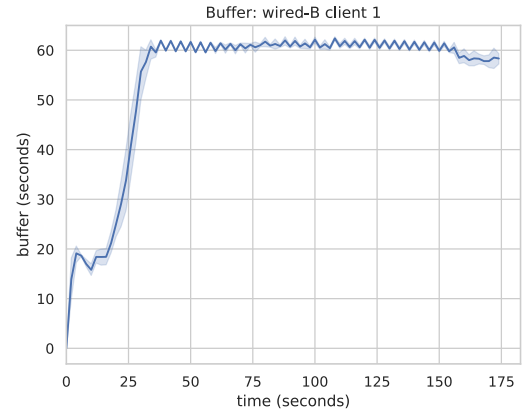


(b) SDN-based Buffer

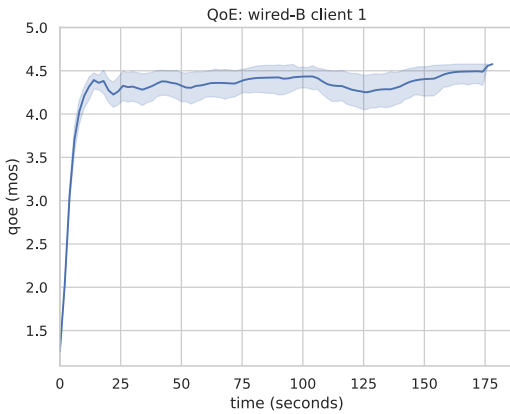
**Figure 6.14:** Buffer comparison of the ADSL-A scenario, part 2 ( $ci=95$ )



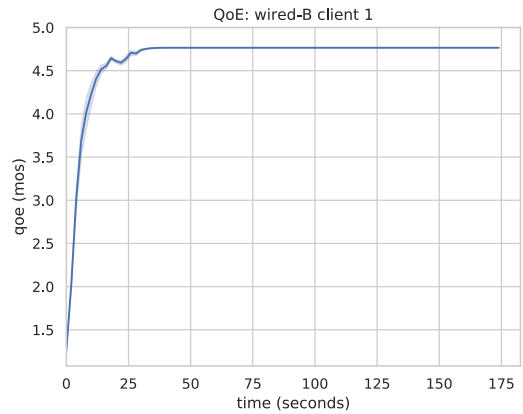
(a) Baseline Buffer



(b) SDN-based Buffer



(c) Baseline MOS

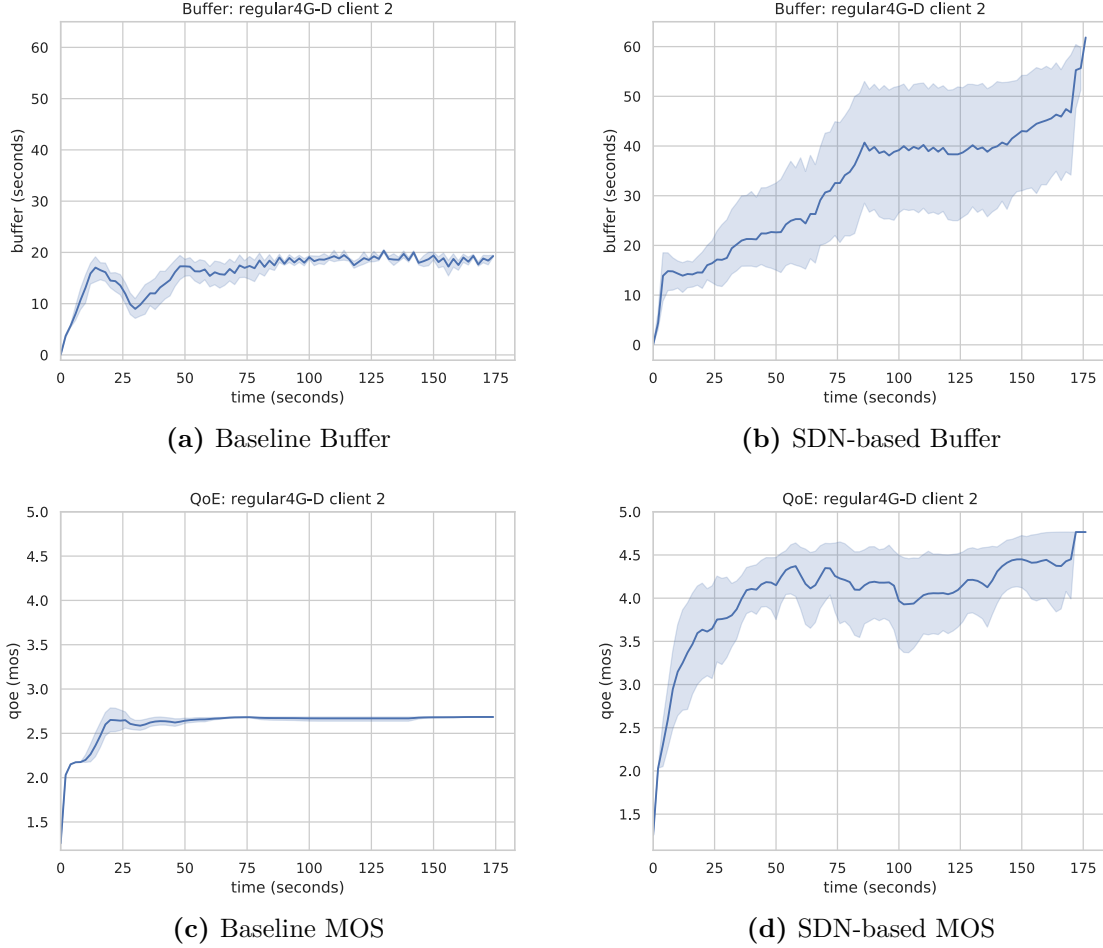


(d) SDN-based MOS

**Figure 6.15:** Buffer and QoE comparison of the Wired-B scenario, part 2 ( $ci=95$ )

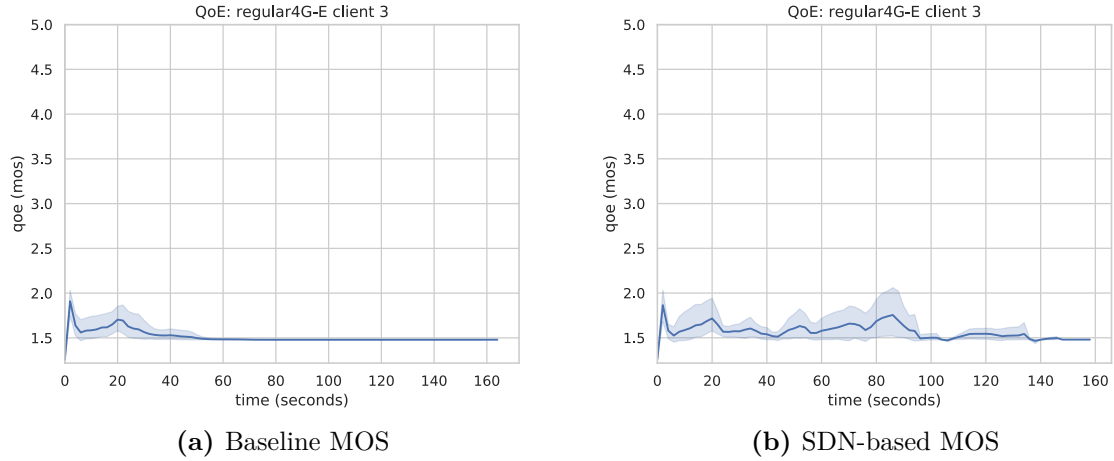
Moving to a more demanding scenario, Regular 4G-D (32 Mbps, 120 ms delay, 100 ms jitter, and 0% loss), the Baseline infrastructure is more affected than the SDN-based. Figure 6.16 depicts a buffer and a MOS comparison of the Regular 4G-D scenario, where on the

Baseline infrastructure, the client's player struggles to fill up the buffer, failing the switch to a higher bitrate stream, resulting in a much lower quality-of-experience compared to the proposed approach.



**Figure 6.16:** Buffer and QoE comparison of the Regular 4G-D scenario, part 2 (ci=95)

However, when packet loss is added to the scenario, there is no possibility to recover the QoE. Figure 6.17 exhibits a MOS comparison with the Regular 4G-E scenario that adds 5% packet loss to the latter scenario discussed. Here, both of the videos stopped just a little bit before the test was completed; still, the proposed approach has a small edge on the average MOS compared to the Baseline infrastructure.

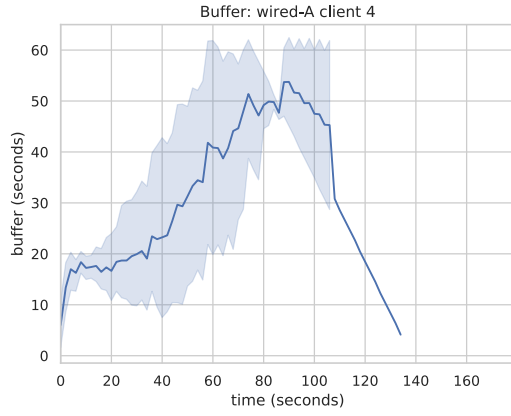


**Figure 6.17:** MOS comparison of the Regular 4G-E scenario, part 2 (ci=95)

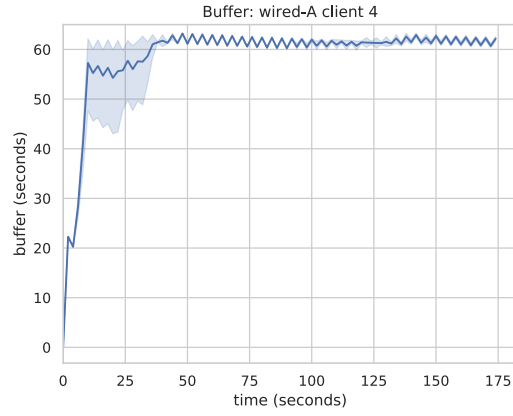
### Part 3

In the last set of performance tests, part 3, instead of throttling the overall infrastructure, only one Origin is throttled, depicting a situation where an Origin is running other services or becomes unresponsive due to some problem. Vegeta performs the same load creation with 13 requests per second, nearly topping the Gigabit connection of the Origin.

Choosing a client's best-case scenario, the Wired-A scenario (100 Mbps, 1 ms delay), figure 6.18 compares buffer and MOS value of the Baseline infrastructure and proposed method. The Baseline client's player does not finish the test while the SDN method finishes. This trend is observable on the Baseline infrastructure throughout every scenario: in every test the player stops after 40-140 seconds of playback. If in the Orings' pool one server becomes throttled or unresponsive, a domino effect happens, hurting every client. On the SDN proposed approach, the MOS plot presents a bigger variation before the 30 seconds mark. Examining the buffer plot, there is also a variation but starting sooner; this could represent the controller changing the client's Origin. When a player starts, a burst of requests for the smallest quality available is performed, and the throttled Origin can provide such small chunks. However, right after the player starts switching to a higher bitrate, the bad Origin will underperform, triggering an Origin switch right at the start, impacting a little later the normal instability at the 20-second mark.



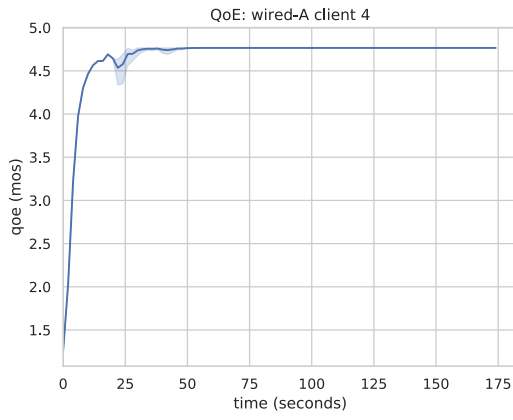
(a) Baseline Buffer



(b) SDN-based Buffer



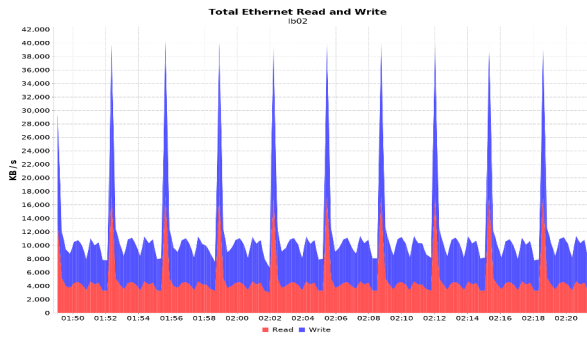
(c) Baseline MOS



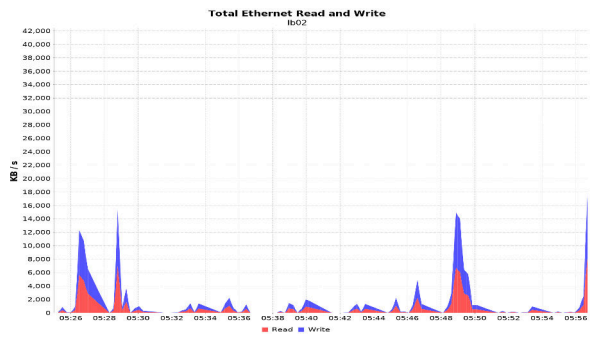
(d) SDN-based MOS

**Figure 6.18:** Buffer and QoE comparison of the Wired-A scenario, part 3 (ci=95)

To further observe the domino effect on the Baseline Load Balancer, figure 6.19 illustrates the total network read and write of the Load Balancer in the Wired-A scenario (10 consecutive runs), between part 1 and part 3 testing. As expected, the plot from the part 3 testing shows a much lower activity, demonstrating the hazardous effect of one Origin affected; the network write average is  $\approx 7\%$  times higher on part 1 testing.



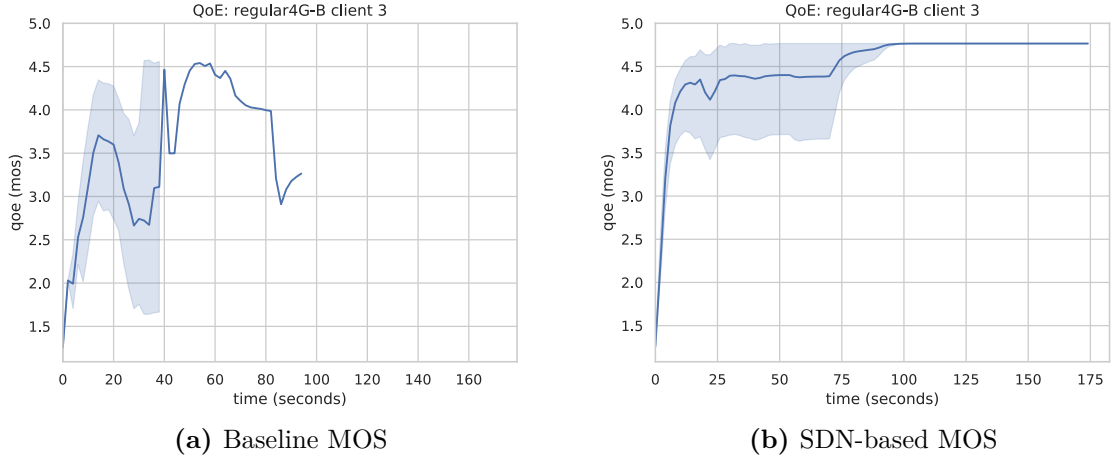
(a) Part 1



(b) Part 3

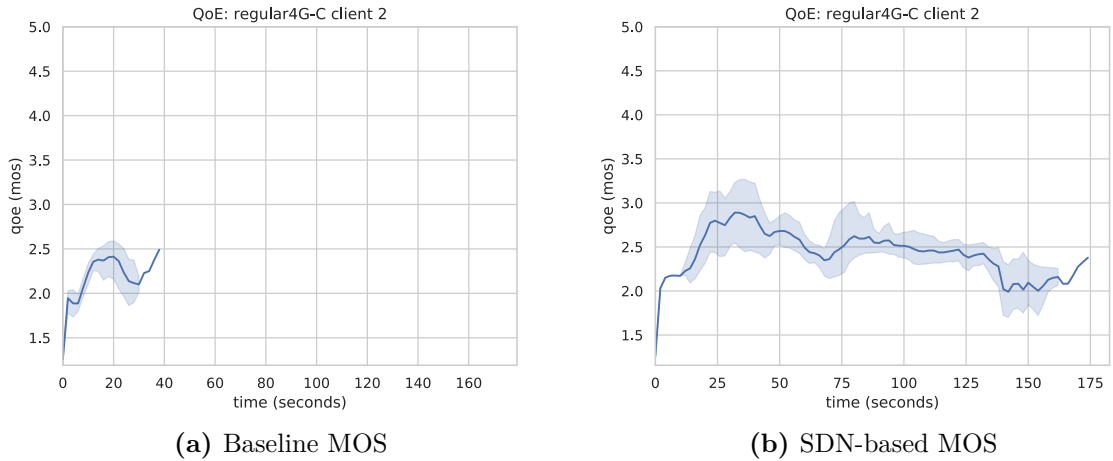
**Figure 6.19:** Total network write and read of the Wired-A scenario, with part 1 and part 3 testing

Figure 6.20 illustrates a comparison on the regular 4G-B (32 Mbps, 45 ms delay, 25 ms jitter, 0% loss). The Baseline infrastructure player stops after  $\approx 90$  seconds, repeating the trend, while the proposed method endures and reaches a good MOS number. However, the SDN related plot shows a higher variation on the first half, meaning that throughout the 10 runs, the controller sometimes gave at the start the bad Origin to the third client, later forcing a live TCP change.



**Figure 6.20:** QoE comparison of the Regular 4G-B scenario, part 3 (ci=95)

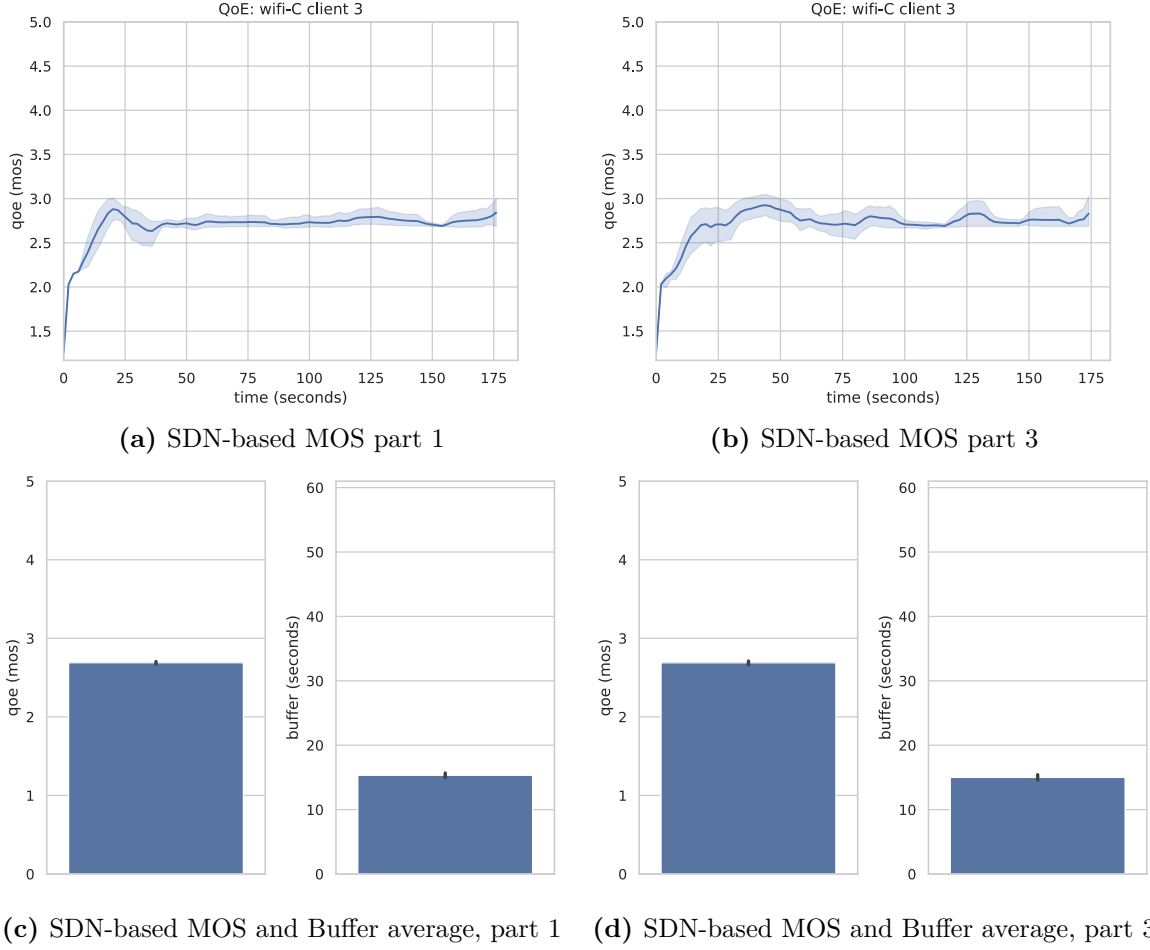
Moving to the next scenario that adds 2.5% packet loss, regular 4G-C, figure 6.21 illustrates another QoE comparison. For the Baseline infrastructure, the player stops just after 40 seconds, while the proposed approach has a similar performance under part 1 tests.



**Figure 6.21:** QoE comparison of the Regular 4G-C scenario, part 3 (ci=95)

The last figure of the section, figure 6.22, compares the SDN approach with a scenario Wifi-C (50 Mbps, 26 ms delay, 25 ms jitter, and 2.5% loss) from the part 1 and part 3. Although it is possible to observe more variation from part 3 testing, the average MOS is the

same, as well as the buffer size, showing that the controller is able to avoid the bad Origin with only data from the client.



**Figure 6.22:** QoE, average QoE, and Buffer comparison between Wifi-C part 1 and part 3 on SDN infrastructure (ci=95)

## 6.4 Summary

This chapter started with the elaboration of scenarios to test the proposed approach, which divides into three parts. Next, the Testbed section described the hardware and software specifications to simulate the infrastructure and run the scenarios. It outlined as well the challenges on software integration to provide an accurate simulation and data retrieval. Lastly, the Evaluation section described the most relevant scenarios results, comparing the Baseline infrastructure with the SDN infrastructure proposed; in part 1 tests, Baseline performed better, while in part 2 and 3, the proposed approach outperformed the Baseline.

## Conclusions and Future Work

Video consumption is a rising trend, creating market value and new opportunities, but not without its challenges, as discussed in chapter 2. Global companies are challenging Telecommunication Operators infrastructure with their OTT platforms. They create revenue without investing in the last mile infrastructure that connects the client to the Internet, challenging the Telcos business model. However, Telcos are in the best position to create their own OTT platforms, at the same time improving their content delivery infrastructure to accommodate future demand, allowing them to stay competitive against major companies. The current network hardware does not offer the flexibility and innovation needed for the content delivery infrastructure of tomorrow. Deployment of new network hardware is a must, and Netflix is the perfect example. They offer on-demand content, that is unicast traffic, to millions of users. If they relied only on a classic CDN structure, they would not be able to support so much downlink traffic. Establishing protocols with Telcos, they deploy hardware near the last mile network, bringing the content near the clients, like a small and more distributed CDN. Although expensive, bringing hardware closer to the client is a must.

All of the work produced for this dissertation leads to the proposed content delivery approach, an SDN load balancing solution based on clients' perceived QoE with Dash.JS as the clients' player.

The proposed forecast model addresses the issue of multicast and unicast transmissions. Two approaches are developed as an alternative to the traditional way of delivering TV-channels and on-demand content: Unicast-only and Hybrid. The scenarios' results show that for a medium scale scenario, Unicast-only proves to be an option, based on the weight-based OPEX estimates; for a large scale scenario, with hundreds of channels and millions of subscribers, the Hybrid approach is considered to be a viable prospect. Furthermore, estimations show that only appliances at the SR are not sufficient to build an alternative method to the traditional multicast approach. Only when appliances are considered at both the SR and the OLT, the results point to a future-proof infrastructure alternative. Therefore, the proposed forecast model shows that, for a real improvement of the established network

infrastructure, Telcos must consider new hardware on the edge of the network.

The results of this model motivated the proposal of the load balancing approach. As seen in chapter 6, Dash.JS can withstand a great deal of packet delay, a problem that affected other streaming technologies discussed in the state of the art chapter. However, packet loss is an issue that affects all HTTP adaptive streaming technologies, and Dash.JS is no exception. Above 5% of packet loss, it is not possible to deliver content with a good experience. The results have shown that playback stops after some time with that type of loss. Then, ABR strategies were put to the test to help decide which one would be used in Dash.JS to enhance the overall performance. As discussed in chapter 6, Throughput strategy performs better at the start when compared to the BOLA strategy, but with time and the usage of past knowledge, BOLA performs better. Thus, the Dynamic strategy that employs the best of both strategies was chosen for the performance tests on the proposed content delivery method and Baseline infrastructure.

The comparison experiments, between the baseline and the proposed approach, were divided into three infrastructure throttling parts. Part 1 does not throttle the infrastructure, only the clients are throttled with the scenarios. The proposed approach underperforms under these circumstances. For every packet in the bidirectional communication between a client and an Origin, two headers are always changed: Ethernet and IP addresses, corresponding to the destination or source field depending on the flow. While on the Baseline simulation, only port forwarding happens. Specific tests were done to identify if the virtualized switch, by changing packet headers, was adding some delay that would explain this performance issue. The tests performed were inconclusive, and as of this writing, the problem is yet to be determined. However, the most plausible cause is the operation of packet header modification done by the virtual switch, that adds just enough delay to affect the clients' player.

In part 2 of the performance tests, the overall infrastructure is stressed. Here, the proposed approach outperforms the traditional method because of one key aspect: the distribution of load decision. The Load Balancer on the Baseline infrastructure is, in fact, a single point of failure, since every request has to go through that machine, while the SDN approach makes the decision at the edge of the network, removing the need for a Load Balancer to sit between the client and the Origin.

The last part of the performance tests, Part 3, has one Origin throttled. Since the Load Balancer distributes requests in a round-robin way, a domino effect happens. At the start, the bad Origin will be able to deliver the small chunks requested; after that, the Dash.JS will switch to a higher bitrate, and those chunks will not be delivered. The clients' player will receive some packets fast (by the other good Origins), but the throttled Origin will fail to deliver the rest, producing mix signals to the ABR strategy, sometimes chunks arrive fast, and sometimes they do not arrive. Further, packet loss will increase, and the video playback will stop. However, the controller from the proposed approach is continuously analyzing the clients' QoE, taking action when needed, live changing a TCP session to another Origin, protecting the clients from infrastructural content delivery problems. Thus, the performance results of the proposed approach for part 1 and part 3 are similar, proving the effectiveness of

the solution.

However, there is room for improvement: fixing known issues and introducing new features. In the future, it is essential to fix the performance issues detected in the tests of part 1, to bring the proposed solution to the same performance level of the Baseline infrastructure in that scenario. Also, it is necessary the development of a controller's logging system to account the clients' changes: with data of when the client switched and to which Origin, and the number of changes. Another valuable improvement in the controller would be to implement a way to detect a page refresh of a known client, to clean the client's information because occasionally the Dash.JS player breaks. Moreover, Dash.JS allows the use of a custom ABR strategy, and it would be interesting to implement a feature that would fit the client into a scenario that would have a set of specific ABR rules, like a mobility scenario (cellular) and a static scenario (wired) would have custom strategies to improve QoE. More, to take advantage of the Origin's information that the controller inferred from the clients' QoE, the implementation of a mechanism to inform a monitoring application of the Telco infrastructure would improve the value of the proposed work.

Lastly, an ambitious feature, a more demanding and time-consuming one but considering the results of the forecast model, would be the development of a mechanism that would create a simulated multicast stream from an unicast stream. One client is watching a live-TV channel on an HTTP player and a consecutive client to request the same live-TV channel, the stream would be duplicated from the first client, this happening at the network edge. Thus, this would obtain the benefits of an HTTP streaming solution and the gains of a multicast approach, something not yet done in both the academic and the private sector.



# References

- [1] Cisco, “Cisco Visual Networking Index: Forecast and Methodology, 2016–2021”, *Cisco Systems*, 2017. [Online]. Available: <https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.pdf>.
- [2] E. Consumerlab, “TV and Media 2017 a consumer-driven future of Media”, *TV and Media ConsumerLab*, 2017. [Online]. Available: [https://www.ericsson.com/assets/local/networked-society/consumerlab/reports/tv%7B%5C\\_%7Dmedia%7B%5C\\_%7D2017%7B%5C\\_%7Dglobal%7B%5C\\_%7Dpresentation.pdf](https://www.ericsson.com/assets/local/networked-society/consumerlab/reports/tv%7B%5C_%7Dmedia%7B%5C_%7D2017%7B%5C_%7Dglobal%7B%5C_%7Dpresentation.pdf).
- [3] T. Kratochvil, “From analog to Digital Television — the common way how to digitize European broadcasting”, in *2008 IEEE History of Telecommunications Conference*, IEEE, Sep. 2008, pp. 164–169, ISBN: 978-1-4244-2530-3. DOI: 10.1109/HISTELCON.2008.4668734. [Online]. Available: <http://ieeexplore.ieee.org/document/4668734/>.
- [4] DVB Worldwide, *DVB History*, 2018. [Online]. Available: <https://www.dvb.org/about/history> (visited on 11/20/2018).
- [5] Anacom, *Anacom DVB-T*, 1998. [Online]. Available: <https://www.anacom.pt/render.jsp?categoryId=342919> (visited on 11/20/2018).
- [6] DVB Worldwide, “DVB-T2”, Tech. Rep., 2018. [Online]. Available: [www.dvb.org/standards](http://www.dvb.org/standards).
- [7] —, “DVB-S2”, Tech. Rep., 2018. [Online]. Available: [www.dvb.org/standards](http://www.dvb.org/standards).
- [8] —, “DVB-C2”, Tech. Rep., 2012. [Online]. Available: [www.dvb.org/standards](http://www.dvb.org/standards).
- [9] Accenture, “The Future of Broadcasting V: The Fundamental Growth”, Tech. Rep., 2016. [Online]. Available: <https://www.accenture.com/us-en/insight-future-broadcasting-search-fundamental-growth.aspx>.
- [10] T. I. Society, *Real Time Streaming Protocol (RTSP)*, 1998. [Online]. Available: <https://tools.ietf.org/html/rfc2326>.
- [11] Unisphere Research, “OTT Video Services, Innovation, Opportunity, Maturation & Technology trends in OTT delivery”, Tech. Rep., 2017. [Online]. Available: [https://www.denverpost.com/wp-content/uploads/2017/04/ott%7B%5C\\_%7Dvideosurv17%7B%5C\\_%7Dc12.pdf](https://www.denverpost.com/wp-content/uploads/2017/04/ott%7B%5C_%7Dvideosurv17%7B%5C_%7Dc12.pdf).
- [12] Christopher Mueller, *MPEG-DASH Dynamic Adaptive Streaming over HTTP*, 2015. [Online]. Available: <https://bitmovin.com/dynamic-adaptive-streaming-http-mpeg-dash/> (visited on 11/21/2018).
- [13] A. Fox, S. Gribble, Y. Chawathe, and E. Brewer, “Adapting to network and client variation using infrastructural proxies: lessons and perspectives”, *IEEE Personal Communications*, vol. 5, no. 4, pp. 10–19, 1998, ISSN: 10709916. DOI: 10.1109/98.709365. [Online]. Available: <http://ieeexplore.ieee.org/document/709365/>.
- [14] Microsoft, *Microsoft Smooth Streaming*, 2009. [Online]. Available: <https://docs.microsoft.com/en-us/iis/media/smooth-streaming/smooth-streaming-transport-protocol> (visited on 11/20/2018).
- [15] Apple, “Apple HLS”, 2018. [Online]. Available: [https://developer.apple.com/documentation/http%7B%5C\\_%7Dlive%7B%5C\\_%7Dstreaming/hls%7B%5C\\_%7Dauthoring%7B%5C\\_%7Dspecification%7B%5C\\_%7Dfor%7B%5C\\_%7Dapple%7B%5C\\_%7Ddevices](https://developer.apple.com/documentation/http%7B%5C_%7Dlive%7B%5C_%7Dstreaming/hls%7B%5C_%7Dauthoring%7B%5C_%7Dspecification%7B%5C_%7Dfor%7B%5C_%7Dapple%7B%5C_%7Ddevices).
- [16] DASH-IF, *DASH-IF About*. [Online]. Available: <https://dashif.org/about/> (visited on 11/20/2018).

- [17] —, *Dynamic adaptive streaming over HTTP (DASH) – Part 1*, 2014. [Online]. Available: <https://www.iso.org/standard/65274.html> (visited on 11/21/2018).
- [18] Sandvine, “The Global Internet Phenomena Report”, Tech. Rep., 2018. [Online]. Available: <https://www.sandvine.com/hubfs/downloads/phenomena/2018-phenomena-report.pdf>.
- [19] W3C, *Media Source Extensions<sup>TM</sup>*. [Online]. Available: <https://www.w3.org/TR/media-source/> (visited on 12/03/2018).
- [20] Dash-IF, *Home · Dash-Industry-Forum/dash.js Wiki · GitHub*. [Online]. Available: <https://github.com/Dash-Industry-Forum/dash.js/wiki> (visited on 12/03/2018).
- [21] Mozilla, *MediaSource - Web APIs | MDN*. [Online]. Available: <https://developer.mozilla.org/en-US/docs/Web/API/MediaSource%7B%5C%7DBrowser%7B%5C%7Dcompatibility> (visited on 12/03/2018).
- [22] Alexis Deveria, *Can I use MSE?* [Online]. Available: <https://caniuse.com/%7B%5C%7Dsearch=mse> (visited on 12/03/2018).
- [23] R. Mueller, “Modern cable TV network, the ideal platform for PCS”, in *Conference Proceedings National Telesystems Conference 1993*, IEEE, 1993, pp. 59–64, ISBN: 0-7803-1325-9. DOI: 10.1109/NTC.1993.293009. [Online]. Available: <http://ieeexplore.ieee.org/document/293009/>.
- [24] Telesail, *Two-way HFC Evolution solution based on GEON*. [Online]. Available: <http://www.telesail.com/solutions/ISP-Carriers-Service-Providers-24.html> (visited on 11/22/2018).
- [25] R. P. Leal, J. N. Marco, and F. Diego Hernandez Martinez, “Analysis of the technologies enabling the broadcast convergence”, in *2017 56th FITCE Congress*, IEEE, Sep. 2017, pp. 50–55, ISBN: 978-1-5386-0566-0. DOI: 10.1109/FITCE.2017.8093007. [Online]. Available: <http://ieeexplore.ieee.org/document/8093007/>.
- [26] A. Yarali and A. Cherry, “Internet Protocol Television (IPTV)”, in *TENCON 2005 - 2005 IEEE Region 10 Conference*, IEEE, Nov. 2005, pp. 1–6, ISBN: 0-7803-9312-0. DOI: 10.1109/TENCON.2005.300861. [Online]. Available: <http://ieeexplore.ieee.org/document/4084875/>.
- [27] S. Cherry, “The battle for broadband [Internet protocol television]”, *IEEE Spectrum*, vol. 42, no. 1, pp. 24–29, Jan. 2005, ISSN: 0018-9235. DOI: 10.1109/MSPEC.2005.1377870. [Online]. Available: <http://ieeexplore.ieee.org/document/1377870/>.
- [28] 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>.
- [29] Y. Bai, B. Jia, J. Zhang, and Q. Pu, “An Efficient Load Balancing Technology in CDN”, in *2009 Sixth International Conference on Fuzzy Systems and Knowledge Discovery*, IEEE, 2009, pp. 510–514, ISBN: 978-0-7695-3735-1. DOI: 10.1109/FSKD.2009.130. [Online]. Available: <http://ieeexplore.ieee.org/document/5360062/>.
- [30] 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>.
- [31] M. Fisher, *Comcast Netflix Agreement*, 2014. [Online]. Available: <https://qwilt.com/comcast-netflix-agreement-online-video-ecosystem/>.
- [32] S. Kasera, R. E. Miller, and M. Hofmann, “A profitable multicast business model”, *Computer Communications*, vol. 27, no. 13, pp. 1278–1287, Aug. 2004, ISSN: 0140-3664. DOI: 10.1016/J.COMCOM.2004.03.003. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0140366404001173>.
- [33] D. Kreutz, F. M. V. Ramos, P. Esteves Verissimo, C. Esteve Rothenberg, S. Azodolmolky, and S. Uhlig, “Software-Defined Networking: A Comprehensive Survey”, *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, Jan. 2015, ISSN: 0018-9219. DOI: 10.1109/JPROC.2014.2371999. [Online]. Available: <http://ieeexplore.ieee.org/document/6994333/>.

- [34] K. Greene, *TR10: Software-Defined Networking - MIT Technology Review*. [Online]. Available: <http://www2.technologyreview.com/news/412194/tr10-software-defined-networking/> (visited on 11/27/2018).
- [35] R. Masoudi and A. Ghaffari, "Software defined networks: A survey", *Journal of Network and Computer Applications*, vol. 67, pp. 1–25, May 2016, ISSN: 1084-8045. DOI: 10.1016/J.JNCA.2016.03.016. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1084804516300297>.
- [36] ONF, *SDN Technical Specifications / Open Networking Foundation*. [Online]. Available: <https://www.opennetworking.org/software-defined-standards/specifications/> (visited on 11/28/2018).
- [37] —, *OpenFlow Switch Specification v1.5.1*, 2015. [Online]. Available: <https://3vf60mmveq1g8vzn48q2o71a-wpengine.netdna-ssl.com/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf>.
- [38] L. L. Zulu, K. A. Ogudo, and P. O. Umenne, "Simulating Software Defined Networking Using Mininet to Optimize Host Communication in a Realistic Programmable Network", in *2018 International Conference on Advances in Big Data, Computing and Data Communication Systems (icABCD)*, IEEE, Aug. 2018, pp. 1–6, ISBN: 978-1-5386-3060-0. DOI: 10.1109/ICABCD.2018.8465433. [Online]. Available: <https://ieeexplore.ieee.org/document/8465433/>.
- [39] L. Mamushiane, A. Lysko, and S. Dlamini, "A comparative evaluation of the performance of popular SDN controllers", in *2018 Wireless Days (WD)*, IEEE, Apr. 2018, pp. 54–59, ISBN: 978-1-5386-5633-4. DOI: 10.1109/WD.2018.8361694. [Online]. Available: <https://ieeexplore.ieee.org/document/8361694/>.
- [40] R. Jawaharan, P. M. Mohan, T. Das, and M. Gurusamy, "Empirical Evaluation of SDN Controllers Using Mininet/Wireshark and Comparison with Cbench", in *2018 27th International Conference on Computer Communication and Networks (ICCCN)*, IEEE, Jul. 2018, pp. 1–2, ISBN: 978-1-5386-5156-8. DOI: 10.1109/ICCCN.2018.8487382. [Online]. Available: <https://ieeexplore.ieee.org/document/8487382/>.
- [41] M. Priyadarsini, P. Bera, and R. Bhampal, "Performance analysis of software defined network controller architecture—A simulation based survey", in *2017 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, IEEE, Mar. 2017, pp. 1929–1935, ISBN: 978-1-5090-4442-9. DOI: 10.1109/WiSPNET.2017.8300097. [Online]. Available: <http://ieeexplore.ieee.org/document/8300097/>.
- [42] O. Salman, I. H. Elhajj, A. Kayssi, and A. Chehab, "SDN controllers: A comparative study", in *Proceedings of the 18th Mediterranean Electrotechnical Conference: Intelligent and Efficient Technologies and Services for the Citizen, MELECON 2016*, IEEE, Apr. 2016, pp. 1–6, ISBN: 9781509000579. DOI: 10.1109/MELCON.2016.7495430. [Online]. Available: <http://ieeexplore.ieee.org/document/7495430/>.
- [43] ITU-T, "P.10 : New definitions for inclusion in Recommendation ITU-T P.10/G.100", Tech. Rep. [Online]. Available: <https://www.itu.int/rec/T-REC-P.10-200807-S!Amd2/en>.
- [44] R. Jain, "Quality of experience", *IEEE Multimedia*, vol. 11, no. 1, pp. 96–95, Jan. 2004, ISSN: 1070-986X. DOI: 10.1109/MMUL.2004.1261114. [Online]. Available: <http://ieeexplore.ieee.org/document/1261114/>.
- [45] DASH-IF, "Proposed QoE Media Metrics standardization for segmented media playback", Tech. Rep., 2016. [Online]. Available: <https://dashif.org/docs/ProposedMediaMetricsforSegmentedMediaDelivery-r12.pdf>.
- [46] R. K. Mok, E. W. Chan, X. Luo, and R. K. Chang, "Inferring the QoE of HTTP video streaming from user-viewing activities", in *Proceedings of the first ACM SIGCOMM workshop on Measurements up the stack - W-MUST '11*, New York, New York, USA: ACM Press, 2011, p. 31, ISBN: 9781450308007. DOI: 10.1145/2018602.2018611. [Online]. Available: <http://dl.acm.org/citation.cfm?doid=2018602.2018611>.
- [47] T. Porter and X.-H. Peng, "An Objective Approach to Measuring Video Playback Quality in Lossy Networks using TCP", *IEEE Communications Letters*, vol. 15, no. 1, pp. 76–78, Jan. 2011, ISSN: 1089-7798. DOI: 10.1109/LCOMM.2010.110310.101642. [Online]. Available: <http://ieeexplore.ieee.org/document/5634160/>.

- [48] Baoxian Zhang and H. Mouftah, "Forwarding stat scalability for a multicast provisioning in ip networks", *IEEE Communications Magazine*, vol. 41, no. 6, pp. 46–51, Jun. 2003, ISSN: 0163-6804. DOI: 10.1109/MCOM.2003.1204747. [Online]. Available: <http://ieeexplore.ieee.org/document/1204747/>.
- [49] J. C-I Chuang and M. A. Sirbu, "Pricing Multicast Communication: A Cost-Based Approach", in *Proceedings of INET*, 1998. [Online]. Available: <http://repository.cmu.edu/tepper>.
- [50] C. Adjih, L. Georgiadis, P. Jacquet, and W. Szpankowski, "Multicast tree structure and the power law", *IEEE Transactions on Information Theory*, vol. 52, no. 4, pp. 1508–1521, Apr. 2006, ISSN: 0018-9448. DOI: 10.1109/TIT.2006.871602. [Online]. Available: <http://ieeexplore.ieee.org/document/1614080/>.
- [51] D. Skorin-Kapov and R. B. Willumstad, *Some Notes on Cost Allocation in Multicasting*, Dec. 2012. [Online]. Available: <https://hrcak.srce.hr/96685?lang=en>.
- [52] S. Herzog, S. Shenker, and D. Estrin, "Sharing the "cost" of multicast trees: an axiomatic analysis", *IEEE/ACM Transactions on Networking*, vol. 5, no. 6, pp. 847–860, 1997, ISSN: 10636692. DOI: 10.1109/90.650144. [Online]. Available: <http://ieeexplore.ieee.org/document/650144/>.
- [53] M. Janic and P. Van Mieghem, "The gain and cost of multicast routing trees", in *2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No.04CH37583)*, vol. 5, IEEE, 2004, pp. 4625–4630, ISBN: 0-7803-8567-5. DOI: 10.1109/ICSMC.2004.1401261. [Online]. Available: <http://ieeexplore.ieee.org/document/1401261/>.
- [54] S. Raghu, T. Subashri, and K. R. Vimal, "Literature survey on traffic-based server load balancing using SDN and open flow", in *2017 Fourth International Conference on Signal Processing, Communication and Networking (ICSCN)*, IEEE, Mar. 2017, pp. 1–6, ISBN: 978-1-5090-4740-6. DOI: 10.1109/ICSCN.2017.8085416. [Online]. Available: <http://ieeexplore.ieee.org/document/8085416/>.
- [55] Y.-J. Chen, Y.-H. Shen, and L.-C. Wang, "Traffic-Aware Load Balancing for M2M Networks Using SDN", in *2014 IEEE 6th International Conference on Cloud Computing Technology and Science*, IEEE, Dec. 2014, pp. 668–671, ISBN: 978-1-4799-4093-6. DOI: 10.1109/CloudCom.2014.37. [Online]. Available: <http://ieeexplore.ieee.org/document/7037734/>.
- [56] U. Zakia and H. Ben Yedder, "Dynamic load balancing in SDN-based data center networks", in *2017 8th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, IEEE, Oct. 2017, pp. 242–247, ISBN: 978-1-5386-3371-7. DOI: 10.1109/IEMCON.2017.8117206. [Online]. Available: <http://ieeexplore.ieee.org/document/8117206/>.
- [57] F. S. Fizi and S. Askar, "A novel load balancing algorithm for software defined network based datacenters", in *2016 International Conference on Broadband Communications for Next Generation Networks and Multimedia Applications (CoBCom)*, IEEE, Sep. 2016, pp. 1–6, ISBN: 978-1-5090-2270-0. DOI: 10.1109/COBCOM.2016.7593506. [Online]. Available: <http://ieeexplore.ieee.org/document/7593506/>.
- [58] I. P. A. Suwandika, M. A. Nugroho, and M. Abdurahman, "Increasing SDN Network Performance Using Load Balancing Scheme on Web Server", in *2018 6th International Conference on Information and Communication Technology (ICoICT)*, IEEE, May 2018, pp. 459–463, ISBN: 978-1-5386-4572-7. DOI: 10.1109/ICoICT.2018.8528803. [Online]. Available: <https://ieeexplore.ieee.org/document/8528803/>.
- [59] M. Qilin and S. Weikang, "A Load Balancing Method Based on SDN", in *2015 Seventh International Conference on Measuring Technology and Mechatronics Automation*, IEEE, Jun. 2015, pp. 18–21, ISBN: 978-1-4673-7143-8. DOI: 10.1109/ICMTMA.2015.13. [Online]. Available: <http://ieeexplore.ieee.org/document/7263504/>.
- [60] R. Gokilabharathi and P. Deenalakshmi, "Efficient Load Balancing to Enhance the Quality of Service (QOS) in Software Defined Networking (SDN)", in *2018 2nd International Conference on Trends in Electronics and Informatics (ICOEI)*, IEEE, May 2018, pp. 1046–1051, ISBN: 978-1-5386-3570-4. DOI: 10.1109/ICOEI.2018.8553901. [Online]. Available: <https://ieeexplore.ieee.org/document/8553901/>.
- [61] A. Bikfalvi, J. García-Reinoso, I. Vidal, F. Valera, and A. Azcorra, "P2P vs. IP multicast: Comparing approaches to IPTV streaming based on TV channel popularity", *Computer Networks*, vol. 55, no. 6, pp. 1310–1325, Apr. 2011, ISSN: 1389-1286. DOI: 10.1016/J.COMNET.2010.12.020. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1389128610003877>.
- [62] J. Nogueira, D. Gonzalez, L. Guardalben, and S. Sargento, "Over-The-Top Catch-up TV content-aware caching", in *2016 IEEE Symposium on Computers and Communication (ISCC)*, IEEE, Jun.

- 2016, pp. 1012–1017, ISBN: 978-1-5090-0679-3. DOI: 10.1109/ISCC.2016.7543869. [Online]. Available: <http://ieeexplore.ieee.org/document/7543869/>.
- [63] A. Salvador, J. Nogueira, and S. Sargento, “QoE Assessment of HTTP Adaptive Video Streaming”, in, 2015, pp. 235–242. DOI: 10.1007/978-3-319-18802-7\_32. [Online]. Available: [http://link.springer.com/10.1007/978-3-319-18802-7\\_32](http://link.springer.com/10.1007/978-3-319-18802-7_32).
  - [64] A. Burger, *4K OTT Video Penetration to Reach 10% by 2021 - Telecompetitor*. [Online]. Available: <https://www.telecompetitor.com/4k-ott-video-penetration-to-reach-10-by-2021/> (visited on 12/05/2018).
  - [65] Glenn Fiedler, *Why can't I send UDP packets from a browser?* [Online]. Available: <https://gafferongames.com/post/why-i-cant-send-udp-packets-from-a-browser/> (visited on 12/05/2018).
  - [66] D. Kunda, S. Chihana, and M. Sinyinda, “Web Server Performance of Apache and Nginx: A Systematic Literature Review”, *Computer Engineering and Intelligent Systems*, vol. 8, no. 2, pp. 43–52, 2010. [Online]. Available: <https://iiste.org/Journals/index.php/CEIS/article/view/35842>.
  - [67] Prakash P, Biju R, and M. Kamath, “Performance analysis of process driven and event driven web servers”, in *2015 IEEE 9th International Conference on Intelligent Systems and Control (ISCO)*, IEEE, Jan. 2015, pp. 1–7, ISBN: 978-1-4799-6480-2. DOI: 10.1109/ISCO.2015.7282230. [Online]. Available: <http://ieeexplore.ieee.org/document/7282230/>.
  - [68] Netcraft, *September 2018 Web Server Survey*. [Online]. Available: <https://news.netcraft.com/archives/2018/09/24/september-2018-web-server-survey.html> (visited on 12/06/2018).
  - [69] Atlassian, *Floodlight Controller - Project Floodlight*. [Online]. Available: <https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/overview> (visited on 12/01/2018).
  - [70] Ryu, *Welcome to RYU the Network Operating System(NOS) — Ryu 4.30 documentation*. [Online]. Available: <https://ryu.readthedocs.io/en/latest/> (visited on 12/15/2018).
  - [71] OpenvSwitch, *Why Open vSwitch*. [Online]. Available: <https://github.com/openvswitch/ovs/blob/master/Documentation/intro/why-ovs.rst> (visited on 12/07/2018).
  - [72] M. Turnbull, *NGINX vs HAProxy — a bit like comparing a 2CV with a Tesla?* [Online]. Available: <http://www.loadbalancer.org/blog/nginx-vs-haproxy/> (visited on 12/09/2018).
  - [73] WHATWG, *HTML Standard*. [Online]. Available: <https://html.spec.whatwg.org/multipage/webstorage.html#dom-localstorage> (visited on 12/09/2018).
  - [74] K. Spiteri, R. Ugaonkar, and R. K. Sitaraman, “BOLA: Near-optimal bitrate adaptation for online videos”, in *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, IEEE, Apr. 2016, pp. 1–9, ISBN: 978-1-4673-9953-1. DOI: 10.1109/INFOCOM.2016.7524428. [Online]. Available: <http://ieeexplore.ieee.org/document/7524428/>.



# Appendix

Forecasting model parameters description:

- Type of Forecast: Select between three types available to forecast.
- Total Subscribers: Number of total subscribers to provide service.
- Hit Ratio Appliance: Hit ratio for IPTV services at the SR, only Unicast and Hybrid are affected.
- Baseline/Unicast ratio: How much expensive is the Baseline in Multicast Live-TV regarding Unicast and Hybrid in Unicast Live-TV. The default value is 1.30, meaning that is 30% more expensive.
- Growth options: Activate growth parameters, available only at the "Forecast 2017" type.
- More options: Activate more options that can be used on all types of forecast.
- Opex Curve: Show the parameters that produces the opex plot and enable the plot.
- Subscribers growth: Percentage of subscribers growth every year until 2021.
- Internet growth: Percentage of internet growth every year until 2021.
- IPTV growth: Percentage of iptv growth every year until 2021.
- Bandwidth surplus per subscriber: Added bandwidth to each subscriber that don't benefit from cache or buffering.
- SR maximum uplink (Mbps): Redefine the maximum uplink bandwidth of a Service Router.
- OLT maximum uplink (Mbps): Redefine the maximum uplink bandwidth of a Optical Line Termination.
- Appliance uplink/downlink (capex): Choose the maximum bandwidth capacity of the appliance that will cache and buffer content.
- Appliance price (capex): Appliance price per unit.
- Ratio stream (max streams / total subscribers): Peak subscribers watching a stream regarding the total number of subscribers.
- Interval for calculation: Calculate opex from time to time, not for every change in subscribers number.
- Max subscribers per OLT: How much subscribers a OLT can support.
- SD percentage client: How much, in percentage, a subscriber sees a SD TV channel compared to the rest.
- SD percentage headend: Regarding the different types of channels, how much of them are SD at the headend.

- HD percentage client: How much, in percentage, a subscriber sees a HD TV channel compared to the rest.
- HD percentage headend: Regarding the different types of channels, how much of them are HD at the headend.
- 4K percentage client: How much, in percentage, a subscriber sees a 4K TV channel compared to the rest.
- 4K percentage headend: Regarding the different types of channels, how much of them are 4K at the headend.
- TCP usability: How much of the link is usable in Unicast transmissions. Affects Unicast and Hybrid models.
- Custom average internet: How much internet a subscriber uses on average.
- Custom average IPTV: How much IPTV services a subscriber uses on average.
- Custom peak streams: The maximum number of streams being watched at the same time.
- Team cost multicast: Cost value for a multicast team. Available only at the Custom forecast.
- Team max (boxes/streams) multicast: How many streams a multicast team manages. Available only at the Custom forecast.
- Team cost unicast: Cost value for a unicast team. Available only at the Custom forecast.
- Team max (boxes/streams) unicast: How many streams a unicast team manages. Available only at the Custom forecast.