



Politecnico di Bari

Repository Istituzionale dei Prodotti della Ricerca del Politecnico di Bari

QoE-aware Control of Video Streaming Systems

This is a PhD Thesis

Original Citation:

QoE-aware Control of Video Streaming Systems / Manfredi, Gioacchino. - ELETTRONICO. - (2023).
[10.60576/poliba/iris/manfredi-gioacchino_phd2023]

Availability:

This version is available at <http://hdl.handle.net/11589/249040> since: 2023-03-26

Published version

<http://hdl.handle.net/11589/249040>
DOI: 10.60576/poliba/iris/manfredi-gioacchino_phd2023

Terms of use:

Altro tipo di accesso

(Article begins on next page)



Department of Electrical and Information Engineering
ELECTRICAL AND INFORMATION ENGINEERING

Ph.D. Program

SSD: ING-INF/04-SYSTEMS AND CONTROL ENGINEERING

Final Dissertation

QoE-aware Control of Video Streaming Systems

by
Gioacchino MANFREDI

Supervisors:

Prof. Eng. Saverio MASCOLO

Prof. Eng. Luca DE CICCIO

Coordinator of Ph.D. Program:
Prof. Eng. Mario CARPENTIERI

Course n° 35, 01/11/2019-31/01/2023



LIBERATORIA PER L'ARCHIVIAZIONE DELLA TESI DI DOTTORATO

Al Magnifico Rettore
del Politecnico di Bari

Il sottoscritto **Gioacchino Manfredi** nato a **Bari** il **24/06/1994**

residente a **Altamura** in via **Gradisca n. 10** e-mail **gioacchino.manfredi@poliba.it**

iscritto al 3° anno di Corso di Dottorato di Ricerca in **Ingegneria Elettrica e dell'Informazione** ciclo **XXXV**

ed essendo stato ammesso a sostenere l'esame finale con la prevista discussione della tesi dal titolo:

QoE-aware Control of Video Streaming Systems

DICHIARA

- 1) di essere consapevole che, ai sensi del D.P.R. n. 445 del 28.12.2000, le dichiarazioni mendaci, la falsità negli atti e l'uso di atti falsi sono puniti ai sensi del codice penale e delle Leggi speciali in materia, e che nel caso ricorressero dette ipotesi, decade fin dall'inizio e senza necessità di nessuna formalità dai benefici conseguenti al provvedimento emanato sulla base di tali dichiarazioni;
- 2) di essere iscritto al Corso di Dottorato di ricerca in **Ingegneria Elettrica e dell'Informazione** ciclo **XXXV**, corso attivato ai sensi del *"Regolamento dei Corsi di Dottorato di ricerca del Politecnico di Bari"*, emanato con D.R. n.286 del 01.07.2013;
- 3) di essere pienamente a conoscenza delle disposizioni contenute nel predetto Regolamento in merito alla procedura di deposito, pubblicazione e autoarchiviazione della tesi di dottorato nell'Archivio Istituzionale ad accesso aperto alla letteratura scientifica;
- 4) di essere consapevole che attraverso l'autoarchiviazione delle tesi nell'Archivio Istituzionale ad accesso aperto alla letteratura scientifica del Politecnico di Bari (IRIS-POLIBA), l'Ateneo archiverà e renderà consultabile in rete (nel rispetto della Policy di Ateneo di cui al D.R. 642 del 13.11.2015) il testo completo della tesi di dottorato, fatta salva la possibilità di sottoscrizione di apposite licenze per le relative condizioni di utilizzo (di cui al sito <http://www.creativecommons.it/Licenze>), e fatte salve, altresì, le eventuali esigenze di "embargo", legate a strette considerazioni sulla tutelabilità e sfruttamento industriale/commerciale dei contenuti della tesi, da rappresentarsi mediante compilazione e sottoscrizione del modulo in calce (Richiesta di embargo);
- 5) che la tesi da depositare in IRIS-POLIBA, in formato digitale (PDF/A) sarà del tutto identica a quelle consegnate/inviolate/inviarsi ai componenti della commissione per l'esame finale e a qualsiasi altra copia depositata presso gli Uffici del Politecnico di Bari in forma cartacea o digitale, ovvero a quella da discutere in sede di esame finale, a quella da depositare, a cura dell'Ateneo, presso le Biblioteche Nazionali Centrali di Roma e Firenze e presso tutti gli Uffici competenti per legge al momento del deposito stesso, e che di conseguenza va esclusa qualsiasi responsabilità del Politecnico di Bari per quanto riguarda eventuali errori, imprecisioni o omissioni nei contenuti della tesi;
- 6) che il contenuto e l'organizzazione della tesi è opera originale realizzata dal sottoscritto e non compromette in alcun modo i diritti di terzi, ivi compresi quelli relativi alla sicurezza dei dati personali; che pertanto il Politecnico di Bari ed i suoi funzionari sono in ogni caso esenti da responsabilità di qualsivoglia natura: civile, amministrativa e penale e saranno dal sottoscritto tenuti indenni da qualsiasi richiesta o rivendicazione da parte di terzi;
- 7) che il contenuto della tesi non infrange in alcun modo il diritto d'Autore né gli obblighi connessi alla salvaguardia di diritti morali od economici di altri autori o di altri aventi diritto, sia per testi, immagini, foto, tabelle, o altre parti di cui la tesi è composta.

Luogo e data **Bari 23/03/2023**

Firma _____

Il sottoscritto, con l'autoarchiviazione della propria tesi di dottorato nell'Archivio Istituzionale ad accesso aperto del Politecnico di Bari (POLIBA-IRIS), pur mantenendo su di essa tutti i diritti d'autore, morali ed economici, ai sensi della normativa vigente (Legge 633/1941 e ss.mm.ii.),

CONCEDE

- al Politecnico di Bari il permesso di trasferire l'opera su qualsiasi supporto e di convertirla in qualsiasi formato al fine di una corretta conservazione nel tempo. Il Politecnico di Bari garantisce che non verrà effettuata alcuna modifica al contenuto e alla struttura dell'opera.
- al Politecnico di Bari la possibilità di riprodurre l'opera in più di una copia per fini di sicurezza, back-up e conservazione.

Luogo e data **Bari 23/03/2023**

Firma _____



Department of Electrical and Information Engineering
ELECTRICAL AND INFORMATION ENGINEERING

Ph.D. Program

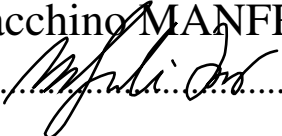
SSD: ING-INF/04-SYSTEMS AND CONTROL ENGINEERING

Final Dissertation

QoE-aware Control of Video Streaming Systems

by

Gioacchino MANFREDI

.....

Referees:

Prof. Eng. Carsten Griwodz

Dr. Eng. Alessandro Giuseppe

Supervisors:

Prof. Eng. Saverio MASCOLO

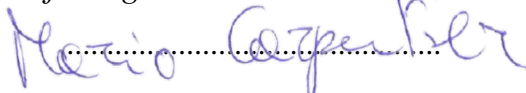
.....

Prof. Eng. Luca DE CICCIO

.....

Coordinator of Ph.D. Program:

Prof. Eng. Mario CARPENTIERI

.....

To my family, always and forever.

Don't let the system get you down

— Mattafix

Abstract

This Ph.D. thesis is composed of two parts: the first and main part is devoted to video streaming and Internet media delivery services. In particular, issues concerning optimal network resource allocation and live video streaming synchronisation are tackled. The second part represents the beginning of a study investigating the asymptotic stability of nonlinear systems with Deep Reinforcement Learning controllers.

Part 1. Video streaming is gaining more and more ground causing an unprecedented growth of multimedia streaming services such as YouTube, Netflix, Twitch, etc. As a consequence, more than half of the global Internet traffic is today due to video contents [11]. To keep high engagement and avoid service abandonment, services delivering videos to massive audiences are required to provide users with a satisfactory Quality of Experience (QoE), which estimates a provider's service from a user's standpoint. On-line video content services should provide users with the best possible Quality of Experience (QoE) given the constraints due to the user device and network. Current video platforms perform a Quality of Service (QoS) fair distribution of network resources. Such an approach is designed to provide concurrent users sharing the same network resources (i.e. network links) with a fair share of network bandwidth with no regard to the heterogeneity of users. As

a consequence, the quality perceived by users is not equalized since, in order to obtain the same level of QoE, users with large screen devices (f.i. Smart TVs) require a larger video bitrate w.r.t. devices with small screens (e.g. smartphones). On the users' side, players run a control algorithm that strives to selfishly improve the quality individually perceived by users. As a consequence, this control architecture leads—in the best case—to maximise the average quality collectively perceived by all users and not to a resource distribution that results fair in terms of user-perceived quality.

In the context of the Cloud-based pLatform for Immersive adaPtive video Streaming (CLIPS) project, an optimisation framework to design the QoE-fair network bandwidth allocation strategy based on the Multi-Commodity Flow Problem (MCFP) [127] has been proposed. QoE has been modelled as the visual quality, which represents the main factor in the definition of such a metric. Visual quality has been evaluated through the Video Multi-method Assessment Fusion (VMAF), which is a full-reference video quality assessment tool. Moreover, to sensibly reduce the number of variables involved in the optimisation procedure, a traffic clustering approach has been integrated into the framework. Although the state of the art provides several ways to estimate the QoE associated with a user, the video quality will be considered as the measurement unit for QoE throughout this work. In addition to optimal network resource distribution, live video streaming synchronisation issues have been investigated.

Live streaming events, such as football matches, are nowadays enjoyed together by users even if they are not physically in the same place. This has been possible thanks to the advent of social media applications and mobile devices. A crucial point concerning this service is the synchronisation of video playback among geographically distributed users, which prevents users' service abandonment. When

comments and reactions on social networks are left, a not synchronised video playback can be easily noticed and be detrimental to users' feelings of *togetherness*.

To this end, a distributed control approach has been proposed to achieve synchronisation among users. In particular, the well-known consensus problem of simple integrators with saturated inputs has been deployed to design a distributed playback synchronisation framework. Furthermore, a leader-follower approach has been adopted with the aim of ensuring a controlled synchronisation among users in order to obtain the least possible delay with respect to the video content provider. Finally, an event-triggered control is introduced as an enhancement to the previously developed control to reduce the information exchanged among users.

Part 2. Recent years have witnessed a considerable spread of machine learning techniques to solve problems and enhance procedures in various fields. In control systems, machine learning has brought several advantages, such as the possibility of controlling nonlinear systems that would be hard to control with conventional techniques, the possibility of controlling systems whose model is not known, and so on. To this end, Deep Reinforcement Learning (DRL) algorithms aim at learning control policies through interaction with an environment. However, despite the encouraging performance, such algorithms are still mainly employed in simulation environments since most of the real-world applications are safety critical. For this reason, it is important to have some guarantees on the asymptotic stability of the system controlled with a DRL policy.

The framework proposed to take a step forward in this direction consists of extracting the DRL control policy that proves good at achieving the control goal. Then, a *Learner-Verifier* scheme leverages a counterexample-based strategy to

synthesise a Lyapunov function that certifies the asymptotic stability of the system controlled with the policy extracted. This framework also provides useful insights into safety guarantees that are often necessary when it comes to real applications.

Keywords— Video Streaming Systems - Quality of Experience - Quality of Service - Network Resources - Synchronisation - Consensus - Event-Triggered control - Machine Learning - Deep Reinforcement Learning - Lyapunov Neural Networks - Nonlinear Systems - Lyapunov Stability Theory

Abstract

(Italian)

Questa tesi di dottorato è suddivisa in due parti: la prima parte, nonché la principale, è dedicata al video streaming e ai servizi Internet multimediali. In particolare, in tale elaborato si analizzano i problemi legati all’allocazione ottima delle risorse di rete nell’ambito del video streaming ed alla sincronizzazione degli utenti durante una diretta streaming. La seconda parte rappresenta l’inizio di uno studio che analizza l’asintotica stabilità dei sistemi nonlineari con controllori Deep Reinforcement Learning.

Parte 1. Il video streaming sta acquistando sempre più importanza causando una notevole crescita dei servizi di streaming multimediale come YouTube, Netflix, Twitch ecc. Di conseguenza, più della metà del traffico globale su Internet è oggi dovuto a contenuti video [11]. Per fidelizzare gli utenti ed evitare l’abbandono del servizio, è necessario che le piattaforme che inviano contenuti video ad un numero massivo di utenti garantiscano agli utenti un livello soddisfacente di Quality of Experience (QoE), la quale mira a stimare il servizio fornito da un provider dal punto di vista dell’utente. I servizi che forniscono contenuti video online dovrebbero garantire agli utenti la migliore QoE possibile dati i vincoli dovuti ai dispositivi degli utenti e alla rete. Le attuali piattaforme video adoperano una distribuzione equa

delle risorse di rete dal punto di vista della Quality of Service (QoS). Con tale approccio, agli utenti che condividono le stesse risorse di rete (cioè i link di rete) vengono assegnate parti delle risorse di banda senza considerare l'eterogeneità degli utenti. Di conseguenza, la qualità percepita dagli utenti non è uguagliata dal momento che, per ottenere lo stesso livello di QoE, gli utenti con dispositivi dotati di ampio schermo (ad esempio le smart TV) hanno bisogno di un bitrate maggiore rispetto a dispositivi con piccoli schermi (ad esempio gli smartphone). Lato utente, i player sul dispositivo eseguono un algoritmo di controllo che mira a migliorare in maniera 'egoista' la qualità percepita individualmente dagli utenti. Tale architettura di controllo porta—nel migliore dei casi—ad una massimizzazione della qualità media percepita collettivamente da tutti gli utenti piuttosto che ad una distribuzione delle risorse che risulti equa in termini di qualità percepita dall'utente.

Nell'ambito del progetto Cloud-based pLatform for immersive adaPtive video Streaming (CLIPS), si è proposto un framework di ottimizzazione per l'allocazione delle risorse di rete in maniera QoE-fair. La QoE è stata modellata utilizzando la qualità visiva, che rappresenta il fattore principale quando tale metrica viene definita. La qualità visiva è stata valutata attraverso il Video Multi-method Assessment Fusion (VMAF), che è uno strumento per la valutazione della qualità video. Inoltre, per ridurre notevolmente il numero di variabili coinvolte nella procedura di ottimizzazione, si è integrato un metodo di clustering del traffico. Nonostante lo stato dell'arte fornisca molti modi per stimare la QoE associata ad un utente, la qualità visiva è considerata l'unità di misura per eccellenza in questo lavoro.

Gli eventi streaming dal vivo, come ad esempio una partita di calcio, sono al giorno d'oggi seguiti da utenti che di solito non sono fisicamente nello stesso luogo.

Ciò è stato reso possibile grazie alla diffusione dei social media e dei dispositivi mobili. Un aspetto cruciale riguardante tale servizio è la sincronizzazione del video playback tra utenti geograficamente distribuiti. Quando vengono lasciati commenti e reazioni sui social network, un video playback non sincronizzato può essere facilmente notato e quindi avere risultati negativi sul senso di *togtherness* che l'utente avverte quando vede un evento in diretta con altri utenti.

A tal fine, è stato proposto un approccio di controllo distribuito per ottenere la sincronizzazione tra gli utenti. In particolare, è stato utilizzato il noto problema del consenso di integratori semplici con input saturati per progettare una sincronizzazione distribuita del video playback. Inoltre, è stato adottato un approccio leader-follower con l'obiettivo di assicurare una sincronizzazione tra gli utenti e al contempo di ottenere il minor ritardo possibile rispetto al provider. Infine, è stato introdotto un controllo di tipo event-triggered per migliorare il controllo precedente e per ridurre le informazioni scambiate tra gli utenti.

Parte 2. Negli ultimi anni si è avuta una notevole diffusione di tecniche di machine learning per risolvere problemi e migliorare procedure in vari campi. Nell'ambito dei sistemi di controllo, il machine learning ha apportato diversi vantaggi, come ad esempio la possibilità di controllare sistemi nonlineari che sarebbero difficili da controllare con le tecniche classiche, la possibilità di controllare sistemi il cui modello non è noto e così via. Gli algoritmi di Deep Reinforcement Learning (DRL) mirano ad apprendere policy di controllo attraverso l'interazione con l'environment. Tuttavia, nonostante le performance incoraggianti, tali algoritmi sono ancora impiegati principalmente in ambienti simulativi dato che la maggior parte delle applicazioni reali presentano problemi di sicurezza. Per questo motivo, è importante avere delle garanzie sulla stabilità asintotica del sistema controllato mediante una policy DRL.

Per ovviare a tale problema, l'architettura proposta consiste nell'estrazione di una policy di controllo DRL in grado di raggiungere l'obiettivo di controllo. Successivamente, uno schema *Learner-Verifier* utilizza un approccio basato su controesempi per provare a sintetizzare una funzione di Lyapunov che certifica l'asintotica stabilità del sistema controllato con la policy estratta. L'architettura permette anche di ottenere delle considerazioni utili per la sicurezza, che è spesso richiesta nel caso di applicazioni reali.

Table of Contents

List of Figures	xiii
List of Abbreviations	xiv
Scientific Contributions	xvi
1 Introduction	1
1.1 Video Streaming Background	1
1.2 Protocols	3
1.3 Video Streaming Quality Metrics	8
1.3.1 Quality of Service	9
1.3.2 Quality of Experience	9
1.3.2.1 QoE estimation methods	10
1.3.2.2 Key factors for QoE evaluation	16
1.3.2.3 Objective quality assessment	19
1.3.2.4 Structural Similarity Index	20
1.4 Machine Learning	24
1.4.1 Reinforcement Learning	28
1.5 Telecommunication System Architecture	33
1.6 Thesis Outline	34

I	Video Streaming Systems	35
2	Optimal QoE-fair Resource Allocation in Multipath Video Delivery	
	Networks	36
2.1	Background and Related Work	36
2.2	The Multi-Commodity Flow Problem	43
2.3	The Proposed Approach	55
2.3.1	Definitions	55
2.3.2	Measuring the visual quality	58
2.3.3	Demand Weights Computation	60
2.3.4	Video Clustering	61
2.4	Results	65
2.4.1	General Settings	65
2.4.2	QoE Fairness vs Average Visual Quality	67
2.4.3	Computation Time	71
2.5	Concluding Remarks	72
3	Synchronising Live Video Streaming Players Via Consensus	74
3.1	Introduction and Background	75
3.2	Playback Time Model	79
3.2.1	Video playback time model	80
3.2.2	The synchronisation issue	83
3.2.3	The Adaptive Media Playout model	85
3.2.4	The Proposed Synchronisation Approach	87
3.3	Distributed event-triggered control	97
3.4	Results	100
3.5	Concluding Remarks	106

II Asymptotic Stability of Systems with Deep Reinforcement Learning Controllers	107
4 On Asymptotic Stability of Nonlinear Systems with Deep Reinforcement Learning Controllers	108
4.1 Introduction	109
4.2 Related Work	111
4.3 Preliminaries	112
4.4 The Proposed Architecture	116
4.5 Results	119
4.6 Concluding Remarks	128
5 Conclusions and Future Research Directions	129
References	133

List of Figures

1.1	Severe Tire Damage	2
1.2	RTSP/RTP sequence diagram	4
1.3	HAS-based Streaming Scheme	5
1.4	MPD Data Model	7
1.5	Irregular pentagon of the 5 KPIs in [68]	12
1.6	SSIM quality assessment diagram [158]	22
1.7	Pensieve scheme [105]	30
2.1	Visual quality function of the video bitrate and the client screen resolution	38
2.2	Distribution network	39
2.3	Resource allocation in a sample network	54
2.4	Video level set representation in a <i>ladder</i> graph	57
2.5	Computation of weights	61
2.6	Proposed clustering procedure	63
2.7	The proposed Video Control Plane	64
2.8	QoE-Fairness vs Average Visual Quality	67
2.9	CDF of visual quality for different user classes and distribution networks in the case of a 500Gbps load	70

2.10	Computation time required to solve the MCFP	72
3.1	Live video streaming system	80
3.2	Playback time dynamics of two users i and j joining at different time instants $t_j^{(i)}$ and $t_j^{(j)}$	84
3.3	The proposed synchronisation control architecture	85
3.4	State dynamics $x_i(t)$ for the ring topology	101
3.5	Ring topology with a leader node	101
3.6	State dynamics $x_i(t)$ for the ring topology with a leader node . . .	101
3.7	Control inputs $u_i(t)$ in the case of a ring topology with a leader node	101
3.8	Network topology with groups of clients and a leader	102
3.9	State dynamics $x_i(t)$ for the topology of Fig. 3.8	103
3.10	State dynamics $x_i(t)$ with the leader marked by the yellow node in Fig. 3.8	103
3.11	State dynamics $x_i(t)$ for the ring topology in the event-triggered case	104
3.12	Control inputs $u_i(t)$ for the ring topology in the event-triggered case	104
3.13	Triggering times for each agent in the ring topology	105
3.14	Triggering times for each agent in the topology of Fig. 3.8	105
4.1	Proposed architecture	116
4.2	Lyapunov function for Inverted Pendulum with DDPG control . .	121
4.3	Region of Attraction for Inverted Pendulum with DDPG control .	122
4.4	Region of Attraction for Inverted Pendulum with TD3 control . . .	123
4.5	Lyapunov function for Bicycle circle tracking with DDPG control	124
4.6	Region of attraction for Bicycle circle tracking with DDPG control	125
4.7	Region of attraction for Bicycle circle tracking with TD3 control .	125

List of Abbreviations

KPI	Key Performance Indicator
HTTP	Hypertext Transfer Protocol
HAS	HTTP Adaptive Streaming
DASH	Dynamic Adaptive Streaming over HTTP
ABR	Adaptive BitRate
MPD	Media Presentation Description
SDN	Software Defined Network
QoE	Quality of Experience
QoS	Quality of Service
MOS	Mean Opinion Score
SSIM	Structural Similarity Index
VMAF	Video Multi-mathod Assessment Fusion
MCFP	Multi-Commodity Flow Problem
MAS	Multi-Agent System

List of Abbreviations

RL	Reinforcement Learning
DRL	Deep Reinforcement Learning
PF	Proportional Fair
BL	Baseline
AMP	Adaptive Media Payout
LNN	Lyapunov Neural Network
DDPG	Deep Deterministic Policy Gradient
TD3	Twin Delayed Deep Deterministic Policy Gradient
SMT	Satisfiability Modulo Theory

Scientific Contributions

Scientific contributions summarising the activity led during the Ph.D. are listed below. Contributions are contained in conference or journal papers as shown.

International Journals

- Manfredi Gioacchino, De Cicco Luca, and Mascolo Saverio. "Optimal QoE-fair Resource Allocation in Multi-Path Video Delivery Networks." IEEE Transactions on Network and Service Management (2022).
- Manfredi Gioacchino, Vito Andrea Racanelli, De Cicco Luca, and Mascolo Saverio. "Live Streaming Synchronisation Using Event-triggered Consensus Control." IEEE Transactions on Networking (Submitted).

International Conferences

- De Cicco Luca, Manfredi Gioacchino, Palmisano Vittorio, and Mascolo Saverio. "QoE-fair resource allocation for DASH video delivery systems." Proceedings of the 1st International Workshop on Fairness, Accountability, and Transparency in MultiMedia. 2019.
- De Cicco Luca, Manfredi Gioacchino, Palmisano Vittorio, and Mascolo Saverio. "A Multi-Commodity Flow Problem for Fair Resource Allocation in Multi-Path Video Delivery Networks." IFAC-PapersOnLine 53.2 (2020):

7386-7391.

- Manfredi Gioacchino, De Cicco Luca, and Mascolo Saverio. "Synchronizing Live Video Streaming Players Via Consensus." 2021 European Control Conference (ECC). IEEE, 2021.
- Manfredi Gioacchino, De Cicco Luca, and Mascolo Saverio. "On Asymptotic Stability of Nonlinear Systems with Deep Reinforcement Learning Controllers." 2022 30th Mediterranean Conference on Control and Automation (MED). IEEE, 2022.

Other Papers

- Suraci Vincenzo, Ricciardi Celsi Lorenzo, Giuseppi Alessandro, Manfredi Gioacchino, Di Giorgio Alessandro "Distributed Wardrop Load Balancing in Multi-MTU SCADA Systems." 2018 26th Mediterranean Conference on Control and Automation (MED). IEEE, 2018.

Chapter 1

Introduction

In this Chapter, a historical background concerning video streaming technologies and services is provided. A brief description of the application domain and the protocols involved is then given along with the most used approaches employed to handle control problems in video streaming. Finally, the fundamentals of machine learning in general and deep reinforcement learning in particular are described as an enhancement to both video streaming and nonlinear control systems.

1.1 Video Streaming Background

It can be said that the first streaming dates back to 1881 and concerned a telephonic distribution system called théâtrophone. It consisted of a telephonic transmission system implemented by Clément Ader at the International Exposition of Electricity in Paris. The system allowed users to listen to opera and theatre performances by placing some telephone transmitters across the front of a stage. In 1890 the system officially became a payment service offered by the Theatrophone Company. In 1992, Starlight Networks created StarWorks [[151](#)], the first video



Figure 1.1: Severe Tire Damage

streaming commercial product, which allowed users to watch on-demand videos on corporate Ethernet networks.

The first live stream to ever include both audio and video dates back to 1993 and was about a band called Severe Tire Damage. The video was 152x76 pixels and the audio quality was no better than a telephone connection. The broadcast used about half of the total available bandwidth over the Internet. Figure 1.1 shows a snapshot of the gig that was performed and livestreamed at Xerox PARC.

Also Starlight Networks implemented live video streaming on Ethernet via Internet Protocol (IP), also followed by Protocomm and RealNetworks. The latter broadcast a baseball match between the New York Yankees and the Seattle Mariners over the Internet in 1995. In the same year, Microsoft released a media player called ActiveMovie for media streaming while Apple introduced its player, QuickTime, only in 1999.

By the beginning of the new century, Adobe Flash implemented the Real Time Messaging Protocol (RTMP), gaining ground in the field of video distribution. Flash-based players were widely adopted by streaming platforms such as YouTube.

Then, HTTP-based technologies came out and outperformed the RTMP protocol for quality and scalability. This was mainly due to the chunk-based nature of the new framework for adaptive streaming that allowed the use of common web servers instead of dedicated streaming servers. However, this transition gave rise to several proprietary technologies such as Apple HLS or Microsoft Smooth. With the aim of defining a standardised framework, Moving Picture Experts Group (MPEG) tried to develop Dynamic Adaptive Streaming over HTTP (DASH) in 2012. From then on, non-proprietary frameworks like DASH became the main solution adopted by streaming platforms such as YouTube.

1.2 Protocols

The Internet is a best-effort, non-real-time network and therefore results not suitable for multimedia content streaming. Being a best-effort network implies that data packets are treated in the same way and there is no guarantee that they are successfully delivered in time. As a consequence, when the load on the network increases, network performance may deteriorate. The first video streaming systems were built on top of User Datagram Protocol (UDP) [132], [107]. However, this protocol is not appropriate for the streaming of multimedia content in that there is no congestion control algorithm necessary to avoid network collapse.

In IP-based streaming, the Internet Engineering Task Force (IETF) devised the Real-time Transport Protocol (RTP) [138] responsible for the transmission, while the setup of the streaming session and the record of the state information is carried out by the Real-Time Streaming Protocol (RTSP) [139]. During transmission, as shown in Figure 1.2, the client sends a feedback to the server through RTP Control Protocol (RTCP) [55] so that the server can operate rate adaptation and data delivery scheduling. This way, IETF managed to stream multimedia contents on

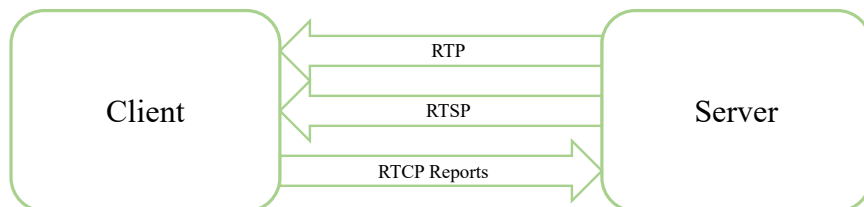


Figure 1.2: RTSP/RTP sequence diagram

IP networks with low overhead and low delay, thus guaranteeing synchronisation [167]. However, at that time all implemented technologies and protocols were proprietary and RTP streaming architectures implied the use of complex and expensive servers and it was not scalable since all intelligence was allocated at the server. In addition, it was thought that end-to-end latency was one of the main key performance indices when designing a video streaming system, but this belief has been shown to be wrong [142], [26].

In 2005 Move Networks [15] proposed a paradigm to adapt multimedia content over the Internet by introducing HTTP Adaptive Streaming (HAS) that works with Transmission Control Protocol (TCP). Such a paradigm is based on chunk distribution with adaptive bitrate approaches without requiring dedicated streaming servers. In addition, TCP guarantees a reliable transmission with an increased resulting video quality. For this reason, HAS became the main approach adopted for video streaming by the most famous platforms and service providers such as Netflix, YouTube, etc. In a nutshell, the computing burden is on the client, which pulls the data from a standard HTTP server. When a multimedia file is ready for streaming, it is divided into fixed-length segments or chunks. Each chunk is then encoded at a different bitrate/resolution. The server creates a manifest, i.e. a file

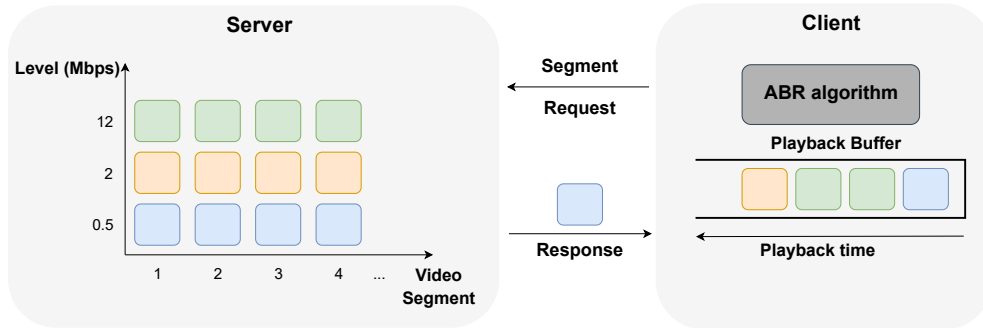


Figure 1.3: HAS-based Streaming Scheme

containing the available segments and other metadata associated to video, audio, etc.

During a session (Figure 1.3), the client first downloads the manifest, then it keeps track of the available bandwidth, buffer status and other features according to which the next more suitable representation of the segment is selected and downloaded. It is worth to remark that each segment is independent of the others and the servers do not have to keep all state information, thus solving the problem of scalability.

In 2009 Apple released the HTTP Live Streaming (HLS), an HTTP-based adaptive bitrate streaming communication protocol for multimedia content delivery. Each media stream is represented at different quality levels and fragmented into chunks of equal length. All the information is contained in a .m3u8 manifest file, which provides the Uniform Resource Locators (URLs) of the video chunks along with other metadata such as chunk duration, resolution, encoding bitrate etc. This way, the user can operate an adaptive bitrate streaming since it can select suitable video fragments according to the available bandwidth and its device resolution. In

the attempt of devising a standard protocol for adaptive video streaming that could be employed by different companies on the market, the Moving Picture Experts Group (MPEG) developed the Dynamic Adaptive Streaming over HTTP (DASH) [150], which in 2011 became an international standard [1]. The DASH standard can be deployed on existing Content Delivery Networks (CDNs) and, just like HLS, it allows seamless switching of the visual quality according to the available network bandwidth. In practice, a DASH server contains segments encoded at different bitrate levels and reported in the Media Presentation Description (MPD) file, which is essentially a manifest. The client reads the MPD and, on the basis of Multi BitRate (MBR) or Adaptive BitRate (ABR) algorithms, fetches the appropriate levels of the segments depending on available bandwidth estimation and playback buffer status. Basically, the goal is to download segments with the highest resolution possible while avoiding rebuffering events. The MPD is an eXtensible Markup Language (XML) document that contains several pieces of information related to the media content and, in particular, to the audio, video and text in it. An MPD file, as shown in Figure 1.4, is composed of a sequence of Periods, each of which regarding a precise temporal interval and containing one or more adaptation sets. In the simplest case there is only one adaptation set containing all audio and video, but to reduce the bandwidth each stream could be split into different adaptation sets. For instance, there could exist more adaptation sets containing video and a specific audio for each language. Inside each adaptation set there may be more than one Representation, each of which is representative of the same media content at different levels of encoding and at multiple screen sizes. Each representation encapsulates Media Segments that represent the actual media content, usually in the form of a URL.

The MPEG Common Media Application Format (CMAF) is an extensible standard for the encoding and packaging of segmented media for delivery and decod-

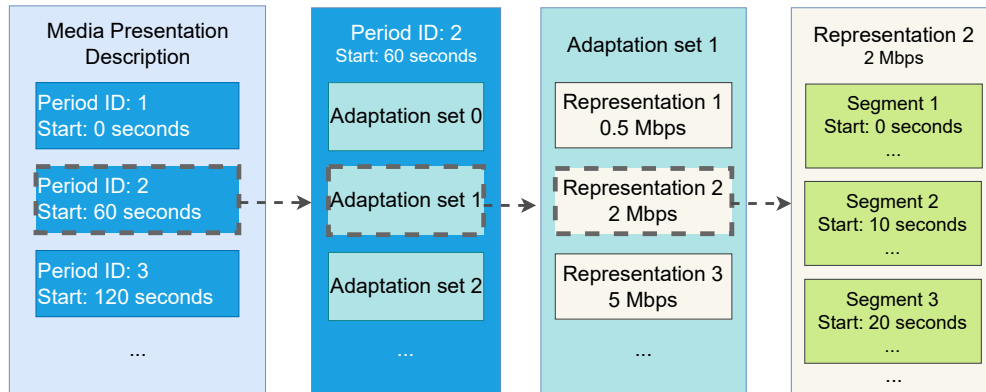


Figure 1.4: MPD Data Model

ing on user devices in adaptive multimedia presentations. The CMAF specification defines several media objects as follows:

- **CMAF Track:** each track contains encoded media samples like audio, video, and subtitles. Such media samples are stored in a CMAF specified container based on ISO Base Media File Format (BMFF). It is also possible to protect media samples with MPEG Common Encryption. Each track has a header and one or more CMAF fragments.
- **CMAF Switching Set:** such sets may contain CMAF alternative tracks with different bitrate and resolution.
- **Aligned CMAF Switching Set:** different CMAF switching sets encoded from the same source with different encodings.
- **Aligned CMAF Switching Set:** different CMAF switching sets encoded from the same source with different encodings.
- **CMAF Selection Set:** a group of switching sets of the same media type with possibly different content (e.g. language, camera angles, etc.).

- CMAF Presentation: one or more presentation time-synchronised selection sets.

The CMAF Hypothetical Reference Model defines how tracks are delivered, combined, and synchronised in CMAF presentations. The same media resources can be delivered to multiple platforms, i.e. both to HLS and DASH platforms, with efficient caching through CMAF Addressable Objects, which consist of:

- CMAF Header: the header contains information for initialising a track.
- CMAF Segment: a sequence of one or more consecutive fragments belonging to the same track.
- CMAF Chunk: a chunk contains a sequential subset of samples from a fragment.
- CMAF Track File: it is an ISOBMFF file.

1.3 Video Streaming Quality Metrics

In this section we are going to analyse two metrics employed when dealing with multimedia streaming services: the Quality of Service (QoS) and the Quality of Experience (QoE). While the former describes the global performance of a service through objective measurements, the latter focuses on the actual user experience based on both objective and subjective metrics. In past years, streaming platforms focused only on network performance, i.e. on QoS, but did not take into account users' opinions and experience. As a result, media streaming services experienced some episodes of user abandonment and did nothing effective to prevent it.

1.3.1 Quality of Service

QoS is the standard way of measuring network performance based on some characteristics such as latency, jitter, packet loss, throughput etc. It could be defined as the ability to guarantee a certain level of performance to a data flow. Traditional video streaming QoS-based schemes, such as for instance Differentiated Service (DiffServ) [20] and Integrated Service (IntServ) [21], guarantee a certain quality level of video sessions in a network on the basis of a set of network measurements and control operations. Such techniques provide QoS guarantees since they are focused on network and packet-based metrics, but do not measure the real impact on the end user. As a consequence, QoS parameters do not consider subjective aspects regarding the Human Visual System (HVS). With the advent of video and real-time communication on the network, the QoS is no longer a sufficient metric in that it does not account for the relationship between the end user and the technology. In particular, QoS does not include the end user satisfaction or the effect of large audio or video level variations. Therefore, this metric shows its utility only when measuring the technical performance of the network.

1.3.2 Quality of Experience

The QoE metric takes into account both objective network indices and subjective metrics that give a measurable grasp of users' opinion on the quality of the service provided. This opinion is not independent of QoS's characteristic parameters: on the contrary, there are independent bounds between these two metrics. The importance of the QoE is due to the fact that a video streaming service provider mainly aims at studying and maximising the quality of the service perceived by users rather than the simple QoS. According to [23], the quality is the outcome of an individual's comparison and judgement process. It includes perception, reflection

about the perception, and the description of the outcome. Experience is an individual's stream of perception and interpretation of one or multiple events. Therefore, the Quality of Experience is the degree of delight or annoyance of the user of an application or service. It depends on the fulfilment of his/her expectations with respect to the utility and/or enjoyment of the application or service in the light of the user's personality and current state. In short, the experience regards a flow of individual perceptions that users transform into delight or annoyance through a personal process called quality. Being both quality and experience subjective factors, it is evident how difficult it is for service providers—especially those in the video streaming distribution—to establish a reliable metric that defines the QoE. Finding a way to objectively compute the QoE is therefore in contrast with the definition of the QoE itself, since it takes into account also personal parameters coming from users. Besides, such parameters are not well defined and known a priori, but result from choices that are subjective, too. For this reason, in the literature there are several different ways to compute and interpret the QoE according to the parameters considered. Consequently, each method has its pros and cons because some aspects are considered but some others are neglected.

1.3.2.1 QoE estimation methods

In [84], the authors use traditional QoS parameters to get a measure of the QoE. This is done after a description about the correlation between QoE and QoS and the design of a methodology to extract an evaluation of the QoE starting from the parameters of the QoS. The expression relating these two metrics is the following:

$$QoE(QoS) = K \frac{e^{QoS-\alpha} + e^{-QoS+\alpha}}{e^{QoS-\alpha} + e^{-QoS+\alpha} + \beta} + 1 \quad (1.1)$$

where α represents a specific class of QoS, i.e. a particular combination of con-

straints on performance variables; β depends on the service class provided; K is a constant mapping the user's satisfaction to the given service. The value of QoE obtained is then mapped on the Mean Opinion Score (MOS) scale ranging from 0 to 5. Gong et al. in [68] propose a model that is not based on surveys or direct evaluations of users to avoid too subjective opinions (and so possibly subject to noise) and to spare time. The proposed model for the QoE measurement is based on five factors: retainability, usability, integrality, availability, and instantaneousness. A factor b is associated with retainability, which denotes the frequency of interruption of a service; the usability of a service is identified with letter d ; integrality, which includes jitter, packet loss and delay, is represented with parameter a ; availability denotes how many times the service is accessed successfully and is identified with parameter c , while instantaneousness, denoted with e , is the response time to establish and access the service.

With these five Key Performance Indicators (KPIs) the authors derived the following model of the QoE:

$$QoE = \frac{1}{2} \sin \lambda (ab + bc + cd + de + ca) \quad (1.2)$$

where λ is a constant properly tuned and the value of the QoE is the area of the irregular pentagon obtained from the five KPIs as shown in Figure 1.5.

Another approach for the QoE computation based on its correlation with the QoS can be found in [111]. The model is based on adaptive learning and can be divided into three submodels:

- User model: it takes into account all relevant characteristics associated with the user such as position, preference, type of device etc.
- Domain model: it represents the concept of the subject domain and it is usu-

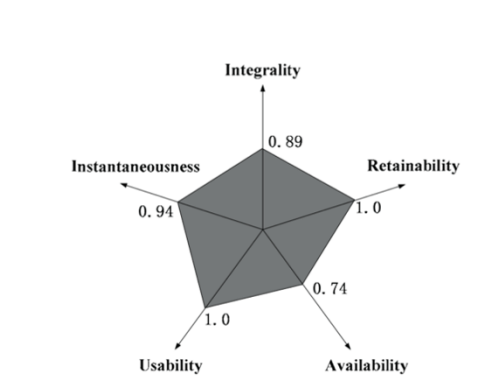


Figure 1.5: Irregular pentagon of the 5 KPIs in [68]

ally described by concept structures like concept maps, semantic networks or concept graphs;

- Adaptive model: it links the two previous models through adaptive learning.

Denoting with QoL the Quality of Learning, i.e. the feedback related to the learning strategy, and with QoF the Quality of Flow, i.e. users' subjective feedback and the technology employed, the QoE can be expressed as:

$$QoE = f(QoL(QoS), QoF(QoS)) \quad (1.3)$$

In this context, flow is a psychological concept and it describes a state where people are so completely immersed in an activity that they lose track of time, which leads to an intensive interaction with the service. The flow is closely related to the QoS. In [69] the concept of QoS and User Experience (UE) is used to derive a formula of the QoE that depends on both parameters. The study is carried out on virtual reality applications but it could be extended also to the case of 2D video streaming without substantial changes. User Experience represents the part of subjective parameters related to the user and can be divided into different

subgroups:

- Perception measurements: they express how a user perceives the service and can be very different according to users;
- Rendering quality: it describes the quality associated with the three fundamental elements of virtual reality, i.e. graphics, audio, and tactile technology;
- Physiological measurements: they refer to the biological parameters directly measured on the user while consuming the service. Such parameters outline the user status while monitoring factors like stress and brain activity;
- Psychological measurements: they express the user status through the observation of his/her behaviour during the service and without the measurement devices typical of the previous point.

The resulting formula of the QoE is a weighted linear combination of the QoS and UE and it can be defined as follows:

$$QoE = \zeta \cdot QoS + (1 - \zeta) \cdot UE \quad (1.4)$$

where ζ between 0 and 1 controls the weight assigned to the QoS and UE. An interesting study carried out in [53] presents in a detailed way the relationship existing between QoS parameters and those related to the QoE. This relationship is called IQX hypothesis for exponential interdependency between QoS and QoE. It is used to compute the immediate quantitative influence that variations of QoS parameters have on the QoE. Results show that the user sensitivity to the QoE is as high as the quality the user is receiving from the service. If the QoE is very high, then a small disturbance will have a considerably negative effect and

the QoE will plummet. On the contrary, if the QoE is already low, a further disturbance or deterioration of the network conditions will not heavily affect the already negative experience of the user. In [53], the authors assume that a change in the QoE is the consequence of a change of the same entity in the QoS. Such a change depends on the current level of the QoE in a linear way, thus obtaining the following relationship:

$$\frac{\partial QoE}{\partial QoS} \sim -(QoE - \gamma) \quad (1.5)$$

The solution of this differential equation allows us to obtain the relationship between the QoE and the QoS, which in other words gives the IQX hypothesis.

$$QoE = \alpha \cdot e^{-\beta \cdot QoS} + \gamma \quad (1.6)$$

The validity of this expression is proven empirically and it is shown to be more accurate than those presented in the literature. Some works like [50], [83], consider a logarithmic relationship, according to which the QoE variation depends on the reciprocal of the QoS.

$$\frac{\partial QoE}{\partial QoS} \sim -\frac{1}{QoS} \quad (1.7)$$

In [4], a different approach is introduced. It aims at improving the quality perceived by the user and at the same time minimising the network resources. This QoE management approach allows service providers to predict the user's experience and, consequently, to optimally allocate network resources. To estimate the QoE qualitative indicators associated with a user starting from known quantitative indicators, a statistical model is employed, and the resulting function is:

$$f_{km} = u_0 + u_1X_{1km} + u_2X_{2km} + \cdots + u_pX_{pkm} \quad (1.8)$$

Where f_{km} is the quality prediction concerning the observation m for the group of users k , X_{ikm} is the value of the quantitative indicators X_i for the observation m for the group k , and u_i is a coefficient associated with $i = 1 \dots p$.

Xu et al. [166] consider a probabilistic approach concerning packets. The assumption is that packets reach the destination buffer with an arrival rate λ according to a Poisson process. The entire video is composed of N consecutive packets while the buffer can contain at most K packets. When a user requests a video, the prefetching phase starts, during which the user accumulates a sufficient number of bits of the video until the so called startup threshold is achieved. When this happens, the user starts playing the video until a possible rebuffering event occurs. In other words, if the buffer depletes, the video freezes and it is necessary to wait until the buffer receives a sufficient number of bit to start the video playback again. The phases of the approach proposed in [166] are:

- Prefetching: the user's device starts downloading the packets following a Poisson process and the playback video hasn't started yet;
- Playback: after downloading a sufficient number of bits and reaching the startup threshold, the video is played by the user;
- Starvation: if some problems occur and the buffer gets empty, the video stops;
- Rebuffering: the device starts downloading the video packets again until the number of bits available is equal to the startup threshold;
- End: after receiving all the N video packets, the connection ends.

The goal is to maximise the QoE of the user by analysing the trade-off between the startup threshold and the starvation phenomenon. A too high startup threshold implies a high waiting time for the user before the video starts (playback phase). This leads to a reduction in the QoE or, in the worst case, the user abandons the service. On the other hand, a too low threshold means increasing the risk the buffer gets empty and goes in starvation. This implies the interruption of the video and, once again, a decrease in the QoE. It is then necessary to design an accurate model of the buffer [33] to obtain better results on the basis of the aforementioned trade-off.

1.3.2.2 Key factors for QoE evaluation

Factors affecting the experience of a user enjoying a video streaming service are different and in many cases are difficult to identify. Among these, the literature provides a deep study of two factors considerably impacting the QoE: initial delays and playback stalls. In order to have a better understanding of initial delays and playback stalls, Hossfield et al. [72] have studied the impact of initial delays on the QoE perceived by a user and have evaluated their effects with respect to those produced by playback stalls during the visualisation of the video. The study was carried out on two videos, one lasting 60 seconds and the other one 30. The metric used to evaluate the QoE associated with a user is the MOS, which is based on a scale of satisfaction ranging from 1 to 5. If, for instance, an 8 second stall occurs, the MOS related to the 30 second video is visibly lower than that associated with the 60 second video. This means that the QoE perceived by users is worse if the stall occurs during the playback of a shorter video. In addition, a logarithmic relationship best approximates the mapping between waiting times and the MOS in the case of initial delays while for stalls the relationship is exponential. Since waiting times for stalls usually bear a MOS lower than waiting times due to initial

delays, when designing a video streaming service initial delays are to be preferred to playback stalls. A further study carried out by the same authors confirms that users prefer to experience an initial delay before watching a video rather than a playback stall in the middle of it of the same length. An important factor employed to explain how playback stalls degrade QoE more than initial delays comes directly from psychology. According to Serial Position Effect [35], a person tends to remember the first and last elements better than those in the middle. The term Recency Effect was forged by the psychologist Hermann Ebbinghaus through a series of studies he carried out on himself. Given a list, he discovered that the precision of memories vary according to the position of the elements in the list [47]. When people are asked to remember a list of elements in any order, they tend to start from the end of it because they remember the last elements better. This is the so-called Recency Effect, which has its roots in the working memory, i.e. a model introduced to describe more accurately the dynamics of the short-term memory. Then, it has been assessed that among the previous elements, those most frequently remembered are those appearing at the beginning of the list. In this case we talk about Primary Effect, a mechanism concerning long-term memory [42], [116]. These temporal effects on human perceptions play an important role with regard to the occurrence of the degradation effects and their timing. According to the Recency Effect, it has been observed that the general quality perceived by a user is strongly affected by a single negative event occurring by the end of the service rather than at the beginning. Moreover, the longer the video watched, the less the impact of the playback stall due to the Recency Effect.

To sum up, in case of bad network conditions, among the possible effects negatively affecting the QoE, users prefer initial delays than intermediate stalls. This result provides an important guideline for video streaming service designers because it suggests to exploit a potential initial delay to perform pre-buffering op-

erations in order to avoid playback stalls once the video has started. However, in case of unfavourable conditions (e.g. bottlenecks, scarce bandwidth resources, etc.) it is not always easy to avoid intermediate stalls. In [73] further results on the impact of initial delays and intermediate stalls on the QoE are reported. Data are drawn from a research carried out through crowd-sourcing where 1349 users participated in the tests on the stalls with 4047 videos watched. In this research, several variables were analysed as possible key factors influencing the QoE. After gathering data about network speed, the browser employed, users' background, etc., it was observed that the most relevant factors are:

- The impact due to the number of playback stalls N ;
- The length of every single stalling event L ;
- The total stalling time T .

According to [75], even if the total stalling time T in a video is the same, users can perceive a different final quality depending on the patterns of occurrence of the stalls, i.e. on parameters N and L . To this end, 2035 users were involved and, after watching some videos with a predefined pattern of stalls, were asked to express their opinion about the quality through a MOS scale. It has been observed that as the length of the stalls L increases, the MOS value decreases faster as the number of stalling events N increases. It is curious to see that a MOS equal to 3 can be obtained both with two stalling events ($N = 2$) lasting 1 second each ($L = 1$) and with a single stall ($N = 1$) lasting 3 seconds ($L = 3$). Hence, it results that $QoE(1 + 1) = QoE(3)$. In other words, the quality perceived by users experiencing 2 stalling events lasting 1 second each would be the same as one stalling event lasting 3 seconds. This analysis shows that the evaluation of a single parameter is not sufficient to adequately model the QoE associated with a video streaming service. Moreover, it is shown that $QoE(1 \cdot 4) \neq QoE(4 \cdot 1)$, which means that

the quality perceived by users when 4 1-second playback stalls occur is not the same as that obtained with a single 4-second stalling event. In general, if the total stalling time $T = N \cdot L$ is constant, the more interruptions occur the faster the user's perceived quality decreases. Therefore, it is the number of stalling events N to have more influence on the perceived quality with respect to the length of each single stalling event L .

1.3.2.3 Objective quality assessment

Objective quality assessment algorithms provide an objective quality measurement of the image that is consistent with subjective human evaluation. Therefore, an ideal objective image quality assessment should be able to mimic the quality predictions of an average human observer. In general, there are two main kinds of objective quality assessment: *no-reference quality assessment* and *full-reference quality assessment*.

No-reference quality assessment: in many real-world applications the reference image is not available and the resulting evaluation is uniquely based on the test image. *Full-reference quality assessment*: the reference image is fully available to be compared with the test image.

Although no-reference algorithms [160], [22], [92], [144], are usually faster, they provide a quite low accuracy [155]. On the contrary, full-reference objective algorithms provide better quality evaluations. The scientific literature is rich of several approaches that attempt to implement effective full-reference algorithms. In the following we will describe only some of the most important metrics widely used in a variety of applications. The first approach is the Mean Square Error (MSE) [66], which is defined as the difference between the reference and test images. Let I_{ref} and I_{test} be the reference and test images respectively and let the image have

a resolution $H \times W$, then the MSE can be computed as follows:

$$MSE = \frac{1}{WH} \sum_{j=1}^H \sum_{i=1}^W (I_{ref}(i, j) - I_{est}(i, j))^2 \quad (1.9)$$

From MSE it is possible to derive the peak-signal-to-noise ratio (PSNR), i.e. the ratio between maximum possible power of a signal and power of distortion. It can be computed by using the following formula:

$$PSNR = 10 \log\left(\frac{D^2}{MSE}\right) \quad (1.10)$$

where D is the dynamic range of pixel intensities. Although MSE is computationally inexpensive, has very nice mathematical properties and is widely used as a measure for optimisation problems, when it comes to matching human perception of the visual quality, it shows poor performance [157]. Therefore, other methods have been designed to overcome this problem [96] and, among them, an important advancement was given by the Structural Similarity Index (SSIM) [158].

1.3.2.4 Structural Similarity Index

SSIM is based on the insight that HVS is highly correlated to the structural information of an image. Therefore, being able to measure structural information change, i.e. those traits that represent the structure of objects in an image [158], implies the possibility of obtaining a good approximation of the perceived image distortion. The SSIM algorithm defines the image degradation as the perceived change in structural information.

The SSIM algorithm is based on three relatively independent components: luminance comparison, contrast comparison and structure comparison.

The luminance comparison is:

$$l(x, y) = \frac{2(1 + R)}{1 + (1 + R)^2 + \frac{C_1}{\mu_x^2}} \quad (1.11)$$

where R is the relative luminance, i.e. the luminance change with respect to the background luminance, μ is the mean intensity. The contrast comparison is:

$$c(x, y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2} \quad (1.12)$$

where σ_x and σ_y are standard deviations estimating the signal contrast. Finally, the structure comparison is:

$$s(x, y) = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3} \quad (1.13)$$

where C_1 , C_2 and C_3 are constants properly chosen. By combining the three comparisons, it is possible to define the SSIM between signals x and y as:

$$SSIM(x, y) = [l(x, y)]^\alpha \cdot [c(x, y)]^\beta \cdot [s(x, y)]^\gamma \quad (1.14)$$

where α , β and γ are appropriate positive weights. If $\alpha = \beta = \gamma = 1$ and $C_3 = C_2/2$, then the SSIM can be simplified in this way:

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \quad (1.15)$$

Figure 1.6 shows the block diagram of the SSIM quality assessment.

However, to have a single overall quality measure, for the entire image, a Mean

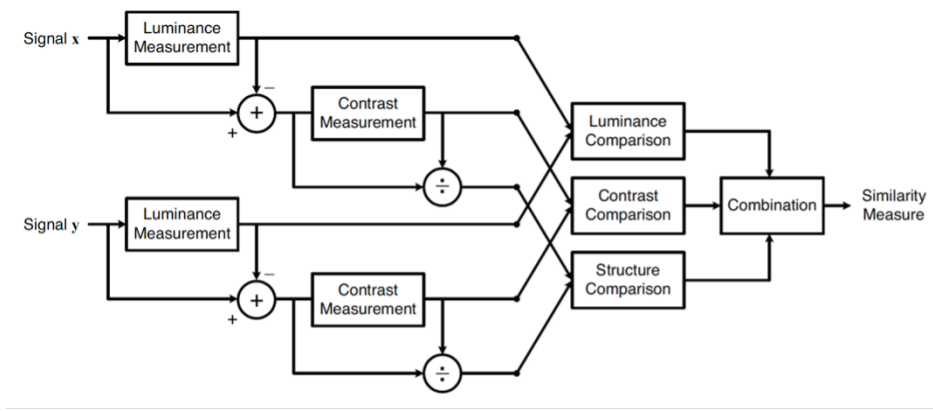


Figure 1.6: SSIM quality assessment diagram [158]

SSIM (MSSIM) index to evaluate the overall quality is introduced.

$$MSSIM(X, Y) = \frac{1}{M} \sum_{j=1}^M SSIM(x_j, y_j) \quad (1.16)$$

where X and Y are the reference and the distorted images, x_j and y_j are the image contents at the j -th local window and M is the number of local windows in the image.

The drawback of SSIM metrics is that they do not take into account image details at different resolutions and viewing conditions. To tackle this problem, in [162] a Multi-Scale SSIM (MS-SSIM) has been designed. Let $j = 1 \dots M$ be the possible different resolutions, then the MS-SSIM equation is:

$$SSIM(x, y) = [l_M(x, y)]^{\alpha_M} \cdot \prod_{j=1}^M [c_j(x, y)]^{\beta_j} [s_j(x, y)]^{\gamma_j} \quad (1.17)$$

where α_M , β and γ are constants that weigh the different components. Several variants of the SSIM have been proposed in the literature to consider different aspects for the visual quality assessment [161], [89], [159]. Another approach for

image quality assessment is the Most Apparent Distortion (MAD) algorithm [88], which is based on the assumption that HVS employs different strategies to determine the quality of an image. Hence, two approaches were derived: detection-based strategy and appearance-based strategy. In the former, when HVS views high-quality images, it goes beyond the image content, looking for distortions. In such a strategy, locations of visible distortions are first determined, then the perceived distortion is computed. In appearance-based strategy, when low-quality images are viewed, it tries to go over distortions and concentrates on the content of the image. This approach is based on local statistical difference maps.

Since there is no method that performs best in all situations, in [95] machine learning is leveraged to obtain a Multi-Method Fusion (MMF) approach that merges the results of multiple methods. Each method is weighed according to the training process of a regression approach.

Usually, Support Vector Machine (SVM) is employed as a regressor. During the training phase, a series of elementary quality metrics are considered for the training video dataset. The corresponding scores are normalised in the interval between 0 and 1 and then the final quality score is computed as the nonlinear combination of the elementary scores whose weights are given by the SVM regressor. After the training phase, the resulting weights are used for the video quality assessment. A further development based on SVM algorithms is the Video Multi-method Assessment Fusion (VMAF), which best reflects the user's perception of a video content [97], [99]. Also VMAF predicts subjective quality by combining multiple quality metrics through SVM regression. The elementary metrics fused are:

- Visual Information Fidelity (VIF) [143]: it is based on the assumption that quality is complementary to the measure of information fidelity loss;

- Detail Loss Metric (DLM) [90]: it measures the loss of details that affects the content visibility and the redundant impairment that distracts the viewer;
- Motion: it is the temporal difference between adjacent frames computed through the average absolute pixel difference for the luminance component.

The SVM regressor is then trained on a Netflix dataset and the resulting model shows visibly better results than other approaches in the literature.

1.4 Machine Learning

Machine Learning (ML) techniques are generally employed to learn from past data in order to reach a certain level of artificial intelligence. The field of ML encompasses statistical models and algorithms able to learn a task on the basis of data previously acquired and to obtain a certain performance.

In general, ML can be divided into three categories [19]:

- Supervised learning;
- Unsupervised learning;
- Reinforcement learning.

Supervised Learning

The goal of supervised learning is to learn a function able to map input data to specific groups. Equivalently, it can be said that the trainer creates labels to let the agent discover the relationship existing between inputs and labels. This category can be further split into regression problems and classification problems. In regression problems the variables associated with the labels are continuous and therefore represented by real numbers. If variables are discrete, then we talk about

classification. Examples of supervised learning algorithms can be linear regression [140], logistic regression [70], decision trees [128] or, as already said, SVM. Supervised learning techniques have been used to improve and predict the QoE in video streaming applications. In [27], such ML techniques are advocated to improve the so-called multidimensional QoE, i.e. a high number of inputs is used to evaluate the impact on playback stalls and initial delays on the QoE. In particular, based on the video length, the type of content, the length of the stalls, etc., regression techniques are used to get a mapping between the type of video and the corresponding MOS. After training the algorithm with a dataset, it is possible to predict the MOS related to a new video with a certain number of stalls in a more detailed way with respect to the aforementioned exponential model.

Another example can be found in [27], where the QoE is estimated using a linear regressor and an SVM to compare the performance. Vasilev et al. [154] considers three main factors to estimate the QoE:

- *Average video bitrate* of the downloaded segments;
- *Average video bitrate variation*: the standard deviation of the video bitrate; it keeps track of the quality changes of the downloaded segments;
- *Rebuffering ratio*: stalling time divided by the total video duration.

The goal is to predict these three factors on the basis of observable parameters associated with the QoS. To this purpose, the authors utilise a Bayesian Network [86] based on logistic regressors and a regression tree.

Unsupervised Learning

Unsupervised learning consists of creating labels that are not known a priori and then finds relationships between a set of data in input and these labels. This kind of learning is useful to understand possible distributions of data. One of the most

important unsupervised learning algorithms is the k-means clustering.

Such an algorithm was introduced by James MacQueen [98] and then applied to several fields ranging from image segmentation [43] to disease prediction [100] to network traffic classification problems [49].

The algorithm is based on the computation of centroids and its goal is to partition the n samples of a dataset in a prefixed number of disjoint subsets/clusters, say k . Centroids are the arithmetic mean points of these clusters and are initially chosen in a random fashion among the n samples to partition.

The k-means algorithm is formed by two steps that are iteratively repeated:

- Assignment step;
- Update step

In the assignment step, every point is assigned to the closest cluster, i.e. that cluster whose centroid has the minimum Euclidean distance from the considered point. Mathematically, if k is the number of clusters and m_j for $j = 1, \dots, k$ is the centroid of cluster i , it results:

$$S_i^{(t)} = \{x_p : \|x_p - m_i^{(t)}\|^2 \leq \|x_p - m_j^{(t)}\|^2 \forall j, 1 \leq j \leq k\} \quad (1.18)$$

where each x_p is assigned to only one cluster S . In the update step, all clusters recompute their centroids by taking into account the newly added data. The new centroids are obtained as follows:

$$m_i^{(t+1)} = \frac{1}{|S_i^{(t)}|} \sum_{x_j \in S_i^{(t)}} x_j \quad (1.19)$$

The algorithm stops once the assignments no longer change. Furthermore, such an algorithm does not guarantee convergence to the optimal solution but presents satisfactory results.

An alternative way to k-means is k-medoids. The main difference concerns the points representing the centres of the clusters. Instead of centroids, medoids are considered. A medoid is a point in a cluster whose distance from all the points in the same cluster is minimal. Medoids must necessarily coincide with some points belonging to the dataset, while in the previous approach centroids could be anywhere and so also not coincide with data points.

In [156], the authors point out that the relationship between the MOS and QoS parameters is still an open problem for the research. For this reason, the MOS prediction is provided by a model including k-means clustering and logistic regression. The latter requires that the independent variable is binary, therefore such a method is employed to classify the QoE as ‘good’ or ‘bad’. However, during the training phase, the MOS values can range between 1 and 5, which is why a k-means algorithm is implemented to classify the MOS values only as ‘good’ or ‘bad’ (therefore $k = 2$) and then the logistic regression algorithm is trained.

In [25] the QoE is defined as the quality of the service perceived by users and as a consequence it can result considerably different from user to user. To this end, the authors define a model able to autonomously identify user profiles on the basis of subjective feedback about the service provided. One of the general objectives is to understand how a user reacts in terms of QoE to the variation of QoS parameters. Once data are gathered, a clustering algorithm profiles users. The procedure aims at creating a user profiling module given a number of video sessions. A clustering algorithm called k-means++ [10] is employed to associate users giving a similar feedback in similar situations with the same group of users. The user profiling

module developed in [25] groups users on the basis of a couple of data: the series of parameters describing the QoS of user i for the session j for every time instant and the sequence of the related feedback. Thanks to this modelling, it has been possible to state that profiles describing users' behaviour can be expressed as the combination of three main elementary profiles that show the QoE variation as a function of a single QoS parameter. Basically, the elementary profiles are composed of:

- A user able to detect every change in the QoS and to affect his/her own QoE consequently;
- A user insensitive to the variation of a specific QoS parameter;
- A user who is either completely satisfied or completely unsatisfied.

1.4.1 Reinforcement Learning

Reinforcement Learning (RL) is usually employed for the control of dynamic systems. It finds practical applications especially when the system to be controlled is not known a priori or is partially known.

RL is a learning technique whose goal is to train an agent that interacts with the environment and that is able to control it through some actions performed on it. In other words, the agent's goal is to learn how to map the state of the environment to specific actions, which represent the control variable, to maximise a reward function. In fact, each action produces a reward whose goal is to encourage or discourage certain behaviours of the agent. RL is generally defined in terms of optimal control of Markov decision processes and can be implemented through different types of algorithms. Among these, the most important are temporal difference algorithms such as SARSA and Q-learning.

The State-Action-Reward-State-Action (SARSA) algorithm [136] has the goal of learning an optimal policy π , i.e. a function that defines the probability of choosing a certain action starting from a particular state.

This method is based on an evaluation and update step of the action-value function $Q_\pi(s, a)$, which indicates the expected reward starting from state s , performing action a and following the policy π thereafter, and on an improvement step of the policy π , in which the policy is updated on the basis of the new values of $Q_\pi(s, a)$. The Q-learning algorithm [163] is very close to SARSA but is off-policy, i.e. the evaluation and update step of the action-value function $Q(s, a)$ does not depend on the policy but approximates the optimal action-value Q^* .

One last important algorithm based on temporal difference is the *actor-critic* [148]. In this approach the agent is composed of two parts:

- The actor, whose task is to select the action and to implement it in the system;
- The critic, which updates the action-value function and ‘criticises’ the actions chosen by the actor on the bases of the reward received.

In the literature, RL techniques have recently found their application in the field of video streaming and QoE optimisation.

An example providing a video streaming application of such techniques is *Pensieve* [105], which is a system defining new ABR algorithms based on RL. Pensieve trains a neural network that chooses the right bitrate for future video chunks on the basis of the observations gathered by the client device. This way, no specific assumption is made on the environment, thus making Pensieve extensible to any context. Figure 1.7 shows how it is possible to employ RL for bitrate adaptation.

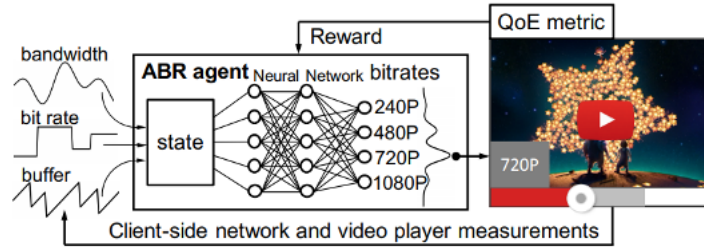


Figure 1.7: Pensieve scheme [105]

It is possible to notice that the policy, which decides the action to be performed, is not defined a priori, but is derived from a neural network. The agent gathers some metrics such as the buffer state, past decisions on the bitrate, etc., and feeds them to the neural network, which outputs the action to perform, i.e. the bitrate to choose to download the next video chunk. The agent's reward is the measured QoE, which is used to update the neural network's parameters.

More specifically, the authors implemented an actor-critic RL technique to compute the policy. The agent selects the actions according to a policy that is derived through an actor neural network having a manageable and tunable number of parameters θ . Such parameters are updated using the policy gradient method in order to find the optimal policy, which maximises the long-time reward. The policy gradient method estimates the gradient of the total expected reward based on the current policy. To compute this gradient, it is necessary to know the value function, i.e. the total expected reward obtained starting from a state s and following the current policy thereafter. Therefore, the critic network computes an estimate of such value function on the basis of the state and the rewards empirically observed until that moment.

It is worth noticing that such an approach is independent of the metric chosen to compute the QoE, i.e. the kind of reward of the RL algorithm. The authors consider three types of metrics, all coming from the following definition of QoE

as proposed in [169].

$$QoE = \sum_{n=1}^N q(R_n) - \mu \sum_{i=1}^N T_n - \sum_{n=1}^{N-1} |q(R_{n+1}) - q(R_n)| \quad (1.20)$$

Where N is the number of chunks, R_n is the bitrate of chunk n , $q(R_n)$ maps the bitrate to the quality perceived by the user, T_n is the rebuffering time and the last term penalises changes in the quality of the video chunks. Three choices for $q(R_n)$ were considered:

- QoE_{lin} : $q(R_n) = R_n$;
- QoE_{log} : $q(R_n) = \log(R/R_{min})$;
- QoE_{hd} : high definition videos are assigned with a higher QoE with respect to non HD videos.

Results show that, independently of the QoE metric adopted, Pnsieve is able to guarantee an average QoE always better than other state-of-the-art approaches, thus confirming the great advantage RL techniques bear.

Tian et al. [149] propose a technique called Deeplive that allows ABR algorithms running on users' devices to maximise the QoE related to video streaming through the use of Deep Reinforcement Learning (DRL). DRL is a combination of neural networks and RL.

The authors define the following QoE associated with a chunk:

$$QoE_{chunk}(i) = \mu_q \sum_k Q(k) + \mu_r \sum_k R(k) + \mu_l \sum_k L(k) + \mu_f \sum_k F(k) + \mu_s S(k) \quad (1.21)$$

Where k represents the k -th frame of the chunk, $Q(K)$ is the video quality of frame k , $R(k)$ is the rebuffering time, $L(k)$ is the end-to-end delay of the k -th frame, $F(k)$ is the frame skipping and $S(i)$ indicates whether a bitrate switch occurred. Notice that coefficients $\mu_q, \mu_r, \mu_l, \mu_f, \mu_s$ are used to define a particular QoE metric. The total QoE is:

$$QoE = \sum_{i=1}^I QoE_{chunks}(i) \quad (1.22)$$

On each user, Deeplive implements an agent employing a neural network to determine which action has to be performed to increase the QoE. In particular, the agent is composed of two neural networks: one used to predict the target QoE and the other to evaluate the actual QoE. On the basis of these data, the agent chooses the actions to perform. These actions consist of regulating the bitrate, the buffer and the latency through the use of a policy that maps the current state with the corresponding values associated with the three actions. Unlike Pensieve, in this case actions do not regulate only the download bitrate of the chunks.

Further improvements can be found in [76], where ABR algorithms are based on the training of two competing agents. This approach results to be very effective in the case the reward function is complicated and affected by several factors. Other techniques of deep learning and RL were explored to improve the QoE on DASH [59], [94], [30]. In particular, in [59] different learning-based architectures are presented and tested. Such architectures perform better than other state-of-the-art approaches in terms of QoE and exhibit a higher convergence rate. Also in [94], on the basis of an advanced deep Q-learning model and a subjective QoE based on the single video chunks, a new approach is obtained, able to guarantee fast convergence.

DRL-based control strategies are also among the most effective ways to achieve control objectives in nonlinear systems. Such systems are generally hard to control with conventional techniques that are model-based. DRL algorithms learn an optimal control policy by directly interacting with the environment. The downside concerning these kinds of algorithms lies in the absence of any stability guarantee. In other words, no stability certificate ensures asymptotic stability of the controlled nonlinear system. This poses an important problem for the safety of such control policies. Intuitively, one should try to find a stability certificate, i.e. a Lyapunov function, that proves the asymptotic stability of the system. However, finding Lyapunov functions is in general a hard task since there is no standard method to construct them. To this end, Neural Networks (NN) can be implemented to synthesise Lyapunov functions in an iterative way as explained in Chapter 4.

1.5 Telecommunication System Architecture

The work reported in Chapter 2 and Chapter 3 proposes functional architectures to solve problems related to video streaming systems. Such architectures need to be supported by specific technologies and protocols: in particular, Software Defined Networking (SDN) and network slicing. In the context of 5G infrastructures, the proposed architectures are perfectly compliant and can be implemented without any relevant modification. As explained by the 5G Infrastructure Public Private Partnership ¹ and the related documents, SDN is a technology that is implemented in the 5G infrastructure and allows the assignment of differentiated network resources to heterogeneous clients. Network slicing is also an already-existing solution in 5G networks and is regulated by 3GPP ² specifications.

¹<https://5g-ppp.eu>

²<https://www.3gpp.org>

1.6 Thesis Outline

The thesis is structured as follows:

Chapter 1 — introduces video streaming technology by providing a brief historical background and the main protocols employed. Particular attention is paid to Quality of Service (QoS) and Quality of Experience (QoE) along with an analysis of video quality metrics. A brief introduction of machine learning techniques applied to video streaming and to the control of dynamic systems is provided, too.

Chapter 2 — describes an optimal network resource allocation strategy to increase the QoE-fairness among users in the context of video content distribution. Multi-Commodity Flow Problem (MCFP) and clustering techniques are employed to solve the problem.

Chapter 3 — tackles the problem of synchronising users attending a live video streaming event through consensus with distributed event-triggered control. After modelling users as first-order integrators, classical consensus protocols and event-triggered control techniques are used to achieve consensus.

Chapter 4 — illustrates the issue of guaranteeing asymptotic stability for those systems controlled via Deep Reinforcement Learning (DRL) techniques. To this purpose, a Learner-Verifier approach is provided to synthesise proper Lyapunov functions able to prove the asymptotic stability of the controlled system.

Chapter 5 — provides discussions and conclusive remarks along with possible future research directions.

Part I

Video Streaming Systems

Chapter 2

Optimal QoE-fair Resource Allocation in Multipath Video Delivery Networks

In this Chapter, a Multi-Commodity Flow Problem (MCFP) optimisation framework is described to address the issue of designing a QoE-fair optimal allocation strategy in video delivery networks. Simulations show that this approach is able to remarkably increase fairness in terms of QoE among the users of a video streaming service while still keeping the average QoE unchanged. Results shown in the following have been published in three scientific papers [\[41\]](#), [\[40\]](#), [\[103\]](#).

2.1 Background and Related Work

An ever-increasing fraction of users prefers to consume video content over the Internet instead of using classical TV broadcast channels. To make their services

profitable, online video content providers aim at increasing the number of engaged users and preventing service abandonment. To this end, such services should be designed to provide users with the best possible QoE given the constraints set by the particular user device and the network.

The control architecture employed today by all leading video platforms (Netflix, Youtube, etc) decouples the problem into two non-cooperating sub-problems:

- video services control their delivery network to guarantee an optimal level of QoS by ensuring that parameters such as end-to-end network bandwidth, packet losses, and network latency meet specific requirements;
- concurrent users consume video contents through players that run Adaptive BitRate control algorithms (ABR) designed to dynamically select the video bitrate (and video resolution) from a discrete set \mathcal{L} to provide the best possible QoE given the user device features and the end-to-end network bandwidth [145].

This approach has the advantage of being fully decoupled and very simple to be implemented. However, some remarkable limitations are present. Since no communication between users is available, ABR algorithms running at the players selfishly try to improve their individual QoE. In addition, the architecture of current delivery networks is designed to provide concurrent users that share the same network resources (i.e. network links) with a fair share of network bandwidth. However, this QoS-fair distribution of network resources does not translate into equalizing the perceived quality. Suppose users have high resolution devices (f.i. Smart TVs). In that case, the video bitrate required to obtain a specific level of QoE might be considerably larger compared to the bitrate needed by devices with low resolution (e.g. smartphones).

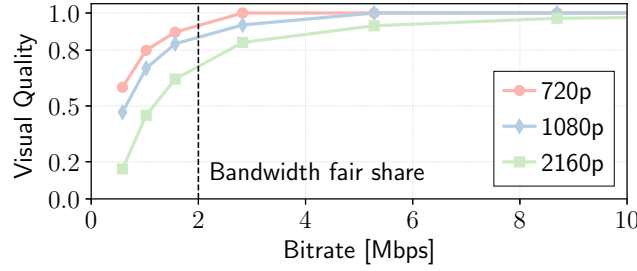


Figure 2.1: Visual quality function of the video bitrate and the client screen resolution

Let us consider the example in Figure 2.1, which shows the measured visual quality as a function of the encoding video bitrate obtained by clients with different screen resolutions. Let us assume that three concurrent users request the same video using clients with different screen resolutions (namely 720p, 1080p, and 2160p) and that the video flows share the same bottleneck link having a bandwidth equal to 6 Mbps. Then, the fair network bandwidth share obtained by each video flow is equal to 2 Mbps. As a result, the visual quality obtained by the 720p, 1080p, and 2160p clients would be respectively equal to 0.9, 0.85, 0.7. Therefore, it results that small screen devices will enjoy better visual quality with respect to large screen devices when given the same network bandwidth share. In other words, current video delivery networks do not provide a *fair* level of QoE to users. Consequently, video distribution networks, which allocate resources without taking into account the user's degree of satisfaction, cannot provide a fair level of QoE to users when the network resources become scarce. For this reason, video service providers should implement a QoE-aware network resource allocation strategy (as opposed to QoS-aware strategies) to assign a differentiated network bandwidth to video flows sharing the same bottleneck with the objective of equalizing the video quality obtained by heterogeneous devices. To this purpose, an interaction between video and network providers is needed. Software

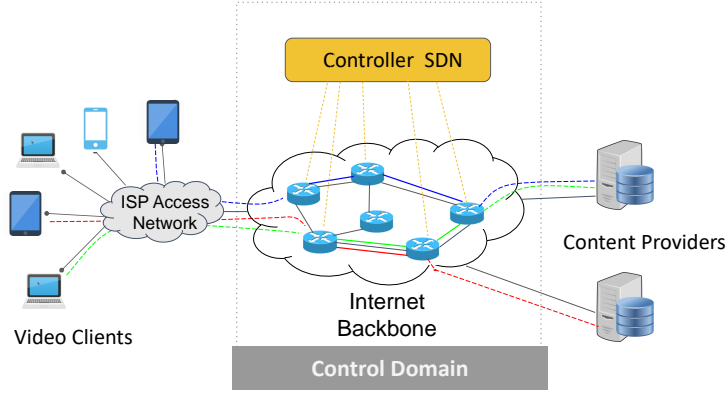


Figure 2.2: Distribution network

Defined Network (SDN) is a technology that allows such interaction through a centralised control plane, as shown in Fig. 2.2. This control plane directly controls the network nodes, i.e. the switches.

Traffic engineering techniques based on *network slicing* are the solution to achieve QoE-fairness. With these techniques, links in a network are sliced and assigned to subsets of video flows, depending on their characteristics. Computing the size of each slice is the objective of the optimisation problem. The aim is to consider a generic distribution network instead of focusing only on a single bottleneck case as studied by several authors [63], [34]. The Multi-Commodity Flow Problem (MCFP) optimisation framework is employed to achieve a QoE-fair distribution of network resources.

The state of the art is full of works that take into account the quality perceived by users (QoE) when addressing resource allocation problems in video streaming. However, as explained in Section 1.3.2, it is not an easy task to find an objective formula able to measure a user's QoE. Similarly, also client classes are not easily definable, since they are characterized by flow sizes, arrival rates, and channel statistics. However, in the following work clients are classified by their device

resolution only, thus avoiding active measurements on the network to be fed to the optimiser.

Throughout this chapter, we will consider a model of QoE defined by the visual quality perceived by the user. This is done under the realistic assumption that the ABR algorithm is designed to avoid rebuffering events. Nevertheless, several works employ more complete models of QoE including the impact of rebuffering events. In [51], for instance, the authors employ the SVR-QoE model and the NARX-QoE model. Both of them take into account the presence of rebuffering events and change the QoE function accordingly. Client-side metrics such as the rebuffering ratio in the utility function to be maximised have not been included in this work since this would have required continuous feedback being sent from all clients to the optimiser thus making the solution less scalable.

In [6], Ai et al. solve the resource allocation problem as a 0-1 programming problem by keeping into account the QoE and fairness among users. This approach entails a cross-layer design, i.e. the resource allocation is based on the information gathered from different layers of the network. The authors in [28] present an optimal QoE-aware scheduling for video segment selection in the framework of HTTP Adaptive Streaming (HAS). Such a technique also addresses rebuffering events avoidance and initial delay minimisation. In [131], the authors provide a solution to optimise the QoE of multiple video streaming sessions. In fact, the bandwidth is not equally allocated among competing flows but its allocation takes into account the content complexity of the requested video and the playout buffer status of the individual clients. However, in the aforementioned works, both clients and videos are not aggregated, thus implying an extremely heavy computational load that may result unmanageable.

The importance of a Software Defined Networking (SDN) architecture is shown

in [85], where a Video Control Plane (VCP) is introduced to allow cooperation between clients and the delivery network. In this scenario, the Dynamic Adaptive Streaming over HTTP (DASH) players are required to cooperate with the Service Manager to obtain an optimal bitrate while in our work clients are not involved in the optimal bandwidth distribution. Additionally, in [85], each active player is assigned an equal bandwidth share with no regard to the device resolution. In the present work, the network bandwidth is not the same for each user since the type of video and device resolution are kept into account.

Samani and Wang [137] propose MaxStream, which is an SDN-based flow maximisation framework based on two Integer Multi-Commodity Flow (IMCF) problem formulations: the first problem, Most-flows IMCF, selects the maximum number of streaming sessions that improve providers' revenue, while the second problem, Maximum-ICMF, selects the paths maximising the bitrate for the considered streaming sessions. In this case, there are some flows that will be rejected and therefore not all session demands will be satisfied. Moreover, the authors consider only a single-path resource allocation strategy.

Multipath routing is a deeply studied research problem. In this work, multipath traffic engineering is preferred to single-paths because of routing robustness, low latency and load balancing for better performance [87]. A well-known property in multipath routing optimisation states that the maximum number of actually utilized paths is limited by the number of session demands (D) plus the number of links (L) [127]. Consequently, when dealing with multipath routing problems, the optimal solution is achieved considering at most $D + L$ paths, where $D + L$ represents an upper bound.

In [63], Georgopoulos et al. propose for the first time a solution to deliver a fair level of QoE to users by slicing shared bottlenecks through a Software Defined

Networking switch. Each video session is assigned to one network slice whose size is obtained by solving a max-min fairness problem. However, the work is limited to single bottlenecks and cannot scale to a large number of users. We now focus on the relevant literature concerning client-side strategies to allocate resources in video streaming. Bentaleb et al. [13] describe the advantage of a client-side solution for the resource allocation problem in the network using SDN to obtain higher scalability and per-client QoE.

In [14], the same authors provide an improvement to communication overhead and client heterogeneity called SDNHAS, which is an intelligent streaming architecture helping HAS players to make efficient adaptation decisions. This work also presents a clustering of the clients that allows a large-scale network implementation but does not take into account a grouping of videos. SDN is considered also in [48] along with a mixed integer linear program for optimal network resource allocation in live video streaming but without taking into account QoE-fairness.

In [32], a hybrid control system for video bitrate maximisation, playback interruptions avoidance, and video bitrate switches minimisation is developed. Such a type of control affects positively the QoE since it optimises the video bitrate and avoids rebuffering events in the case of a single bottleneck. The same authors in [34] compare different Adaptive Bitrate Algorithms (ABR) and analyse two possible allocation strategies: network slicing (or bandwidth reservation) and bitrate guidance. The first strategy assigns video flows to network slices whose size is determined by solving an optimisation problem; the second strategy employs DASH Assisting Network Elements (DANes) to guide video clients in the choice of the video level. The paper shows that the bandwidth reservation strategy provides better results in terms of achievable video fairness. However, no stress is given to the concept of client classes and video clustering. Additionally,

QoE-fairness is not considered in the optimisation problem. Recently, machine learning techniques have been employed to address the problem of QoE-fairness. Altamimi and Shirmohammadi [8] propose a server-side QoE-fair rate adaptation method that uses Reinforcement Learning to select the best bitrate for each client. This approach implies a cooperation between clients sharing a bottleneck link and their server, which modifies the Media Presentation Description (MPD) files to regulate the available bitrate at one client. In [78], a Q-learning based bandwidth allocation algorithm called Q-FDBA is implemented. The authors adopt a centralised approach based on SDN framework and test it on a single bottleneck with three players. At this point, we remark that our approach is implemented in a realistic network scenario with several thousands concurrent users and potentially more than one bottleneck.

It is important to point out that in all aforementioned works the resource allocation is not decoupled from ABR algorithms running at the clients. In addition, such works also consider the clients' buffer occupancy and the video level selection in the optimisation problem.

2.2 The Multi-Commodity Flow Problem

In this section, we briefly introduce the *multi-commodity flow problem* (MCFP), i.e. the optimisation framework that we leverage to design the proposed QoE-fair network bandwidth allocation strategy. The term commodity refers to the tuple composed of a source node, a destination node, and a volume, i.e. the resources required to satisfy the commodity. Referring to the network bandwidth allocation problem considered here, a commodity identifies a video session where the source node is the video server, the destination node is the client, and the volume represents the video bitrate required to achieve the maximum video quality. In

general, the MCFP aims at maximising a properly defined utility function with a set of constraints in order to guarantee a network resource allocation so that all the commodities are optimally satisfied.

The MCFP will be described using the *link-path formulation* and the terminology provided in [127]. Let us represent the delivery network with a capacitated graph $G = (\mathcal{N}, \mathcal{E})$, where $\mathcal{N} = \{n_1, n_2, \dots, n_N\}$ is the *node* set and $\mathcal{E} = \{e_1, e_2, \dots, e_E\}$ is the *edge* set. Each edge or link $e \in \mathcal{E}$ is identified by a node pair and has a capacity c_e expressed in terms of bandwidth. The commodities related to the delivery network can be represented by the set of *demands* $\mathcal{D} = \{1, 2, \dots, D\}$, where each demand $d \in \mathcal{D}$ identifies a source-destination node pair and the corresponding *traffic volume* H_d , i.e. the required network bandwidth needed by that demand. Furthermore, a demand d can be satisfied, i.e. it receives sufficient bandwidth, through a set of admissible paths \mathcal{P}_d where each path $p \in \mathcal{P}_d$ connects the source node to the destination node of the demand. All paths contained in \mathcal{P}_d are computed off-line and represent the shortest paths connecting the source node to the destination node of demand d . As a consequence, the demand volume H_d is split in *path flows* routed on paths belonging to \mathcal{P}_d , where each path flow is denoted with x_{dp} ($p \in \mathcal{P}_d$). The MCFP aims at optimising such path flows—by means of a proper utility function—with two constraints. The first imposes that all commodities must be taken into account and therefore must be brought from their sources to their destination; the second requires that the total flow on each edge must not be greater than the maximum edge capacity.

Let us assume, without loss of generality, that the graph is fully connected, i.e. there always exists a path connecting each possible pair of nodes in \mathcal{N} . These nodes are represented by network switches in our problem, whereas edges identify links connecting a couple of switches. According to what has been just stated,

in the following, nodes and SDN switches will be referred to interchangeably as well as edges with links. *Bandwidth slices* of an appropriate size are obtained by dividing each link. The number of slices depends on the demands in the network. To compute the size of bandwidth slices, a multi-path weighted α -fairness optimisation problem is solved by employing the following utility function [110]:

$$U(X) = \begin{cases} \sum_d w_d \log X_d & \text{if } \alpha = 1 \\ \sum_d w_d \frac{X_d^{1-\alpha}}{1-\alpha} & \text{otherwise} \end{cases} \quad (2.1)$$

where $X_d = \sum_p x_{dp}$ is the total bandwidth (or total flow) allocated to demand d , $X = [X_1, X_2, \dots, X_D]^T$ is the vector of the total bandwidths for each demand and w_d is a *weight* associated to the demand d . It has been shown that the maximisation of (2.1) provides a balance between link utilisation (which is related to the solution *efficiency*) and fairness, by varying the scalar parameter α in the interval $[0, +\infty]$ [110]. $\alpha = 0$ implies that the link utilisation is maximised without keeping into account the fairness among flows, whereas if $\alpha \rightarrow +\infty$, the flow assignment becomes *max-min* fair, i.e. the assignment allocates resources such that the flow obtaining the minimum rate is maximised. When $\alpha = 1$, the optimisation problem results in the so called *Proportional Fairness* (PF) [117], which has the advantage of providing a good trade-off between fairness and link utilisation. Therefore, the proportional fair case ($\alpha = 1$) has been considered thus leaving to future studies a performance comparison for different values of α . In the PF case, it results that $U(X) = \sum_d w_d \log X_d$. Let us now derive the optimisation problem that has been considered for this work. To this end, let us start with the single-path case, i.e. the easier case in which the demand volume H_d can be routed only on one possible path connecting the source node to the destination node; in other words $|\mathcal{P}_d| = 1 \forall d \in D$ and $X_d = x_d$. We will then pass onto the analysis of the multi-

path optimisation problem in order to carry out a comparison of the performance of the two problems in terms of QoE-fairness.

MCFP single-path weighted proportional fair optimisation problem:

$$\text{Maximise } \sum_d w_d \log x_d \quad (2.2)$$

$$\text{s.t. } \sum_d \delta_{ed} x_d \leq c_e, \forall e \in \mathcal{E} \quad (2.3)$$

$$x_d \leq H_d \quad (2.4)$$

In (2.3), δ_{ed} is the link-path indicator, which is equal to 1 if the demand d uses the link e , otherwise it is set to 0. Constraints (2.3) are imposed to respect the capacity of each link c_e , i.e. the sum of all the path flows x_d using link e should not exceed the capacity of that link. The last constraints (2.4) ensure that the total bandwidth x_d allocated for demand d is bounded by the demand traffic estimation given by H_d . This condition depends on the network traffic load: in the best case (no network bottlenecks) the total bandwidth x_d allocated to the demand d should be equal to the requested traffic volume H_d . However, since bottlenecks are generally present, the total bandwidth x_d allocated to a demand is less than H_d . Notice that this constraint ensures that a demand d is assigned with a bandwidth x_d that is not greater than H_d , thus avoiding a possible waste of bandwidth.

The optimisation problem shown above is convex since the objective function is convex and the constraints are linear. Thus, the solution is represented by a unique global maximum that could be achieved either at one single point or at a convex set of feasible points ([127], [18]).

By leveraging the theory contained in [127], let us provide a better understanding of the relationship between the optimisation variables and the parameters in-

volved. We start from the easier case of a single-path *uncapacitated* problem, i.e., when link capacities are to be sized, and then we consider the case when link capacities are given (*capacitated* problem).

Proposition 1 (Uncapacitated Case). *Suppose that the link capacities c_e of the network are not given a priori but need to be sized and that we want to set the total network capacity to a value Z , i.e., $\sum_e c_e = Z$, where Z is a constant. If constraints (2.4) are always satisfied, then denoting with $x^* = [x_1^*, x_2^*, \dots, x_D^*]^T$ the optimal solution of Problem (2.2)-(2.4) and with $F(x)$ the objective function in (2.2), i.e. $F(x) = \sum_d w_d \log x_d$, it results that:*

$$F(x^*) = (\log Z) \sum_d w_d - \sum_d w_d \log \left(\sum_e \delta_{ed} \right) + \sum_d w_d \log w_d - \left(\log \left(\sum_d w_d \right) \right) \sum_d w_d$$

$$x_d^* = \frac{Z}{\left(\sum_e \delta_{ed} \right)} \frac{w_d}{\left(\sum_d w_d \right)} \quad (2.5)$$

Proof. Let us start by noticing that in the optimal case, the inequalities related to (2.3) become equalities, so it follows that $\sum_e \sum_d \delta_{ed} x_d = Z$. If we then suppose that constraints in (2.4) are always satisfied, the Lagrangian function associated with the optimisation problem is:

$$L(x, \sigma) = - \sum_d w_d \log x_d + \sigma \left(\sum_e \sum_d \delta_{ed} x_d - Z \right)$$

where σ denotes the Lagrangian multiplier. The function can be rewritten as follows:

$$L(x, \sigma) = \sum_d \left(\left(\sigma \sum_e \delta_{ed} \right) x_d - w_d \log x_d \right) - \sigma Z$$

In this way, we can define the dual objective function:

$$W(\sigma) = \min_{x \geq 0} L(x, \sigma)$$

By fixing σ , differentiating w.r.t. $x_d(\sigma)$ and setting the resulting derivatives equal to zero, we obtain:

$$\frac{w_d}{x_d(\sigma)} - \sigma \sum_e \delta_{ed} = 0$$

that produces:

$$x_d(\sigma) = \frac{w_d}{\sigma \sum_e \delta_{ed}}$$

If we substitute the previous expression in $L(x, \sigma)$, it results that:

$$W(\sigma) = \sum_d (w_d - w_d \log \frac{w_d}{\sigma \sum_e \delta_{ed}}) - \sigma Z$$

In order to compute the Lagrangian multiplier σ , we differentiate $W(\sigma)$ and find the stationary point:

$$\sum_d \frac{w_d}{\sigma} - Z = 0 \Rightarrow \sigma^* = \sum_d \frac{w_d}{Z}$$

from which $F(x^*)$ and x_d^* follow.

Let us now take into account constraints (2.4). In this case we get:

$$x_d(\sigma) = \begin{cases} \frac{w_d}{\sigma \sum_e \delta_{ed}} & \text{if } 0 < \frac{w_d}{\sigma \sum_e \delta_{ed}} \leq H_d \\ H_d & \text{if } \frac{w_d}{\sigma \sum_e \delta_{ed}} > H_d \end{cases}$$

The threshold value for σ is:

$$\bar{\sigma}_d = \frac{w_d}{H_d \sum_e \delta_{ed}}$$

Therefore, $x_d(\sigma)$ can be rewritten as:

$$x_d(\sigma) = \begin{cases} \frac{w_d}{\sigma \sum_e \delta_{ed}} & \text{if } \sigma \geq \bar{\sigma}_d \\ H_d & \text{if } 0 \leq \sigma < \bar{\sigma}_d \end{cases} \quad (2.6)$$

Let us now sort all $\bar{\sigma}_d$'s in non-decreasing order and if some of the elements are equal, we keep just one of them until we get the sequence (s_1, s_2, \dots, s_n) s.t. $s_1 < s_2 < \dots < s_n$, where we set $s_1 = 0$ and $s_n = +\infty$. Then, we can form $n - 1$ intervals $[s_1, s_2], [s_2, s_3], \dots, [s_{n-1}, s_n]$ and for each of them we can define two disjoint sets of demands:

$$F_j = \{d : x_d(\sigma) = \frac{w_d}{\sigma \gamma_d} \text{ for } \sigma \in [s_j, s_{j+1}]\}$$

$$U_j = \{d : x_d(\sigma) = H_d \text{ for } \sigma \in [s_j, s_{j+1}]\}$$

where $\gamma_d = \sum_e \delta_{ed}$ and $F_j \cup U_j = \{1, 2, \dots, D\}$ for $j = 1, 2, \dots, n - 1$. For each j identifying the interval $[s_j, s_{j+1}]$, the set F_j contains all the demands whose associated flow x_d is such that $x_d \leq H_d$ when $\sigma \in [s_j, s_{j+1}]$. On the contrary, U_j is the set of the demands whose flows satisfy $x_d \geq H_d$ when $\sigma \in [s_j, s_{j+1}]$. This

allows us to partition the demands in each interval according to constraints (2.4) and as a result, in each $[s_j, s_{j+1}]$, the dual function can be written as follows:

$$W(\sigma) = \sum_{d \in F_j} (w_d - w_d \log \frac{w_d}{\sigma \gamma_d}) - \sigma (Z - \sum_{d \in U_j} \gamma_d H_d) - \sum_{d \in U_j} w_d \log H_d$$

This function is continuous and differentiable and in the interval $[s_j, s_{j+1}]$ its first derivative is:

$$W'(\sigma) = Z - \sum_{d \in U_j} \gamma_d H_d + \sum_{d \in F_j} \frac{w_d}{\sigma} \quad (2.7)$$

For any point s_j , $j = 2, \dots, n-1$, and given the expression in (3.13), it results that for all demands d changing set from U_j to F_j and vice versa in s_j , the equations $H_d = \frac{w_d}{s_j \gamma_d}$ hold. Then, when (2.7) is evaluated in s_j , d belongs both to U_j and F_j and the terms $\gamma_d H_d$ and $\frac{w_d}{s_j}$ cancel each other. The same happens for $W'(\sigma)$ associated to the interval $[s_{j-1}, s_j]$. Therefore, the left and right derivatives of the dual function are equal in each point s_j , $j = 2, \dots, n-1$, thus implying the differentiability. However, the dual function is not differentiable twice since the second derivative is in general discontinuous at the ends of the intervals because it is equal to:

$$W''(\sigma) = - \sum_{d \in F_j} \frac{w_d}{\sigma^2}$$

Nevertheless, this implies the concavity of the function, which is differentiable. As a consequence, the maximum can be computed as $W'(\sigma) = 0$, $\sigma \in [0, +\infty)$ and the resulting stationary point σ^* can only belong to one of the intervals. Only in such an interval, the resulting stationary point of the dual function associated to

that interval actually belongs to the interval. This property does not hold for the stationary points computed in the other intervals. For each other interval $[s_j, s_{j+1}]$, the resulting σ^* does not belong to the interval. Therefore, the stationary point is given by:

$$\sigma^* = \begin{cases} \sum_{d \in F_j} \frac{w_d}{Z - \sum_{d \in U_j} \gamma_d H_d} & \text{if } F_j \neq \emptyset \\ \text{any } \sigma \in [0, +\infty) & \text{if } F_j = \emptyset \\ & \text{and } Z = \sum_{d \in U_j} \gamma_d H_d \\ \text{does not exist} & \text{if } F_j = \emptyset \\ & \text{and } Z \neq \sum_{d \in U_j} \gamma_d H_d \end{cases}$$

Once we get σ^* , we can compute the optimal demand flows through (3.13). \square

The previous analysis was carried out under the assumption that link capacities are to be sized. When link capacities are fixed, it is no longer guaranteed that the optimisation problem admits an optimal solution such that all constraints in (2.3) are equalities. In this case, it is not possible to obtain a closed-form expression of the solution. On the other hand, if an optimal solution exists such that all constraints in (2.3) are equalities, then we can prove the following.

Proposition 2 (Capacitated case). *Suppose that the link capacities c_e of the network are given a priori. If constraints (2.4) are always satisfied and there exists an optimal solution $x^* = [x_1^*, x_2^*, \dots, x_D^*]^T$ to Problem (2.2)-(2.4) such that all constraints in (2.3) are equalities, then it results that*

$$x_d^* = \frac{w_d}{\sum_e \delta_{ed} \sigma_e(c_1, \dots, c_{E_p}, w_1, \dots, w_D, \delta_{11}, \dots, \delta_{E_p D})}$$

with

$$\sum_e \delta_{ed} \sigma_e (c_1, \dots, c_{E_p}, w_1, \dots, w_D, \delta_{11}, \dots, \delta_{E_p D}) > 0$$

where $e \in \mathcal{E}_p$, \mathcal{E}_p is the set of all the edges that serve at least one demand, $E_p = |\mathcal{E}_p|$, and $\sigma_i(c_1, \dots, c_{E_p}, w_1, \dots, w_D, \delta_{11}, \dots, \delta_{E_p D})$ means that each Lagrangian multiplier σ_i , $i = 1, \dots, E_p$, depends in general on the link capacities $c_e, \forall e \in \mathcal{E}_p$, on the demand weights $w_d, \forall d \in \mathcal{D}$, and on $\delta_{ed}, \forall e \in \mathcal{E}_p, d \in \mathcal{D}$. It also appears clear that the higher the weight w_d associated to a demand d , the higher the bandwidth allocated to that demand by the optimisation problem.

Proof. Since now we have E_p constraints given by (2.3), which we consider to be equalities by hypothesis, the Lagrangian function becomes:

$$L(x, \sigma) = \sum_d \left(\sum_e \sigma_e \delta_{ed} x_d - w_d \log x_d \right) - \sum_e \sigma_e c_e$$

where the σ_e 's denote the Lagrangian multipliers. By fixing all the σ_e 's, differentiating w.r.t. $x_d(\sigma)$ and setting the resulting derivatives equal to zero, we obtain:

$$x_d(\sigma) = \frac{w_d}{\sum_e \delta_{ed} \sigma_e}$$

for each demand $d \in \mathcal{D}$.

By substituting $x_d(\sigma)$ in $L(x, \sigma)$ and differentiating w.r.t. each σ_e , we obtain a system with E_p equations in E_p unknowns. Such a system admits a solution—for the hypotheses made in Proposition 2—represented by the σ_e 's, $e = 1, \dots, E_p$, that depend on the link-path indicators, on the demand weights and on the link capacities. Obviously, it must result that:

$$\sum_e \delta_{ed} \sigma_e > 0, \forall d \in \mathcal{D}$$

Notice that if we include constraints (2.4), there is no possibility of finding a closed-form expression of the solution. \square

Let us now analyse the more general case of a multi-path weighted proportional fair optimisation.

MCFP multi-path weighted proportional fair optimisation problem:

$$\text{Maximise } \sum_d w_d \log\left(\sum_p x_{dp}\right) \quad (2.8)$$

$$\text{s.t. } \sum_d \sum_p \delta_{edp} x_{dp} \leq c_e, \forall e \in \mathcal{E} \quad (2.9)$$

$$X_d \leq H_d \quad (2.10)$$

where $X_d = \sum_p x_{dp}$ is the total bandwidth allocated for demand d . Also in this case, constraints (2.9) are imposed to respect the capacity of the link c_e , i.e. the sum of all the path flows x_{dp} insisting on the link e should not exceed the capacity of that link. Constraints (2.10) ensure that X_d is bounded by the demand traffic estimation given by H_d .

It is straightforward to show that also Problem (2.8)-(2.10) is convex, thus implying the existence of a unique global maximum. However, it appears that deriving a closed-form solution in any case in a multi-path scenario is not possible due to the complexity of the problem posed.

To provide a better understanding of the resource allocation process in multi-path

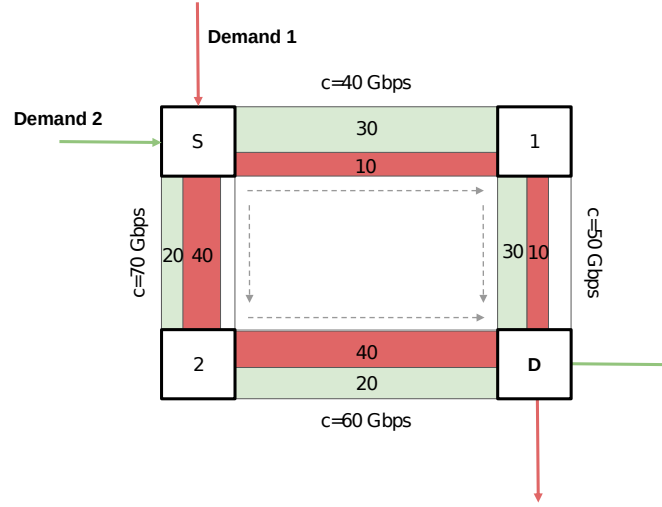


Figure 2.3: Resource allocation in a sample network

video delivery networks, let us briefly discuss the basic delivery network shown in Fig. 2.3. In this case, the switch at the source node S has to satisfy two demands coming from the destination switch D , both of which have a volume of 50 Gbps. Clearly, the paths connecting node S to node D are two: $p_1 = \{S, 1, D\}$ and $p_2 = \{S, 2, D\}$. Thus, the MCFP could use both of them to transmit the video flows composing a demand. In the example provided, the solution of the MCFP for demand 1 takes two slices: the first belonging to path p_1 with an assigned network bandwidth equal to 10 Gbps and the other to path p_2 with a bandwidth equal to 40 Gbps. In a similar fashion, demand 2 is assigned with a 30 Gbps slice on path p_1 and 20 Gbps on path p_2 . An important observation is that the MCFP chooses the appropriate path based on the bandwidth required by each flow composing a demand. For this reason, the size of a slice associated with a particular path has to take into account the available bandwidth of each link composing that specific path according to constraint (2.9).

2.3 The Proposed Approach

At this point, the proposed control strategy to operate a QoE-fair network resource distribution among concurrent heterogeneous users can be presented. To this end, we show how to employ the MCFP (Problem (2.8)-(2.10)) to achieve such a goal. This entails the design of demand weights w_d so that the maximisation of (2.8) results in a QoE-fair resource allocation. Such demand weights are computed through a utility function mapping the relationship between the network bandwidth assigned to a video session and the obtainable visual quality. In the following, we provide some definitions useful to understand the problem tackled.

2.3.1 Definitions

Given a video catalog $\mathcal{V} = \{v_1, \dots, v_V\}$, the DASH standard encodes each video $v \in \mathcal{V}$ into different representations or *levels* $l \in \mathcal{L}_v$ that can be identified by the couple $l = (b, r)$, where $b \in \mathcal{B}_v$ is the encoding bitrate and $r \in \mathcal{R}_v$ is the video resolution. Different videos can present remarkably different sets of encoding bitrate depending on the video content. At the client side, the ABR algorithm dynamically selects the video level $l \in \mathcal{L}_v$ according to the network bandwidth currently available on the path connecting the user to the video server. Even though we do not focus on a specific ABR algorithm, we make the sensible assumption that the control algorithm selects a video level whose bitrate b matches on average the average end-to-end path bandwidth. This is a nonrestrictive assumption since all well-designed ABR algorithms are in practice implemented in this way (see for instance [33]).

Let us now provide the first important definition concerning a *video request* t ,

which is a couple (v, c) , where $v \in \mathcal{V}$ and c is the *user class* belonging to the set $\mathcal{C} = \{c_1, c_2, \dots, c_C\}$. In order to classify users, parameters having an impact on the obtainable QoE should be considered. Since the screen resolution is one of the most important elements affecting the QoE, we propose to classify users based on their maximum screen resolution. It is possible that two devices have a different screen size, such as a phone and a TV, but the same maximum screen resolution. In this case the screen size can affect the QoE, especially when a video is streamed at a low bitrate. In the present work, we assume that each user in a client class has the same screen size, and leave to future studies the optimisation of the QoE also w.r.t. the screen size as highlighted in [52], [64], [54]. As a consequence, the terms “user class”, “user screen resolution” and “user screen size” are used interchangeably in this study. Notice that, with this notation, a video request t denotes which video v a user having a client resolution c is willing to watch.

For each video request $t = (v, c)$, the set \mathcal{L}_t is defined as $\mathcal{L}_t = \{l \in \mathcal{L}_v : r \leq c\} \subseteq \mathcal{L}_v$. In other words, \mathcal{L}_t contains the levels of \mathcal{L}_v whose resolution is not higher than c . Furthermore, clients having a screen resolution equal to c are assumed to request video levels whose resolution is not higher than c . Consequently, given a video request t , the ABR algorithm actually chooses only the levels contained in the set \mathcal{L}_t . Notice that this is how ABR algorithms work in practice. Although they can choose among all possible video levels, those having resolutions higher than the user screen resolution are usually never selected since it would increase the bandwidth consumption without producing a perceivable improvement in terms of visual quality.

It is now immediate to assign to each video request t its *reference level* $\bar{l}_t = (\bar{b}_t, c) \in \mathcal{L}_t$ as the representation with resolution c having the maximum bitrate \bar{b}_t , which indicates the minimum bitrate necessary to obtain visual quality equal to

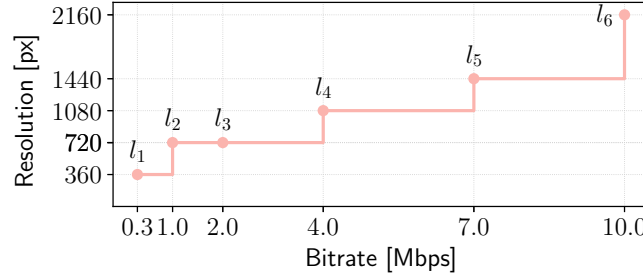


Figure 2.4: Video level set representation in a *ladder* graph

1. As an example, let us consider a 4K video v (resolution 2160p) being encoded into six video levels $l = (b, r) \in \mathcal{L}_v$ as shown in Fig. 2.4, where $\mathcal{L}_v = \{(0.3, 360), (1, 720), (2, 720), (4, 1080), (7, 1440), (10, 2160)\}$ (the bitrate is expressed in Mbps). If we consider a video request $t = (v, 720p)$, i.e. a user with a 720p screen requesting the video v , then the level set \mathcal{L}_t and the reference level \bar{l}_t would be respectively $\mathcal{L}_t = \{(0.3, 360), (1, 720), (2, 720)\}$ and $(2\text{Mbps}, 720p)$.

A *video session* is the tuple $(\text{src}, \text{dst}, t)$, where $\text{src} \in \mathcal{N}$ is the switch the server delivering the requested video is connected to; $\text{dst} \in \mathcal{N}$ is the switch the client is connected to; $t = (v, c)$ is the video request.

Finally, we are ready to define the demand d as the aggregate of the video sessions represented by the same tuple $(\text{src}, \text{dst}, t)$. In other words, a demand d contains all the video sessions from the same source node src to the same destination node dst associated with clients with the same video resolution c and requesting the same video content v . Consequently, if there are n_d video sessions with the same tuple $(\text{src}, \text{dst}, t)$, the demand volume H_d is equal to $n_d \bar{b}_t$, where \bar{b}_t is the bitrate of the reference level \bar{l}_t defined above. H_d can be interpreted as the minimum amount of network bandwidth that has to be allocated to the aggregate of the n_d video sessions composing the demand d so that each of these video sessions is served with a bandwidth share \bar{b}_t . It is straightforward to see that, in this case, if constraint (2.10) is strictly verified (i.e. $X_d = H_d$), it results that all video flows belonging

to this demand will enjoy the maximum visual quality possible. Conversely, in cases when the delivery network is overloaded, it might occur that the solution of the MCFP leads to $X_d < H_d$ for some demands. In such cases, video sessions belonging to those demands will obtain a bandwidth share less than the bitrate associated with the reference level \bar{l}_t . This means that the ABR algorithm will select a video level with a lower resolution, thus obtaining a degraded video quality. In the following, we describe how to measure the visual quality as a function of the allocated network bandwidth share.

2.3.2 Measuring the visual quality

The main goal of our resource allocation strategy is achieving a fair level of QoE among users. Such an objective is reached through the allocation of network resources using a multi-path approach. For this reason, a mapping between the allocated network bandwidth related to a video session and the achieved QoE is needed [53], [24]. We will use this mapping as a base to design appropriate demand weights w_d that allow to solve Problem (2.8)–(2.10) by allocating the network bandwidth based on the users' obtainable visual quality.

Notice that the procedure described in the following should be performed off-line each time a video is added to the catalog \mathcal{V} . At the end of this procedure, we will obtain a number of mappings equal to the number of defined user classes for each video. The resulting mappings will be associated to the corresponding video as metadata.

The visual quality of a video $v \in \mathcal{V}$ is measured as follows: for each video $v \in \mathcal{V}$, level $l \in \mathcal{L}_v$, and user class $c \in \mathcal{C}$, a mapping denoted as $Q_t : \mathcal{L}_v \mapsto [0, 1]$ is computed, which relates the video level to the corresponding visual quality when the video is played on a device with resolution c . The procedure is shown in Algo-

Algorithm 1 Visual quality measurement for a video

```

1: for each client class  $c \in \mathcal{C}$  do
2:    $t \leftarrow (v, c)$ 
3:   Select reference level  $\bar{l}_t$  from  $\mathcal{L}_t$ 
4:   for each  $l \in \mathcal{L}_v$  do
5:     if  $l \in \mathcal{L}_t$  then
6:        $\tilde{l} \leftarrow \text{Upscale } l \text{ to } c \text{ resolution}$ 
7:        $Q_t(l) \leftarrow \text{FRVQ}(\tilde{l}, \bar{l}_t)$ 
8:     else
9:        $Q_t(l) \leftarrow 1$ 
10:    end if
11:  end for
12: end for

```

rithm 1 and the output for a sample video is displayed in Fig. 2.1 in the case of a level set composed of 7 elements and a client class set $\mathcal{C} = \{720p, 1080p, 2160p\}$, which contains some of the most common device resolutions. After fixing the video v and the client class c (Line 2), we compute $Q_t(l)$ for each $l \in \mathcal{L}_v$ as follows (Lines 4–11). First, we select the reference video level \bar{l}_t from the set \mathcal{L}_t as described in Section 2.3.1. Then, for each video level $l \in \mathcal{L}_v$, the video quality is computed using a Full-Reference Video Quality (FRVQ) assessment tool such as the Structural SIMilarity (SSIM) [158], the Peak Signal to Noise Ratio (PSNR), or the Video Multi-method Assessment Fusion (VMAF) [91], normalized in the range $[0, 1]$. These tools estimate the visual quality by comparing each frame of a *degraded* video with the *reference* frames of the non-degraded video. This operation is performed in Lines 6–7. In particular, to obtain the degraded video \tilde{l} (Line 6), the video level l is up-scaled to the client device resolution. This reflects the way actual video streaming players typically work: if the user device has a given screen resolution (say 1080p) the ABR algorithm will select video representations characterized by a resolution up to the client screen resolution (i.e. 1080p in the example). The rationale is that rendering on the screen a video having a

resolution higher than the screen resolution would not lead to an improved video quality. This is particularly evident in Fig. 1 where the visual quality (measured as VMAF score) is shown to saturate at 1 when the video representation has a resolution higher than the one of the user screen device. Then, the FRVQ assessment tool estimates the video quality by comparing the degraded video \tilde{l} with the reference video \bar{l}_t (Line 7). This estimation process captures exactly what happens during video playback. In fact, the video player has to upscale the decoded video to the device screen resolution if the client screen resolution is higher than the video resolution served by the content provider, leading to a degradation in terms of perceived video quality and user QoE. Conversely, when the user is provided with a video resolution equal to his device resolution, no upscaling is needed and the user perceives the best visual quality experience. This situation is taken into account by Line 9. In this case, the video level l does not belong to \mathcal{L}_t , i.e. if the resolution of l is larger than that of the reference level \bar{l}_t , the video quality is set to 1.

2.3.3 Demand Weights Computation

It is crucial that the demand weights w_d used in (2.8) are properly computed since the solution of Problem (2.8)–(2.10) will result in the optimal QoE-fair (rather than a throughput-fair) allocation of resources. From Section 2.2, we know that the larger the weight w_d the larger the assigned bandwidth slice X_d to the video flows belonging to demand d . It is then important to compute suitable weights in order to obtain a higher bandwidth for demands associated to users with high resolution screens and lower bandwidth for users with low resolution screens.

Notice that expression (2.8) is a weighted sum of logarithms, where the bandwidth slice X_d is the optimisation variable while the weights are constant. This is due to the fact that the weights only depend on the demand d , which is not variable in the

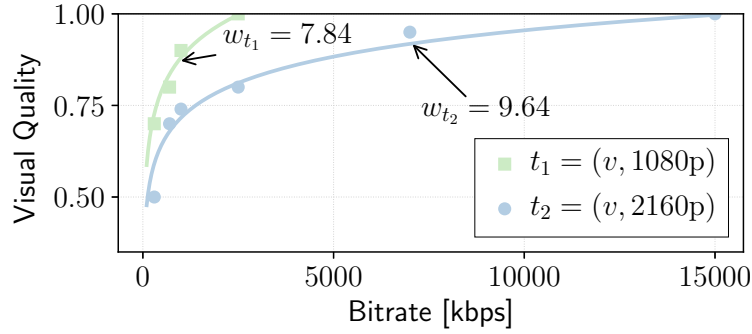


Figure 2.5: Computation of weights

optimisation problem. It is also important to stress that the weight w_d associated to a demand $d = (\text{src}, \text{dst}, t)$ does not depend on the source and destination node, but only on the particular video v and the user class c , i.e. on the video request t . As a consequence, given two demands d_1 and d_2 containing the same video request t , the weights associated to them will coincide, namely $w_{d_1} = w_{d_2} = w_t$.

Let us now define the couples (x_i, y_i) for $i = 1, \dots, L_t$ ($L_t = |\mathcal{L}_t|$) where $x_i = b_i \in \mathcal{B}_v$ and y_i is obtained through the mapping Q_t as described in Algorithm 1, i.e. $y_i = Q_t(l_i)$. The weight w_t can be computed as the parameter of a fitting function. In particular, we propose to consider a least square problem fitting the data (x_i, y_i) through the function $y = a \cdot \log x$, having a as the unique fitting parameter. Then, we impose $w_t = 1/a^\beta$, where β is a positive parameter to be tuned. It is easy to show that this proposed procedure assigns weights that increase as the user device resolution becomes higher (Fig. 2.5), which is exactly what is needed to provide higher network bandwidth shares to users with high resolution.

2.3.4 Video Clustering

The previous sections describe how to retrieve the necessary information to compute all the inputs to the optimisation Problem (2.8)–(2.10). As previously stated, the goal of the optimisation problem is to find the path flows x_{dp} such that the

QoE-aware objective function (2.8) is maximised, i.e. resources are distributed to provide a fair level of visual quality to heterogeneous users. In the multipath case, each demand d is split among a pre-computed number of available paths of the delivery network, i.e. $|\mathcal{P}_d|$. It follows that, in the multi-path case, the number of variables involved in the solution of the optimisation problem is equal to the number P of all possible paths available for each demand, i.e. if $\mathcal{P} = \{\mathcal{P}_1, \mathcal{P}_2, \dots, \mathcal{P}_D\}$, where D is the cardinality of the demand set \mathcal{D} , then $P = |\mathcal{P}|$. Since a demand is defined as the triple $(\text{src}, \text{dst}, t) \in \mathcal{N} \times \mathcal{N} \times \mathcal{T}$, it follows that $D = N \cdot (N - 1) \cdot T$. Now, recalling that a video request $t \in \mathcal{T}$ is defined as the couple $(v, c) \in \mathcal{V} \times \mathcal{C}$, it turns out that the cardinality of \mathcal{T} is equal to $V \cdot C$, i.e. the product of the video catalog size and the number of user classes. Notice that in practice, video catalogs of the order of millions or billions are possible for user providers distributing user-generated videos such as YouTube and Vimeo. Thus, considering a video provider serving a catalog size in the order of millions, it is easy to understand that the number of the video requests would make the cardinality D , and consequently P , too high and would result in an intractable optimisation problem.

This issue can be addressed by acting on the video catalog. The employed procedure is the following: for each user class $c \in \mathcal{C}$, we partition the video catalog \mathcal{V} in a number K of clusters $\{\mathcal{V}_1^c, \dots, \mathcal{V}_K^c\}$ according to a clustering algorithm, with $\mathcal{K} = \{1, \dots, K\}$ the set of the video cluster indexes. Since K is a design parameter, it can be chosen such that $K \ll V$. As a consequence, we can assign each video request $t = (v, c)$ to a *traffic class* $\tilde{t} = (k, c)$ where $k \in \mathcal{K}$ is the cluster the video v belongs to (i.e., $v \in \mathcal{V}_k^c$). In this way, the video requests $t = (v, c)$, whose video v is mapped to the same video cluster \mathcal{V}_k^c , belong to the same traffic class $\tilde{t} = (k, c)$.

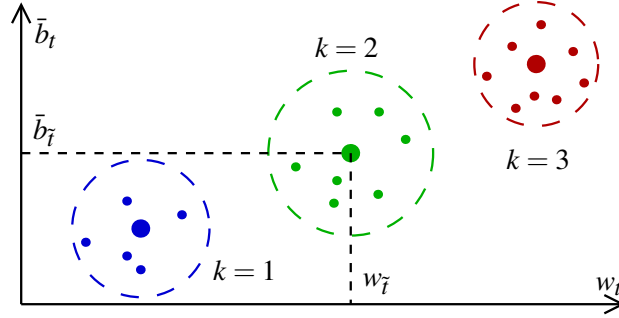


Figure 2.6: Proposed clustering procedure

Let us now redefine the demand as the aggregate of video sessions having the same triple $(\text{src}, \text{dst}, \tilde{t})$. Then, the cardinality of the new demand set will be equal to $N \cdot (N - 1) \cdot K \cdot C$, that can be made manageable by properly setting $K \ll V$.

Let us consider all the video requests t having a user class equal to $c \in \mathcal{C}$. Each video request is associated to a couple (w_t, \bar{b}_t) where w_t is the weight computed as discussed in Section 2.3.3 and \bar{b}_t is the associated reference video level bitrate. Fig. 2.6 shows an example of how the couples (w_t, \bar{b}_t) are distributed for a specific user class c . Notice that each point in the figure represents a single video. Next, we employ the k -medoid clustering algorithm to form K clusters as shown. As a result, each point in a cluster k represents a video belonging to the cluster \mathcal{V}_k^c . Moreover, for each cluster $k \in \mathcal{K}$, the algorithm computes the *medoid*, which is represented with a large dot in Fig. 2.6. Thus, the medoid of cluster k obtained for the user class c is representative of the traffic class $\tilde{t} = (k, c)$, i.e. of all videos in that cluster. Therefore, it is easy to associate to each \tilde{t} the weight $w_{\tilde{t}}$ and the bandwidth $\bar{b}_{\tilde{t}}$, which are the coordinates of the medoid. As an example, consider the cluster $k = 2$ in Fig. 2.6. The traffic class $\tilde{t} = (2, c)$ is associated with the weight $w_{\tilde{t}}$ and bandwidth $\bar{b}_{\tilde{t}}$ that are the coordinates of the medoid of cluster $k = 2$ (large green dot). It follows that changing the number of clusters K implies a trade-off between the number of variables involved in the optimisation problem and the ob-

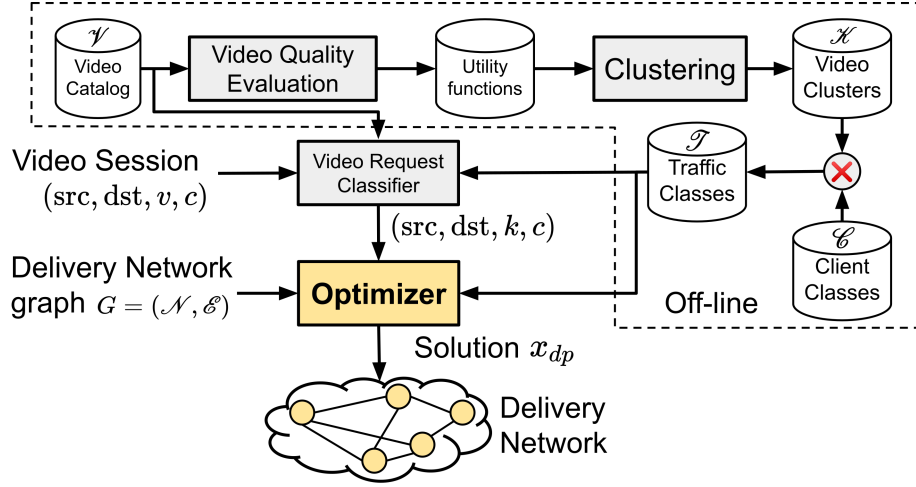


Figure 2.7: The proposed Video Control Plane

tainable QoE-fairness, as explained further on. It is important to point out that also other clustering algorithms have been considered (i.e. k-means and spectral clustering) but since they provide worse results, they have been omitted. In general, any clustering approach could be employed in the proposed methodology.

Fig. 2.7 provides an overview of the proposed resource allocation strategy and how it can be implemented in a Video Control Plane. In particular, after the visual quality evaluation of the video catalog, the fitting functions described in Section 2.3.3 are employed to perform the clustering procedure of the videos in order to obtain the traffic classes. Notice that these operations can be performed off-line since the video catalog and user classes are always available. Then, a video request classifier associates each received video session to the corresponding traffic class and then the optimiser, on the basis of the demands defined as (src, dst, k, c) , the traffic classes, and the delivery network graph, solves the MCFP we have discussed so far. The solution is implemented through programmable network elements such as SDN switches.

2.4 Results

In this section simulations and experimental results are shown. As a first step, a clustering of video requests is performed. Then, the *QoE-Proportional Fair* (PF) multi-path optimisation problem described in Section 2.2 is implemented. This allows us to carry out a performance evaluation of the proposed allocation strategy via simulations.

2.4.1 General Settings

The analysis is performed by varying three main parameters: the delivery network *load*, which represents the total traffic volume of concurrent video sessions, the number of paths P , i.e. the maximum number of paths that can be employed to realise a specific demand from a source node, and the number of clusters K .

In order to prove the effectiveness of the proposed allocation strategy, we consider as the *baseline* (BL) the QoE-unaware allocation strategy that associates each video session to the same traffic class. It is important to notice that the BL case is the approach currently used by video delivery services, which are unaware of the heterogeneity of user devices and video contents.

We have developed the proposed PF allocation strategy in a simulator composed of three modules implementing realistic scenarios of typical video distribution networks. The first module is the *video session generator* that, after receiving the network graph G , the video catalog \mathcal{V} , and the set of user classes \mathcal{C} as inputs, randomly generates a configurable number of video sessions (src, dst, t) . The second module is the solver, that employs the CVXPY Python tool [44] to implement Problem (2.8)-(2.10) by making use of the *Splitting Conic Solver* (SCS)¹ [120].

¹<https://github.com/cvxgrp/scs>

Finally, after the optimisation problem is solved, the third module, called *QoE evaluator*, computes the obtained QoE for each video session (src, dst, t) composing the load. The resulting QoE depends on the bandwidth share assigned by the solver and the corresponding visual quality given by the Q_t mapping. Finally, we use the definition of *fairness* F among video sessions proposed by [74]:

$$F = 1 - 2\sigma$$

where σ is the standard deviation of the QoEs obtained by concurrent video sessions. The maximum of the fairness index is 1, which is obtained only when concurrent video sessions are served exactly with the same visual quality.

YouTube videos have been chosen to build a realistic video catalog. In particular, we have built a video catalog of ~ 200 heterogeneous videos, all with a maximum resolution equal to 4K, fetched from YouTube on which we have performed the visual quality measurement reported in Algorithm 1. It is worth remarking that we do not re-encode the videos fetched from YouTube. Thus, the bitrate ladder of a given video is the one set by YouTube at time of encoding. In particular, for each video we select the six video representations encoded at resolutions 360p, 480p, 720p, 1080p, 1440p, 2160p and made available by YouTube (an example of bitrate ladder is given in Fig. 2.4). We have employed the VMAF metric to compute the video-level/video-quality mapping Q_t as described in Section 2.3.2. The VMAF metric has been implemented by using the open-source tools released by Netflix². We have assumed that clients can belong to three possible user classes—which are representative of most common user devices—belonging to the set $\mathcal{C} = \{720p, 1080p, 2160p\}$. The *load* values considered to generate the video sessions range in the set $\{100, 200, 300, 400, 500\}$ Gbps while two networks have

²<https://github.com/Netflix/vmaf>

been chosen as delivery network topologies in order to make a comparison as well as a performance evaluation. The first network topology is the GARR network³, composed of 61 switches and 73 links with an average capacity of ~ 4 Gbps. We fixed 10 switches as server nodes and the remaining 51 as clients. The second network topology, the Abilene network, is smaller sized, being it composed of 11 switches and 14 links. In this case we set 4 switches as server nodes and 7 as switches used by the clients. Finally, the set of clusters is such that $K \in \{3, 5\}$, the set of paths is such that $P \in \{1, 2, 5\}$ and the weight parameter β can be chosen among the values in the set $\{1.1, 1.2, 1.3, 1.4, 1.5\}$.

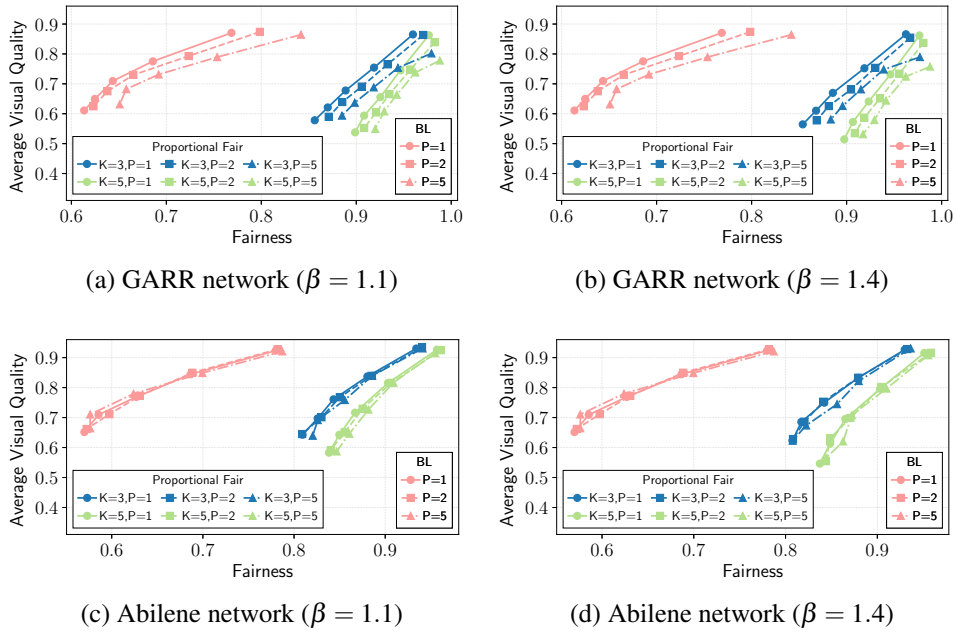


Figure 2.8: QoE-Fairness vs Average Visual Quality

2.4.2 QoE Fairness vs Average Visual Quality

At this point, we can compare the obtained results on both the considered networks. Fig. 2.8 shows the trade-off between the average QoE obtained by video

³<http://www.topology-zoo.org/files/Garr201201.gml>

sessions and the corresponding QoE-fairness when BL is employed or in the case of the proposed PF resource allocation strategy. It is worth stressing that the considered fairness is obtained by computing the fairness metric F for each slice and then taking the average value of the fairness associated to all the slices. Each line represents a particular scenario where a different line style denotes a specific number of paths P involved in the allocation and each marker on a line is representative of a specific load in $\{100, 200, 300, 400, 500\}$ Gbps. In the PF case, different colors indicate a different number of clusters K . Moreover, Fig. 2.8 shows only the cases of $\beta = 1.1$ and $\beta = 1.4$. As it is clear from any of the figures, the average visual quality and the QoE fairness decrease as the load on the delivery network increases. This is expected since a higher load results in a lower allocated average bandwidth share per video session and consequently to a lower visual quality. Consider Fig. 2.8a as an example: it shows that in the BL case, independently of the number of paths, the average visual quality is close to 0.9 when the load is 100 Gbps, then it decreases to 0.8 for a 200 Gbps load and so on. The fairness presents values in the range 0.62-0.84 with a corresponding average visual quality in the range 0.6-0.88. However, the proposed PF approach proves remarkably better in terms of achieved QoE fairness for each considered number of clusters K and paths P . The visual quality is basically unchanged compared to the BL case. Although this result may appear unexpected, we need to keep in mind that we are referring to the *average* visual quality and that also in the BL case the QoE is maximised, but without considering the fairness. This leads to a situation in which, in the BL case, some users experience a high level of QoE while others have a low level of it. The average of all these values results similar to the average of the values obtained in the PF case, in which fairness is considered and therefore the QoE of users is more uniform. In this way, even though the average QoE levels may be similar, the fairness is not. Furthermore, as expected, the QoE fairness improves

as K increases and, consequently, each line associated to a particular number of clusters and paths moves to the right and becomes steeper. Such considerations also hold for all the other cases in the figures, where $K = 5$ clusters appears to be the best trade-off between average visual quality and QoE fairness.

Next, consider Fig. 2.8a and 2.8b related to the GARR network. By varying β from 1.1 to 1.4, the average visual quality slightly drops while the average fairness remains almost unchanged. This is due to the fact that the fairness among user classes is increasing—as will be better shown in the following—and this implies a graceful degradation of the visual quality of all users in the network. Similar considerations could be made in the case of Abilene network in Fig. 2.8c and 2.8d. Indeed, the multi-path resource allocation is preferable with respect to the single-path case due to the possibility of exploiting more paths to realise a demand. Therefore, when passing from Abilene to a wider and more complex network such as GARR, the multi-path approach outperforms the single-path case.

Let us now analyse in more detail the effect that the choice of the parameter β has on the QoE fairness in the single-path case (the multi-path approach gives similar results). Fig. 2.9 reports the CDF of the visual quality obtained by all video sessions grouped by user class, i.e. the maximum screen resolution of users, in the case of a 500 Gbps load (results for different loads are similar). Notice that the resolutions reported in the figures are used only to identify the user classes. In other words, they represent the maximum screen resolution of users in a class, but the actual resolution of each user during playback can be lower. The figure shows that, regardless of the chosen network, when passing from $\beta = 1.1$ to $\beta = 1.4$ (Fig. 2.9b and 2.9c or Fig. 2.9e and 2.9f) the MCFP gives a better performance in terms of fairness. Moreover, the obtained fairness in the PF case with $\beta = 1.4$ is remarkably better than the BL case. The median value of the visual quality

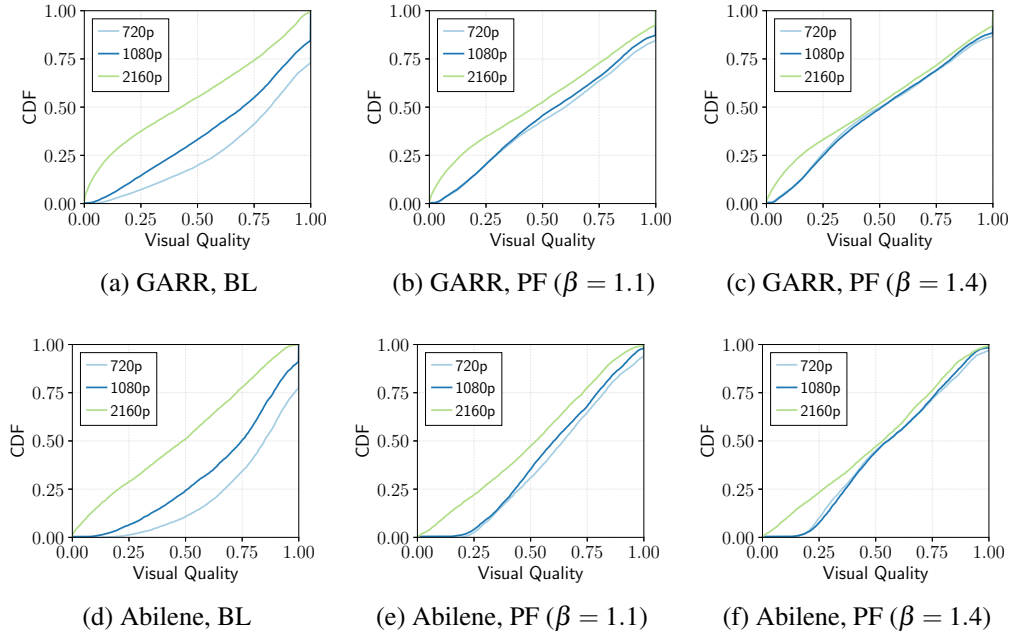


Figure 2.9: CDF of visual quality for different user classes and distribution networks in the case of a 500Gbps load

for clients with 720p, 1080p and 2160p resolution in Fig. 2.9a is respectively 0.45, 0.7, 0.8, while for the PF case in Fig. 2.9c the values are 0.48, 0.5, 0.5. As a result, the PF case with $\beta = 1.4$ guarantees a high level of fairness since all users belonging to any user class will enjoy a similar visual quality. However, an increase in fairness will unavoidably imply an overall decrease in the average visual quality as shown in Fig. 2.8. The same considerations could be made for the Abilene network. Moreover, simulations for $\beta = 1.5$ are not shown because in the case of the GARR network the results in terms of visual quality fairness deteriorate compared to the case in which $\beta = 1.4$. In particular, for $\beta = 1.5$, clients with 720p and 1080p resolution obtain a worse visual quality than 2160p clients. However, in the Abilene network, the fairness improves, i.e. the three curves are more closely aligned, thus making $\beta = 1.5$ preferable to the previous values. For this reason, β is a parameter that has to be properly tuned according

to the considered network topology.

Throughout this work we have supposed that good ABR algorithms run at the players. It is important to clarify that if this is not the case, i.e., ABR algorithms do not work properly and make bad decisions about chunk levels, then, although fairness may be preserved, the average visual quality can drop dramatically.

2.4.3 Computation Time

In this subsection, we provide some insights into the computation time required to solve the proposed optimisation problem. Notice that, since ABR algorithms are independent and free to run at the clients, the optimiser can be run every Δt seconds, where usually Δt is significantly larger than a single video segment (which is in general in the order of 1-10s) and must be greater than the time spent to solve the optimisation problem. In fact, the optimisation loop can be seen as the *outer loop* that sets the bandwidth slice for video flows, while the *inner loop* is represented by the ABR algorithm that selects the video representation at a chunk time scale [34]. Regarding this architecture, named *cascaded control system*, a well-known practice used in controlling such systems entails separating the time scales at which each control loop works. The idea is to have the outer loop work at a higher sampling time and the inner loop to run faster at a lower sampling time. This is motivated by the fact that, with such a control architecture, having two controllers working at around the same sampling time might provoke adverse effects on stability. Thus, we argue that a sampling time that is greater than 4-5 times the duration of a segment (~ 25 seconds) is *required* to enforce the aforementioned separation of time scales.

Fig. 2.10 shows the computation time needed to solve the MCFP in the case of Abilene and GARR networks. As expected, time complexity increases when

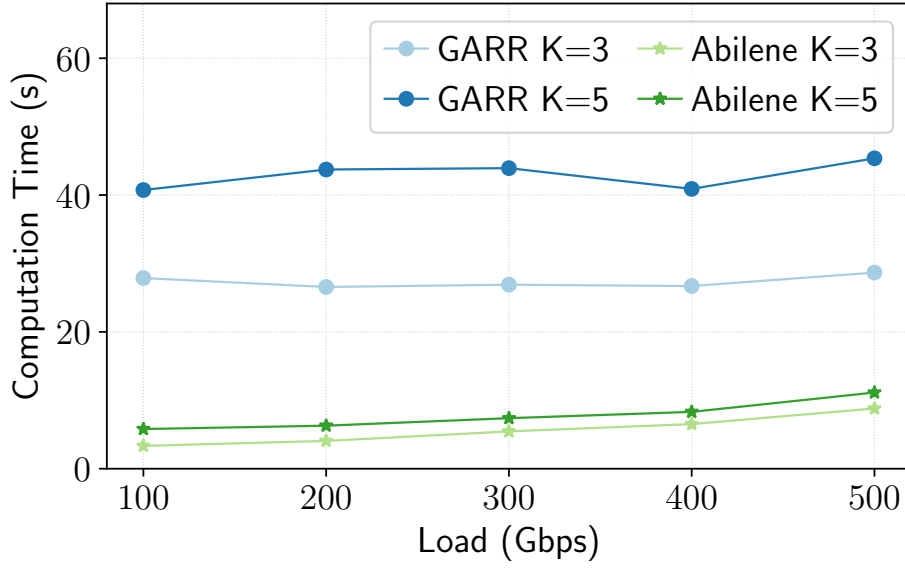


Figure 2.10: Computation time required to solve the MCFP

passing from 3 to 5 clusters and from 100 to 500 Gbps for both the considered networks. Moreover, the time required to solve the optimisation problem is higher ($\sim 4x$) in the case of the GARR network compared to Abilene due to the larger number of nodes of the GARR network. It is worth noting that the computation times are obtained using only one CPU core of an i7 workstation using an open-source tool, thus one can expect a significant speed-up using a commercial optimiser that supports multi-threading. Nevertheless, it is important to observe that, even in this non-ideal operating scenario, the obtained worst case for a large network is $\sim 50s$ which is in the order of ~ 10 video chunks.

2.5 Concluding Remarks

In this Chapter, a Proportional Fair (PF) resource allocation strategy has been proposed to achieve QoE-fairness among concurrent heterogeneous users in video delivery networks. To this purpose, a Multi-Commodity Flow Problem has been

properly formulated and employed to satisfy the maximum number of video requests coming from clients. Then, a clustering procedure has been adopted to partition the video catalog with the purpose of making the number of variables involved in the optimisation problem manageable. The performance of the proposed PF allocation strategy has been compared to the case of a QoE-unaware allocation strategy, which represents the currently deployed video delivery networks. Experimental results on two realistic networks show that the proposed PF allocation strategy is able to remarkably improve fairness among heterogeneous clients. It is important to point out that, since the clustering process depends on the video catalog which may change many times, a continuous update of the clusters should be guaranteed.

Chapter 3

Synchronising Live Video Streaming Players Via Consensus

In this Chapter, we introduce the problem of synchronising the playout times of users attending a live streaming event. To tackle this issue, the problem is set as a multi-agent system where a consensus control protocol is applied. As a further enhancement, in order to avoid continuous communication among users, a distributed event-triggered control is proposed. Results shown in the following are described in [101] and [104]. The latter has been submitted to IEEE Transactions on Networking.

3.1 Introduction and Background

Classical TV broadcast channels are gradually being replaced by video streaming services since they offer a wide range of contents both on-demand (e.g. movies, TV series) and live (e.g. sports, news). One of the advantages of online services w.r.t. traditional TV channels is that providers have the possibility to gather a considerable amount of real-time feedback information on viewers' behaviour and preferences. In this way, providers are able to efficiently personalise advertisements and recommendations.

The drawback of video streaming services consists of two main issues that negatively affect the user's Quality of Experience (QoE). First, video playback might stall or video quality might degrade impacting *service smoothness* and second, in live events, geographically distributed users might watch the same content but with different playback times having detrimental effects on the *service togetherness*, which is defined as the level of satisfaction users perceive when feeling that the service is experienced together with a number of users (for instance friends). The goal of video streaming providers is to design their services in such a way that the user's perceived quality is maximised and service abandonment is reduced, which avoids detrimental impacts on revenues. Service smoothness and QoE are an issue on the Internet because network resources, e.g. bandwidth and buffers, are shared and can become congested. On the contrary, traditional broadcast services deliver content via *dedicated medium*, i.e. not shared, and the quality of the service is guaranteed.

As we have seen, video streaming content is delivered to clients through an Internet path whose network bandwidth is unpredictable and time-varying. When the video encoding bitrate is larger than the available network bandwidth, the video

playback will eventually stall due to playout buffer depletion [33]. To this end, the standard employed for media streaming content is the MPEG-Dynamic Adaptive Streaming over HTTP (MPEG-DASH or DASH) [146], where the video bitrate is adapted to the available network bandwidth. DASH requires that each video is divided into *segments* or *chunks* of fixed duration τ (typically from 1 to 10 seconds). Then, each segment is compressed to create a number of *representations*, or *levels*, to which different encoding bitrates and video resolutions are associated. The DASH standard allows clients' players to dynamically choose the video level that matches the available network bandwidth using an Adaptive BitRate (ABR) algorithm. The goal of the ABR algorithm is to maximise the overall quality perceived by the client given the Internet available bandwidth. As a consequence, such control algorithms are designed to avoid rebuffering events, i.e. when the playout buffer gets completely depleted and video reproduction is interrupted [142].

Although there exists a wide literature on ABR control algorithms [38], [169], [33], when users in different locations watch together a live video content, such as in the case of a football match, the issue of synchronising video streams to offer service *togetherness* is still unsolved.

Consider a group of people geographically distributed in different locations and concurrently consuming a live event (e.g. a football match). If streams are not synchronised among users, some of them may watch a particular event (e.g. a goal in a football match) before the others, thus negatively impacting users' feelings of togetherness. As it will be explained more in detail later, synchronisation issues arise mainly due to the different time instants users join a live streaming event.

When a user starts watching a live streaming event, the ABR algorithm starts fetching the chunks that have been produced most recently by the content provider. Since ABR algorithms aim at mitigating stalls, a suitable number of segments

must be fetched and placed in the playout buffer before the playback can be started. Thus, the time instant the video starts being reproduced depends on the ABR algorithm controller's parameters and on the network bandwidth that is available at the client.

Few works tackling the aforementioned problem of synchronisation are present in the literature, all of them advocating web-based solutions using Real-Time Transport Protocol (RTP) or HTTP Adaptive Streaming (HAS). For instance, some works adopt different approaches for MPEG-DASH in which all nodes have to synchronise through the exchange of specific Media Presentation Description (MPD) files. Besides, most of such works employ centralised approaches [124], [106], whose drawback is that they do not scale with the number of users since they require a server to handle synchronisation messages from/to possibly millions of nodes concurrently. In this context, several standards have been conceived for web-based media synchronisation [152]. An interesting centralised approach to achieve video playback synchronisation among users could leverage *adaptive piggybacking*, which was initially proposed in [67]. The authors suggest modifying the video playback rate of users watching the same content in order to make them *meet* and merge their streams into a single stream provided by the server, which can now serve the entire group of merged requests. Though the purpose of their work was different, we believe that their technique could lead to satisfactory results when the number of users is low, thus avoiding scalability issues. On the other hand, works proposing a decentralised approach make use of heuristics that do not allow a rigorous and systematic analysis of system properties [130], [113]. Notice that also these works adopt solutions based on web protocols or make use of modified versions of MPDs, thus resulting in approaches that are not easily implementable.

In this Chapter, we tackle the problem from a rigorous control-theoretic perspective, whose main advantage is to be context-independent, i.e. control laws can be implemented independently of the protocols and frameworks employed. In particular, a fully distributed approach for video playout synchronisation is proposed. With this approach, each client in the network can communicate/receive the playout time only to/from a limited set of neighbours and clients do not require centralised information from servers, i.e. the absolute playback time.

In the following sections, a mathematical model of the playback time of an event that clearly explains the causes for synchronisation issues is provided. Then, the problem of synchronising video players is stated as a consensus problem involving integrators with saturated control inputs. At this point, the absolute delay of users with respect to the video content provider is reduced by adopting a leader-follower approach. Finally, an event-triggered control avoids users continuously sending information to their neighbours so that communication occurs only at specific time instants, which reflects realistic cases and still guarantees leader-follower consensus achievement. Simulations on some sample network topologies show that consensus is achieved and therefore all clients reach synchronisation.

The application of consensus theory to the field of video streaming is not new in the literature, e.g. in [36] consensus is leveraged to provide video quality fairness. In particular, a discrete-time linear distributed consensus problem is considered to provide quality fairness among users sharing limited network resources. Therefore, such a work is focused on guaranteeing visual quality fairness to users consuming heterogeneous multimedia content over the Internet rather than achieving playout time synchronisation in a group of users enjoying the same live streaming event, which is our case. Notice that in the solution proposed in our work, an important design choice is that the synchronisation algorithm is decoupled from the

ABR algorithm. With such an approach the deployment of our solution is made very easy and does not require any changes in the ABR algorithm.

The playback rate will be the control input used to synchronise clients, which is an approach denoted as Adaptive Media Playout (AMP) in the literature [80], [122]. In practice, the playback rate, i.e. the speed of reproduction of the video, *can be slightly adjusted* to control the playback time. This technique has been used also to enhance the client's buffer memory requirements and to reduce playout interruptions [147], [114]. It is also important to say that issues related to audio streaming with AMP have been widely addressed [134], [115], [135], and therefore are not accounted for in our work. Furthermore, several studies on the QoE have shown that the playback rate can be increased/decreased by only a small amount to prevent the QoE from being negatively impacted [62], [129]. Thus, to limit QoE degradation, the proposed approach considers the playback rate (i.e. the control input) to be bounded in a given interval. Under these conditions, we show that the proposed approach makes all clients converge asymptotically to the same playback time (Section 4.4) also in the case of event-triggered control (Section 4.5).

3.2 Playback Time Model

In Section 3.2.1 we provide a model of the playback time. Based on this model, in Section 3.2.2 we show how the de-synchronisation arises among video players. Finally, in Section 3.2.3 we present a model of the Adaptive Media Playout approach to be used to synchronise players.

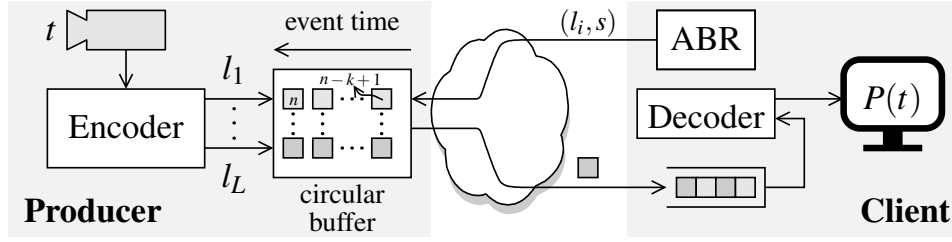


Figure 3.1: Live video streaming system

3.2.1 Video playback time model

As a first step, let us provide a model of the playback time of a user watching a live event that starts at time $t = 0$. If we suppose that the time of the live event is the same as the real time t , then the term *time* and *event time* can be used interchangeably in the rest of the Chapter.

Figure 3.1 shows how a DASH-compliant live streaming system, such as YouTube or Facebook, produces live content. A camera captures a live scene that is compressed in real-time by an encoder. The latter produces video segments each τ seconds by encoding the same video content at different bitrate levels (or resolutions) l_i belonging to a discrete set $\mathcal{L} = \{l_1, \dots, l_M\}$ ($l_i < l_{i+1}$). For each level $l_i \in \mathcal{L}$, the *circular buffer* depicted in the figure receives and stores the last k segments produced by the encoder. A circular buffer is a fixed-size buffer where the starting location chosen to store the first segment is not important and the extraction of data follows a *first in, first out* (FIFO) logic. Therefore, clients will start downloading the oldest segments in the buffer. In the same way, when the circular buffer is full and a new chunk has to be stored, the oldest segment is overwritten. In other words, this buffer contains only the most recently produced k segments, which are made available for download to the clients that want to join the live streaming event. Each segment of duration τ is present in the circular buffer with M different levels of resolution. The circular buffer contains a number of seg-

ments representing the scene in a time window $W(t)$, as defined in the following.

Let $s(t)$ be the segment index that contains the video scene at time t :

$$s(t) = \left\lfloor \frac{t}{\tau} \right\rfloor \in \mathbb{N} \quad (3.1)$$

where $\lfloor \cdot \rfloor$ is the *floor* operator that maps a real number x to the greatest natural number less than x . According to its definition, the segment $s(t)$ contains the event scene in the time interval $[s(t)\tau, (s(t) + 1)\tau[$. Hence, at time t , the circular buffer of k segments contains video scenes in the time window:

$$W(t) = [n\tau - k\tau, n\tau[\quad (3.2)$$

where $n = s(t)$ is the n -th chunk.

Let t_J be the time at which a user joins the live event. In t_J the user immediately starts downloading the oldest segment available in the circular buffer, whose index is $s(t_J) - k$, since the buffer contains k segments. Once the segments are downloaded, they are temporarily stored in the client's playout buffer whose level, measured in seconds, is denoted with $q(t)$. As already explained, for each segment to be delivered, the ABR control algorithm selects the video level $l(t) \in \mathcal{L}$ that matches the available network bandwidth. In general, before playing the video, an ABR control algorithm fills the playout buffer until the level $q(t)$ reaches a minimum value, say q_L , that is considered enough to mitigate the rebuffering events occurring when the buffer gets completely depleted and video reproduction is interrupted ($q(t) = 0$). The playout buffer level $q(t)$ can be modeled as an integrator [33]:

$$\dot{q}(t) = f(t) - p(t) \quad (3.3)$$

where $f(t)$ is the video fill rate in seconds of video for a unit of second, and $p(t)$ is the *playback rate*, i.e., the rate of seconds of video drained by the playout buffer and fed to the decoder. In other words, the playback rate is the speed at which the video is played on the user's screen. Therefore, the following holds:

$$f(t) = \frac{\Delta T_{video}}{\Delta T} = \frac{\Delta data}{\Delta T} \cdot \frac{\Delta T_{video}}{\Delta data} = \frac{r(t)}{l(t)} \quad (3.4)$$

where $r(t) = \Delta data / \Delta T$ is the download rate, which depends on the time-varying network bandwidth measured in bytes/s, and $l(t) = \Delta data / \Delta T_{video}$ is the encoding quality level, [33].

When the video is playing, the playback rate $p(t)$ is equal to 1, whereas it is 0 when the video is paused. Thus, the following holds:

$$p(t) = \begin{cases} 1 & \text{playing} \\ 0 & \text{paused} \end{cases} \quad (3.5)$$

Let us now model the playback time $T(t)$, i.e. the time instant of the video that the user is watching at time t . In general, the playback time $T(t)$ is different from the time t of the live event because of the unavoidable delays due to encoding, decoding, and propagation. Once the user joins the live event at time $t = t_J$, the playback time of the first segment downloaded is:

$$T_0 = s(t_J)\tau - k\tau \quad (3.6)$$

where T_0 corresponds to the initial time of the first segment stored in the producer's circular buffer at time t_J . Notice that the video is not played at the client ($p(t) = 0$) until the queue level $q(t)$ is higher than q_L . Let t_B be the time needed to fill the playout buffer to reach q_L . Then, it can be found by integrating (3.3) and imposing the constraint $q(t_B) = q_L$. Therefore, t_B depends on both the download rate $r(t)$, which can be considered as a disturbance, and $l(t)$, which is chosen by the ABR control algorithm. As a consequence, the video starts being reproduced at the user's device at time $t_P = t_J + t_B$. Thus, when $t \geq t_P$, the playback time dynamics is given by:

$$\dot{T}(t) = p(t) \quad (3.7)$$

whose initial condition is $T(t_P) = T_0$. Keeping in mind that the video starts at t_P and supposing no rebuffering events occur after t_P , i.e. $p(t) = 0$ for $t < t_P$ and $p(t) = 1$ for $t \geq t_P$, we can integrate (3.7) between t_P and a generic time instant t taking into account the initial condition:

$$\begin{aligned} T(t) &= \int_{-\infty}^t p(\xi) d\xi = \int_{-\infty}^{t_P} p(\xi) d\xi + \int_{t_P}^t p(\xi) d\xi = \\ &= T_0 + \int_{t_P}^t p(\xi) d\xi = T_0 + t - t_P \end{aligned} \quad (3.8)$$

3.2.2 The synchronisation issue

In Figure 3.2 the issue of synchronisation is explained. The figure shows the playback time of two users, say i and j , interested in watching the same live streaming event. Let us consider a live video streaming system, with a circular

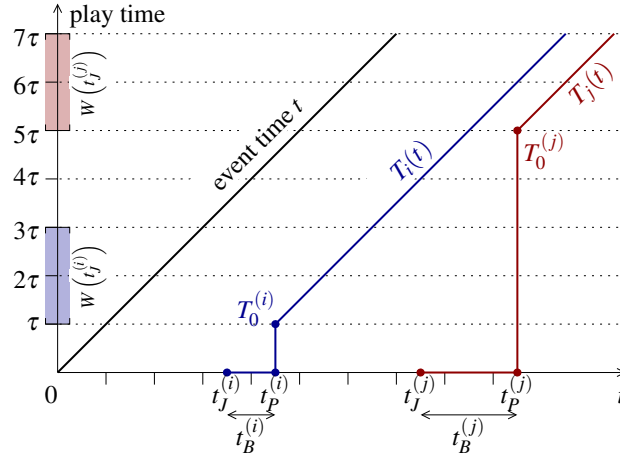


Figure 3.2: Playback time dynamics of two users i and j joining at different time instants $t_J^{(i)}$ and $t_J^{(j)}$

buffer containing the two most recently produced segments, i.e. $k = 2$. From the figure, it is possible to see that user i is the first to join the event at time $t_J^{(i)}$, after which the ABR algorithm immediately starts to download and store in the playout buffer the video chunks contained in the circular buffer, which are those contained in the time window $W(t_J^{(i)})$ (see the blue shaded area in the figure). The buffer time length $q(t)$ reaches q_L after $t_B^{(i)}$ seconds so that at $t_P^{(i)} = t_J^{(i)} + t_B^{(i)}$ the playback on the user's device starts, i.e. $p(t) = 1$ for $t \geq t_P^{(i)}$. Moreover, the playback starts at a play time $T_0^{(i)}$, as defined in (3.6), which is the play time of the video at the beginning of the first chunk stored in the buffer, i.e. at the beginning of the window $W(t_J^{(i)})$. At time $t_J^{(j)} > t_J^{(i)}$, user j also joins the live event. Analogously, the ABR algorithm retrieves the video chunks identified by the time window $W(t_J^{(j)})$ (see the red shaded area in the figure) and stores them in the playout buffer until it reaches the minimum level q_L needed to start playing the video on the user's device. It is important to point out that, in general, the value of q_L might be different for the two users. Additionally, the time $t_B^{(j)}$ needed to reach q_L in the playout buffer of user j is different in general from $t_B^{(i)}$. Therefore,

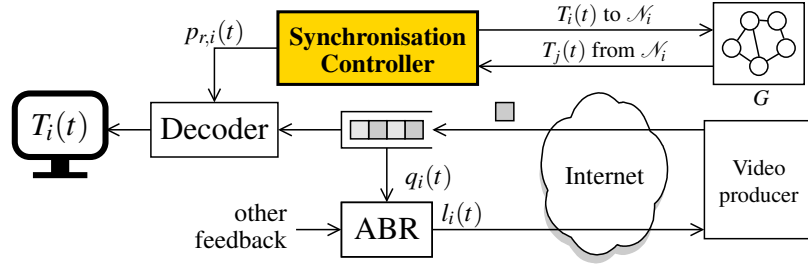


Figure 3.3: The proposed synchronisation control architecture

user j will start watching the event at time $t_p^{(j)} = t_j^{(j)} + t_B^{(j)}$ from a play time $T_0^{(j)}$. As found in (3.8), the playback time of user i is $T_i(t) = T_0^{(i)} + t - t_p^{(i)}$ while the playback time of user j is $T_j(t) = T_0^{(j)} + t - t_p^{(j)}$. Hence, the lack of synchrony between the two users is equal to:

$$T_i(t) - T_j(t) = [(T_0^{(i)} - t_p^{(i)}) - (T_0^{(j)} - t_p^{(j)})] + (t_B^{(j)} - t_B^{(i)}) \quad (3.9)$$

where $T_0^{(i)} - t_p^{(i)}$ is the distance between $t_j^{(i)}$ and the play time $T_0^{(i)}$ of the video at the beginning of the first chunk stored in the buffer. The same can be said for $T_0^{(j)} - t_p^{(j)}$ in the case of user j . The two users are synchronised when $T_i(t) - T_j(t) = 0$.

Based on the above, we can conclude that the lack of synchrony between two users depends on their join and buffering times.

3.2.3 The Adaptive Media Playback model

In the previous section, we have proved that if the video playback rate of players is constant, then playback times are in general not synchronised in accordance to (3.9). In this section, the AMP approach is presented since it can be leveraged to synchronise users. The AMP approach consists of slightly varying the playback rate around the nominal value 1, i.e. the playback is slightly slowed down or sped

up of a time-dependent amount, say $u(t)$. Clearly, $u(t)$ must be small enough to be barely noticeable by the user so that the user's QoE is not affected [62], [129].

At this point, the playback rate must be redefined as follows:

$$p_r(t) = p(t)(1 + u(t)) \quad (3.10)$$

where $u(t) \in [-\delta, \delta]$, with δ small enough, and $p(t)$ is the playback rate (3.5). Thus, the playback time now becomes:

$$T(t) = \int_{t_P}^t p(\xi)(1 + u(\xi))d\xi, \quad \forall t \geq t_P \quad (3.11)$$

with initial condition $T(t_P) = T_0$. Suppose N clients are watching the same live streaming event and since, as already mentioned, ABR algorithms are specifically designed to avoid rebuffering events, assume none of the clients is experiencing a rebuffering event (i.e. $p(t) = 1 \quad \forall t \geq t_P$). Then, for each client i and $\forall t \geq t_P^{(i)}$, the playing time is:

$$T_i(t) = \int_{t_P^{(i)}}^t (1 + u_i(\xi))d\xi \quad (3.12)$$

whose initial condition is $T_i(t_P^{(i)}) = T_0^{(i)}$. The playback rate variations $u_i(t)$, $i = 1, \dots, N$, represent the control variables that will enforce synchronisation among users. Synchronisation can be formally stated as: $e_{ij}(t) = T_i(t) - T_j(t) \rightarrow 0 \quad \forall i, j = 1, \dots, N, i \neq j$, exponentially.

3.2.4 The Proposed Synchronisation Approach

At this point, having introduced the model of the playback time for each client consuming the same live streaming content, we are ready to present the proposed distributed approach to achieve synchronisation through the formulation of a consensus problem involving integrators and saturated inputs. To this end, the following design requirements are needed: (R1) the system must be horizontally scalable, i.e. it should work also in the case of large events (N large); (R2) it has to be implementable using technologies already available in the media distribution industry.

To meet the design criterion (R1), a decentralised control approach must be used, which involves messages being sent only among selected users. To exchange synchronisation messages directly among users, without the need for a central server, the WebRTC open standard can be used, which is a widely available technology that allows real-time communication among browsers and is supported by all major Internet browsers. To meet the requirement (R2), the video is delivered using the standard DASH protocol specification without requiring any change to the ABR algorithm running at the client.

The architecture associated with the proposed approach for playback time synchronisation is shown in figure 3.3. The generic i -th client runs an ABR algorithm that, on the basis of some information such as the estimated available bandwidth and the playout buffer level $q_i(t)$, dynamically selects the chunk level $l_i(t)$ and requests it to the video producer server [33], [38]. The server sends the video segments encoded at the level required by the ABR algorithm. Once received by the client, these segments are stored in the playout buffer waiting to be played. The *synchronisation controller*, which sends to and receives information from neighbouring devices, is the core of the proposed algorithm: it selects the most suitable

value of the playback rate $p_{r,i}(t)$ at which the video has to be played at client i . As previously said, the value of $p_{r,i}(t) = 1 + u_i(t)$ should remain as close as possible to 1 to mitigate QoE degradation. As a consequence, $u_i(t)$ has to be bounded in a set $[-\delta, \delta]$ and has to converge to zero when synchronisation is achieved. As a last step, the decoder decompresses the video frames drained from the playout buffer and renders them on the client's screen at the playback rate $p_{r,i}(t)$ imposed by the synchronisation controller.

Before describing the proposed approach for playback time synchronisation of users, let us introduce the following non-restrictive assumption:

Assumption 1. Once each user $i = 1, \dots, N$ starts the playback of the live streaming content, no rebuffering event occurs, which implies (i) $p_i(t) = 1 \forall t \geq t_p^{(i)}$ and (ii) the ABR algorithm is able to avoid playout buffer depletion, i.e. $q_i(t) > 0, \forall t \geq t_p^{(i)}$.

The proposed approach is based on the dynamic model of the playing time evolution, which is derived from (3.12). Firstly, the initial delay of the i -th user is defined as $T_{0i} = T_0^{(i)} - t_p^{(i)} < 0$. The synchronisation algorithm starts at time $t = t_s$ when all N users attending the event have started the video playback, i.e. $p_i(t) = 1 \forall t > t_s, \forall i \in \{1, \dots, N\}$. This means that for $t < t_s$ the synchronisation algorithm is not active, therefore it must result that $u_i(t) = 0, \forall i \in \{1, \dots, N\}$. Thus, the playback time can be obtained as follows:

$$T_i(t) = T_{0i} + t + \int_{t_p^{(i)}}^t u_i(\xi) d\xi = T_{0i} + t + \int_{t_s}^t u_i(\xi) d\xi \quad (3.13)$$

$\forall t \geq t_s$, where the last equality comes from the fact that $u_i(t) = 0$ for $t_p^{(i)} \leq t \leq t_s$. Hence, the dynamical model of $T_i(t)$ is:

$$\dot{T}_i(t) = 1 + u_i(t) \quad (3.14)$$

In order to simplify the modelling, a change of coordinates is performed to obtain a set of integrators: let $x_i(t) = T_i(t) - t$, then $\dot{x}_i(t) = \dot{T}_i(t) - 1 = u_i(t)$, $\forall t \geq t_s$. As a consequence, controlling the models of all clients is equivalent to controlling a set of integrators as defined in the following:

$$\begin{cases} \dot{T}_1(t) = 1 + u_1(t) \\ \vdots \\ \dot{T}_N(t) = 1 + u_N(t) \end{cases} \equiv \begin{cases} \dot{x}_1(t) = u_1(t) \\ \vdots \\ \dot{x}_N(t) = u_N(t) \end{cases} \quad (3.15)$$

Notice that the physical meaning of the variable $x_i(t)$ is the temporal delay of user i with respect to the current playout time transmitted by the video provider. It is at this point that the stated problem can be modelled as a consensus problem. To this end, let us define a directed graph (or digraph) $G(\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, \dots, v_N\}$ is the set of nodes, also identified simply by its indices $i = 1, \dots, N$, and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges. An edge (v_i, v_j) denotes the information flow from node i to node j . Moreover, the set of neighbours of node v_i is defined as $\mathcal{N}_i = \{v_j \in \mathcal{V} : (v_j, v_i) \in \mathcal{E}\}$.

It is now straightforward to model the clients as the nodes $i \in \mathcal{V}$ of the graph G , where each node i is associated with the corresponding state $x_i(t)$ and receives information from a certain number of neighbours \mathcal{N}_i .

If control inputs $u_i(t)$ are supposed to be unbounded, then, given a set of integrators, we can employ the following well-known control strategy [119]:

$$u_i(t) = \sum_{j \in \mathcal{N}_i} a_{ij}(x_j - x_i) = \sum_{j \in \mathcal{N}_i} a_{ij}(T_j - T_i) \quad (3.16)$$

where $A = A(G) = (a_{ij})$ is the adjacency matrix with $a_{ij} = 1$ if $(j, i) \in \mathcal{E}$ and $a_{ij} = 0$ otherwise.

The resulting system is:

$$\dot{x}(t) = -Lx(t) \quad \forall t \geq t_s \quad (3.17)$$

where $x(t) = [x_1(t), \dots, x_N(t)]^\top$ is the stack vector of all agents' states and L is the Laplacian matrix associated with the adjacency matrix A . Let $D = D(G) = (d_{ij})$ be the in-degree matrix with $d_{ij} = \text{in-deg}(v_i)$ if $i = j$ and $d_{ij} = 0$ otherwise, where $\text{in-deg}(v_i)$ is the in-degree of node v_i , i.e. the number of edges pointing towards node v_i . Then, the Laplacian matrix can be defined as $L = D - A$. If G is strongly connected, $-L$ has eigenvalues such that $-\lambda_{N-1} \leq -\lambda_{N-2} \leq \dots < -\lambda_0 \leq 0$, where $\lambda_0 = 0$ and $\text{rank}(L) = N - 1$.

It is well-known that the control protocol (3.16) solves the consensus problem for the set of integrators (3.15) globally and asymptotically [119]. Therefore, it can be stated that (3.17) is stable and there exists an $\alpha \in \mathbb{R}$ s.t. the system converges to the equilibrium point $\bar{x} = \alpha \mathbf{1}$, i.e., $\bar{x}_i = \alpha, \forall i \in \mathcal{V}$. Moreover, if G is balanced, i.e. the in-degree is equal to the out-degree for each node, then $\alpha = \mathbb{E}[x_i(t_s)] = \frac{1}{N} \sum_{i=1}^N x_i(t_s)$.

Once consensus is achieved, $x_i(t) = T_i(t) - t = T_{0i} + \int_{t_s}^t u_i(\xi) d\xi = \alpha$ and thus $T_i(t) = t + \alpha$ for all i . In other words, if the consensus problem is solved, then also the playback time synchronisation problem is solved. Notice that when G is balanced, $\alpha = \mathbb{E}[x_i(t_s)]$, so it turns out that $T_i(t)$ converges to $t + \mathbb{E}[T_{0i}]$, where the

second term represents the average of the initial delays of the clients.

The previous approach considers unbounded control inputs $u_i(t)$, which is not a realistic case. As previously stated in Section 3.2.3, all control inputs should be bounded, i.e. $|u_i(t)| \leq \delta \forall i \in \mathcal{V}$, to avoid a perceptible speedup or slowdown of the video, which would imply a QoE deterioration. To solve this problem, the following saturation function is introduced:

$$\sigma(x) = \begin{cases} \delta & x > \delta \\ x & -\delta \leq x \leq \delta \\ -\delta & x < -\delta \end{cases} \quad (3.18)$$

that, in the case of an N -element vector, will define $\sigma(x) = [\sigma(x_1), \sigma(x_2), \dots, \sigma(x_N)]^\top$.

Therefore, to guarantee that $|u_i(t)| \leq \delta, \forall i \in \mathcal{V}$, we can write:

$$\dot{x} = \sigma(u) \quad (3.19)$$

By employing the same control strategy as in (3.16), we end up with $\dot{x}_i(t) = \sigma(u_i(t)) = \sigma\left(k_i \sum_{j \in \mathcal{N}_i} a_{ij}(x_j - x_i)\right)$, where $k_i > 0, \forall i \in \mathcal{V}$, are controller gains. Therefore, we can write:

$$\dot{x} = \sigma(-KLx) \quad (3.20)$$

where $K = \text{diag}\{k_1, \dots, k_N\}$.

Before introducing the main result, let us recall the following lemma from [170]:

Lemma 1. *Given a strongly connected digraph G , the associated Laplacian ma-*

trix L has a simple eigenvalue in zero and all the nonzero eigenvalues have positive real part. Let $r = [r_1, \dots, r_N]^\top > 0$ be a left eigenvector of L associated to the zero eigenvalue, i.e., $r^\top L = L^\top r = 0$, and let

$$R = \text{diag}\{r_1, \dots, r_N\}, \quad Q = RL + L^\top R \quad (3.21)$$

Then $R > 0$, $Q \geq 0$ and the kernel of Q has dimension 1 and is given by $\text{span}\{1_N\}$.

We are now ready to prove the following:

Theorem 1. *Consider a multi-agent system represented by a graph G whose dynamics are described by (4.1). Suppose each agent applies the control (3.16) and assume that G is a strongly connected and directed graph. Then the control (3.16) globally asymptotically solves a consensus problem.*

Proof. Let us state the following Corollary of LaSalle theorem from [82]:

Corollary 1.1. *Let $x = 0$ be an equilibrium point for the system $\dot{x} = f(x)$. Let $V : \mathbb{R}^n \rightarrow \mathbb{R}$ be a continuously differentiable, radially unbounded, positive definite function such that $\dot{V}(x) \leq 0$ for all $x \in \mathbb{R}^n$. Let $\mathcal{S} = \{x \in \mathbb{R}^n | \dot{V}(x) = 0\}$ and suppose that no solution can stay identically in \mathcal{S} other than the trivial solution $x(t) \equiv 0$. Then, the origin is globally asymptotically stable.*

Consider (3.20) and let $z = -Lx$. Then, the following dynamics can be defined:

$$\dot{z} = -L\dot{x} = -L\sigma(Kz) \quad (3.22)$$

In order to prove the convergence to 0 of such a system, consider the Lyapunov candidate function [57], [164]:

$$V(z) = \sum_{i=1}^N \int_0^{z_i} r_i \sigma(k_i q) dq \quad (3.23)$$

where r_i is the i -th element of vector r (Lemma 1). It can be observed that $V(z) > 0 \forall z \neq 0$, $V(0) = 0$ and $V(z)$ is radially unbounded, i.e. $V(z) \rightarrow +\infty$ as $|z| \rightarrow +\infty$. By the fundamental theorem of calculus, the derivative of (3.23) yields:

$$\begin{aligned} \dot{V}(z) &= \sum_{i=1}^N \frac{dV}{dz_i} \frac{dz_i}{dt} = \sum_{i=1}^N r_i \sigma(k_i z_i) \dot{z}_i = \sigma(Kz)^\top R \dot{z} = \\ &= -\sigma(Kz)^\top RL\sigma(Kz) \end{aligned} \quad (3.24)$$

Given any $A \in \mathbb{R}^{n \times n}$ and $x \in \mathbb{R}^n$, it is straightforward to show that $x^\top (A + A^\top)x = x^\top Ax + x^\top A^\top x = 2(x^\top Ax)$. Then, recalling that R is symmetric and that $Q = RL + L^\top R$, it follows that:

$$\dot{V}(z) = -\sigma(Kz)^\top RL\sigma(Kz) = -\frac{1}{2}\sigma(Kz)^\top Q\sigma(Kz) \leq 0 \quad (3.25)$$

It can be easily seen that $\mathcal{S} = \{z : \dot{V}(z) = 0\} = \{z : \sigma(Kz) = a1_N\}$, with $a \in \mathbb{R}$. However, if we suppose $a > 0$, then, by definition of the saturation function, it follows that $z_i \geq a/k_i, \forall i \in \mathcal{V}$. Since by Lemma 1 $r > 0$, it follows that if z_i is positive for each i , then we have that $r^\top z = -r^\top Lx > 0$, which is a contradiction because, being r a left eigenvector of L , it must result that $r^\top z = 0$. By using the same arguments, we can also rule out the case $a < 0$ and thus the only possible case is $a = 0$. This accounts for the fact that the only solution that can stay identically in \mathcal{S} is the trivial solution $z(t) = 0$. Hence, by Corollary 1.1, it is possible to state

that z globally asymptotically converges to zero.

Notice that $\bar{z} = -L\bar{x} = 0$, where \bar{x} is a right eigenvector of L associated to the zero eigenvalue. Since the dimension of the eigenspace associated with the zero eigenvalue of L is one, then $\exists \alpha \in \mathbb{R}$ s.t. $\bar{x} = \alpha 1_N$. Therefore, it results that $x_i - x_j \rightarrow 0$ as $t \rightarrow +\infty \forall i, j$, which concludes the proof. \square

As a direct consequence of Theorem 1, we can state the following:

Corollary 1.2. *Consider N users watching the same live streaming event. Suppose users can receive the playback times $T_j(t)$ from a set of clients $j \in \mathcal{N}_i$ according to an established strongly connected digraph whose adjacency matrix is $A = (a_{ij})$. Then, if the playback rate is set as $p_r^{(i)}(t) = 1 + u_i(t)$ with $u_i(t) = \sigma \left(k_i \sum_{j \in \mathcal{N}_i} a_{ij} (T_j(t) - T_i(t)) \right)$ bounded in $[-\delta, \delta]$, where k_i are appropriate control gains, the playback times will be synchronised asymptotically.*

Proof. Recalling that $x_i(t) = T_i(t) - t$, according to Theorem 1, when consensus is reached, $x_i(t) \rightarrow \alpha$ and the playback time $T_i(t) \rightarrow t + \alpha$ for all i when $t \rightarrow +\infty$, thus implying the achievement of consensus also for the playback time. \square

Remark 1. When consensus is reached at a playing time $t + \alpha$ ($\alpha < 0$), although all clients are synchronised, they will watch the live streaming content with a temporal delay of α with respect to the current event time transmitted by the video provider.

Remark 2. Plugging (3.10) into (3.3) for a generic user i under Assumption 1, we obtain the following dynamics of the playout buffer:

$$\dot{q}_i(t) = f_i(t) - p_{r,i}(t) = (f_i(t) - u_i(t)) - 1 \quad (3.26)$$

For the ABR algorithm, $u_i(t)$ can be seen as a disturbance in the filling rate. However, the ABR algorithm is able to reject it in order to avoid buffer depletion and therefore a rebuffering event. At steady state, $u_i = 0$, which implies that when consensus is achieved, the normal behavior $\dot{q}_i(t) = f_i(t) - 1$ is recovered. At this point, the ABR algorithm, responsible for filling the buffer, no longer depends on consensus, which is represented by the control variable $u_i(t)$. Hence, consensus affects the playout buffer dynamics $\dot{q}_i(t)$ but $\dot{q}_i(t)$ does not affect the consensus, and therefore the playback rate, as long as $q_i(t) > 0$, which is always true under Assumption 1.

Theorem 2. *Consider a strongly connected digraph G to which a leader node is added imposing a constant state x_0 . If it is possible to define a spanning tree in the new graph with the leader as its root, then the system (4.1) augmented with the leader node achieves leader-follower consensus under the control strategy (3.16).*

Proof. In this setting, the leader node influences—but it is not influenced by—one or more nodes of the graph through directed edges. Let \bar{G} denote the augmented graph including the graph G , the leader node v_0 and all associated edges. Let $A_1 = \text{diag}\{a_{10}, \dots, a_{N0}\}$, where $a_{i0} > 0$ only if there is a directed link from the leader to node i , otherwise $a_{i0} = 0$, and let $U = L + A_1$, with L as defined in Lemma 1. In order to continue the proof, let us introduce the following [170]:

Lemma 2. *If the augmented graph \bar{G} has a spanning tree with the leader node as the root, then U is full rank. In addition, let*

$$\begin{aligned}
 r &= [r_1, \dots, r_N]^T = (U^T)^{-1} \mathbf{1}_N, \\
 R &= \text{diag}\{r_1, \dots, r_N\}, \\
 W &= RU + U^T R,
 \end{aligned} \tag{3.27}$$

then, $R > 0$ and $W > 0$.

At this point, let $\hat{x}_i = x_i - x_0$, where x_i is the state associated to the i -th node and x_0 is the state imposed by the leader node. Then, $\hat{x} = x - x_0 \mathbf{1}_N$, where x follows the dynamics in (4.1) with the control actions (3.16). It is easy to verify that $\dot{\hat{x}} = \sigma(-KU\hat{x})$ and that by defining $z = -U\hat{x}$, it results that $\dot{z} = -U\sigma(Kz)$. Analogously to the proof of Theorem 1, let us consider the candidate Lyapunov function in (3.23), where now r_i is the i -th element of the vector r defined in (3.27). Its derivative is:

$$\dot{V}(z) = -\frac{1}{2} \sigma(Kz)^\top W \sigma(Kz) \leq 0 \tag{3.28}$$

The candidate Lyapunov function is radially unbounded and is such that $V(z) > 0 \forall z \neq 0$ and $V(z) = 0$ when $z = 0$. Since $W > 0$, then $\dot{V}(z) = 0$ if and only if $z = 0$, otherwise $\dot{V}(z) < 0$. Then, for the Lyapunov stability criterion, z globally asymptotically converges to zero.

Recalling that $z = -U\hat{x}$ and that U is full rank, it follows that $z = 0$ implies $\hat{x} = 0$. Therefore, $x_i \rightarrow x_0$ as $t \rightarrow +\infty \forall i \in \mathcal{V}$. □

3.3 Distributed event-triggered control

The analysis carried out so far was made under the unrealistic assumption that users communicate their playback time to neighbours continuously in time. To limit the overall need for communication—and therefore the number of messages exchanged among users—and the unnecessary control updates, an event-triggered control is introduced. This technique has been widely studied in the literature concerning consensus problems [168], [165], [45]. It consists of updating the control input only when a certain error exceeds a threshold (triggering events). This way, each agent communicates its state only at specific time instants, called *triggering times*. As a consequence, the control action is piecewise constant since it changes only when triggering times occur. Following this idea, an event-triggered linear feedback law for each agent has been designed.

Let $\mathcal{T}_i = \{0, t_1^i, t_2^i, \dots, t_{\phi_i}^i, \dots\}$ be the event-triggering time instants for agent i . Then, the following control strategy can be defined:

$$\hat{u}_i(t) = k_i \sum_{j \in \mathcal{N}_i} a_{ij} (x_j(t_{\Phi_j}^j) - x_i(t_{\Phi_i}^i)) \quad (3.29)$$

where $\Phi_j \in \mathbb{N}, \forall j \in \mathcal{N}_i$, and $\Phi_i \in \mathbb{N}$, denote the fact that different agents have in general different triggering times. Moreover, $t_{\Phi_j}^j \in \mathcal{T}_j$ is such that $t_{\Phi_j}^j = \max\{t_{\phi_j}^j | t_{\phi_j}^j \leq t\}$, where $t_{\phi_j}^j$ is the ϕ_j -th triggering time for agent j . The same can be said for agent i . At this point, we can define the sampled error as $e_i(t) = x_i(t_{\Phi_i}^i) - x_i(t)$, i.e. the difference between the state in the last triggering time instant and the actual value of the state at the present time t . It can be shown that given the following triggering time update:

$$t_{\phi_i+1}^i = \max_{c > t_{\phi_i}^i} \{c \mid |e_i(t)|^2 - \alpha_i e^{-\beta_i t} \leq 0, \forall t \in [t_{\phi_i}^i, c]\} \quad (3.30)$$

with $\alpha_i > 0$ and $\beta_i > 0$, global leader-follower consensus is achieved if and only if the digraph is strongly connected and there exists a spanning tree having the leader node as its root [168].

Then, for each agent $i \in \mathcal{V}$, the system dynamics is:

$$\dot{x}_i = \sigma(\hat{u}_i(t)) \quad (3.31)$$

Notice that the value of $\hat{u}_i(t)$ is kept constant until a new triggering time instant is generated as shown in (3.30). In other words, for agent i , if the error—defined as the difference between the state in the last triggering time instant and the current state—remains below a certain threshold (i.e. $\alpha_i e^{-\beta_i t}$), then the agent will not communicate its state to its neighbours nor will it change its control. This way, whilst the actual state of agent i keeps changing according to (3.31), the value of the state used to compute the control protocols is considered constant. Therefore, between a triggering time instant and the next one, the agent and its neighbours will use the value of the state in the last triggering time to compute the new value of their control protocol. Only at the new triggering time will the agent update its control protocol and send its state to all of its neighbours so that they can update their own control. As simulations will show, the aforementioned approach allows agents to communicate with their neighbours only in discrete time instants. However, due to the presence of a decreasing exponential term in the definition of (3.30), the messages exchanged among agents will never stop. This is also due to numerical issues associated with the specific computer employed for simulations. At this point, one could leverage the concept of *finite-time event-triggered consensus*

to tackle such a problem. In a nutshell, it consists of proving that consensus is reached after a finite time t^* rather than at infinity. This could be a good solution to the previous problem since after t^* no triggering event would happen and, consequently, no more information would be exchanged among agents. Finite-time consensus can be guaranteed by finding a Lyapunov function that satisfies additional and more restrictive conditions.

Several works focus on finite-time event-triggered consensus [172], [171], [46], [56], but, to the best of our knowledge, none of them proves finite-time consensus for the system considered in this work. In particular, finding a Lyapunov function guaranteeing finite-time consensus is not an easy task. For this reason, we consider the following approach: agent i uses the control protocol in (3.29) when $\exists j \in \mathcal{N}_i$ s.t. $|x_j(t_{\phi_j}^j) - x_i(t_{\phi_i}^i)| > \gamma$, where $\gamma > 0$ is a small adjustable parameter, and 0 otherwise. This means that once the playback times of the neighbours are close enough to the playback time of agent i , the draining rate of agent i is set back to 1, i.e. $\hat{u}_i(t) = 0$, thus making $x_i(t)$ constant. Since leader-follower consensus is guaranteed according to [62] under the control protocol (3.29) and the triggering time update given by (3.30), when γ is set to be small and close to zero, the approach just described prevents agents from sending messages indefinitely by stopping the variation of their states and therefore avoiding further triggering times when a *de-facto* consensus is reached, i.e. when all states are close to each other and to the state imposed by the leader. In other words, at steady state, leader-follower consensus is almost achieved. At this point, the maximum lack of synchrony between two neighbouring agents is γ , which is small enough not to be noticeable by users. Let us precise that small communication delays may occur among clients exchanging their current state in a triggering time. This does not impair convergence since, at each agent, the previous control action is kept constant. Therefore, if the change in the control is slightly delayed, consensus will be

achieved anyway but with some additional time. Furthermore, in the worst case, the lack of synchrony between the leader node and any other node is at most $N \cdot \gamma$, which is still negligible and unnoticeable by users if γ is set to be close to zero. As stated in [62], users start to notice differences in the playback time among them when they are above 500 ms. Therefore, such delays should be kept below this threshold.

3.4 Results

Let us now consider some network topologies to show the effectiveness of the proposed approach in baseline cases. Suppose t_s is reached, i.e. a time instant when all clients are playing the video content. Then, it is possible to define the initial states of the nodes identifying the initial time delays T_{0i} that each client experiences when the server sends the streaming content. Let $k_i = 1$ and $|u_i(t)| \leq \delta$, $\delta = 0.3$, $\forall i \in \mathcal{V}$. Throughout the simulations, the following assumption holds: each node sends information to and receives information from all of its neighbours. Notice that this assumption could also be removed as long as the graph stays strongly connected. As a first example, let us consider a directed strongly connected ring topology with $N = 13$ nodes, where each node in the ring represents a client and all clients in the network watch the same live streaming event from different devices. In this topology, each node can communicate only with two neighbours.

As Figure 3.4 shows, consensus is achieved at roughly -17 seconds, which means that $T_i(t) = t - 17$, $\forall i \in \mathcal{V}$, thus implying that all clients are synchronising around a playout time delayed of 17 seconds. It is important to point out that in the simulations the zero corresponds to the chosen t_s .

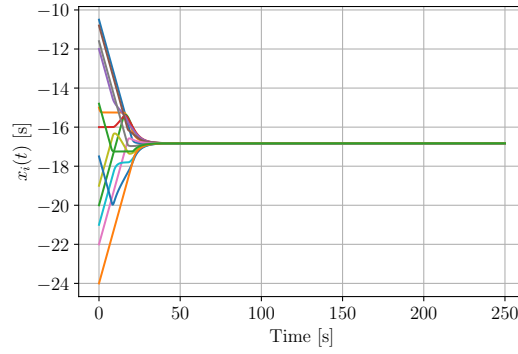


Figure 3.4: State dynamics $x_i(t)$ for the ring topology

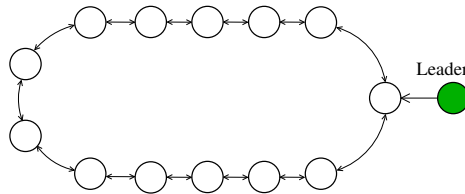


Figure 3.5: Ring topology with a leader node

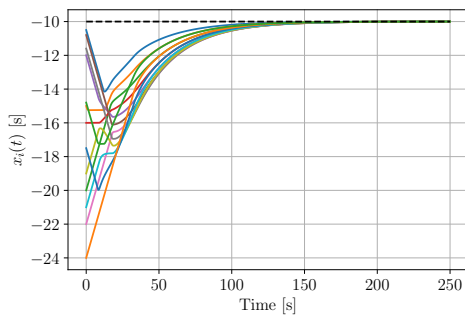


Figure 3.6: State dynamics $x_i(t)$ for the ring topology with a leader node

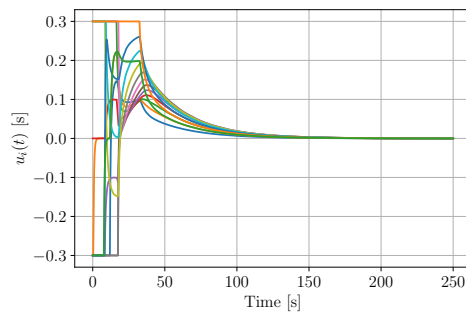


Figure 3.7: Control inputs $u_i(t)$ in the case of a ring topology with a leader node

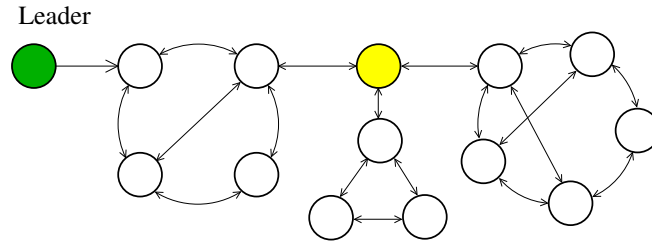


Figure 3.8: Network topology with groups of clients and a leader

In Figure 3.5 the leader-follower approach for the ring topology is considered by adding a fourteenth leader node (green node in the figure), which imposes a state equal to -10 , thus guaranteeing a lower temporal delay.

In the hypothesis that the leader node can communicate only with one other node, consensus is reached at -10 , as expected, (Figure 3.6) after about 150s. The dashed black line in the figure represents the state set by the leader node. Moreover, the transient could be made smaller if the leader is allowed to communicate also with other nodes or by properly increasing the gains k_i . Once consensus is achieved, it results that $T_i(t) \simeq T_j(t) \simeq t - 10$, $\forall i, j = 1, \dots, N$, which implies that all clients are synchronised with a delay lower than the leaderless case (Figure 3.4).

In Figure 3.7 the dynamics of control inputs are shown. As one can see, control variables, after a transient in which most of them experience saturation, converge to zero as desired. The figure shows that for some clients it is necessary to increase the playback rate to make it greater than one because their initial delay was higher. On the other hand, for those clients with a low initial delay, the playback rate is made less than one at the beginning to 'meet' the others and eventually converge together.

Let us now switch to a different topology of the network composed of three groups of clients. Each client in a group is connected to the others through several edges,

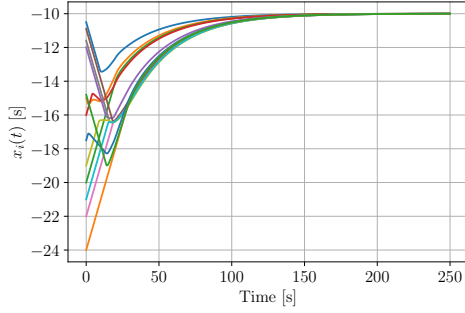


Figure 3.9: State dynamics $x_i(t)$ for the topology of Fig. 3.8

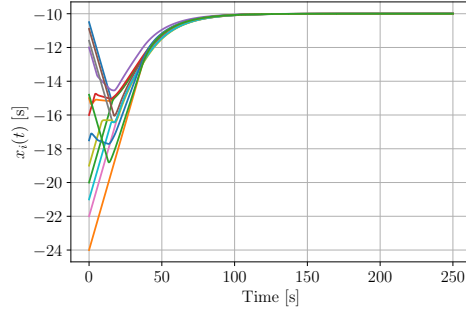


Figure 3.10: State dynamics $x_i(t)$ with the leader marked by the yellow node in Fig. 3.8

while groups are connected with fewer edges as depicted in Figure 3.8. Like in the previous examples, this graph contains 13 nodes with the same initial states and control input bounds. Also, we add a fourteenth leader node (green node in the figure) influencing only one other node in the network with the purpose of making states converge to -10 . In this case, the transient time needed to reach consensus is about 150 seconds (Figure 3.9) due to the different network topologies and the specific node influenced by the leader. Now suppose the leader communicates with the yellow node instead. Then, the time required to achieve consensus with input saturation constraints considerably decreases, up to 100 seconds (Figure 3.10).

To avoid continuous communication among users, and thus an unnecessary exchange of information, an event-triggered control is considered. Let $\gamma = 10^{-4}$, $\alpha_i = 10$ and $\beta_i = 0.1$ for each $i \in \mathcal{V}$. Figure 3.11 shows the trend of the states when a leader-follower approach for the ring topology is employed. As can be observed, convergence is still guaranteed in approximately the same amount of time as the continuous communication case. Notice that the evolution of the states is less smooth due to the abrupt changes in the control actions (Figure 3.12), which, as desired, converge to zero. More precisely, once states get close enough to each

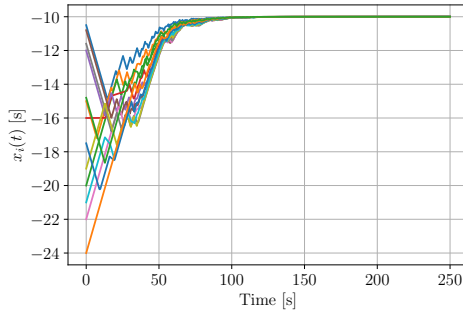


Figure 3.11: State dynamics $x_i(t)$ for the ring topology in the event-triggered case

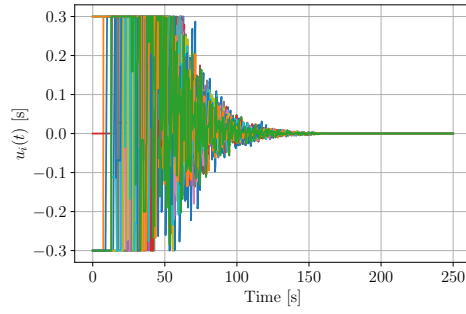


Figure 3.12: Control inputs $u_i(t)$ for the ring topology in the event-triggered case

other and to the state imposed by the leader so that the difference of neighbouring states is less than γ , control inputs, already close to zero, are set to zero. This way, states remain constant, practically synchronised, and will no longer exchange information. In Figure 3.13, it is possible to see the triggering time instants in which an agent updates its control action and sends its new state to the neighbours. Over a total simulation time of 500 seconds, agents exchange information only for the first 300 seconds (roughly), after which no additional communication is needed. Another important point is that users triggered an average of only 87 events. This translates into a considerable decrease in the information exchanged among users, thus implying the effectiveness of the distributed event-triggered control approach.

Finally, such considerations could be generalised to other topologies, such as the one depicted in Figure 3.8, of which we report only the triggering time instants in Figure 3.14. Also in this case, there is no continuous exchange of information and with the proposed approach only a limited number of events is triggered. Again, whilst the simulation time is 500 seconds, the figure shows only the simulation until 200 seconds since nothing happens afterwards.

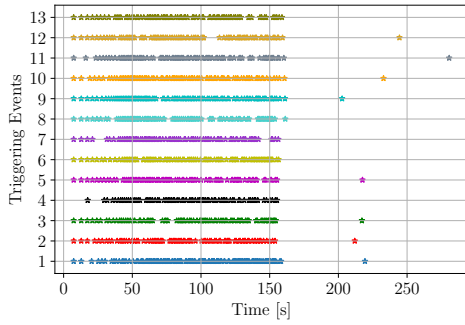


Figure 3.13: Triggering times for each agent in the ring topology

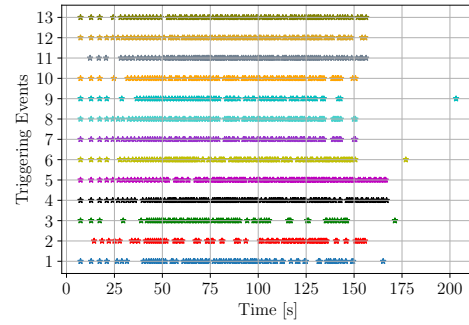


Figure 3.14: Triggering times for each agent in the topology of Fig. 3.8

Depending on the network topology, there are some nodes that have more influence with respect to others. Since the topology is not usually known a priori, the more nodes the leader is able to influence, the faster, in general, the achievement of synchronisation. It is also important to highlight that convergence is guaranteed independently of the value of initial states, the number of nodes, the number of nodes influenced by the leader, and the network topology—as long as the graph remains strongly connected. Variations of all such parameters will only positively or negatively affect the synchronisation time, which is also influenced by the saturation value δ that, according to a recent study [125], should not exceed 10% of the original rate not to impact the QoE. Moreover, initial delays associated with clients are not usually high. Actually, the number of available video segments encoded by the DASH standard when a client connects to the server is of the order of 5-10. Considering the typical case of segments of duration $\tau = 5$ s, the clients' playback time can be delayed with respect to the video provider of an amount that does not exceed 50 seconds.

3.5 Concluding Remarks

In this Chapter, a distributed control approach to synchronise clients watching live streaming content in geographically distributed locations has been presented. To achieve synchronisation, the playback rate, i.e. the speed at which the video is played, has been chosen as the control variable. This approach is known as Adaptive Media Payout, where the playback rate is limited and can only be slightly adjusted to avoid QoE degradation. To this end, the widely studied consensus problem of simple integrators with input saturation has been leveraged to design a distributed playback synchronisation framework. In addition, to reduce the delay between agents and video content providers, a leader-follower approach has been employed. To enhance this framework and limit the amount of information exchanged among agents, we have adopted an event-triggered control protocol. With such a protocol, it has been possible to drop the assumption of continuous communication among users. Finally, in order for the communication to come to an end, the event-triggered control at the agents is turned off when their states are close enough to each other. Therefore, the consensus process stops when synchronisation is achieved in practice. Simulations on different network topologies prove the effectiveness of the proposed approach, which guarantees asymptotic synchronisation along with the satisfaction of playback rate saturation constraints independently of the protocols adopted.

Part II

Asymptotic Stability of Systems with Deep Reinforcement Learning Controllers

Chapter 4

On Asymptotic Stability of Nonlinear Systems with Deep Reinforcement Learning Controllers

In this Chapter, we tackle the problem of guaranteeing the asymptotic stability of systems controlled with Deep Reinforcement Learning (DRL) control strategies. After extracting the DRL control policy, a framework trying to synthesise a Lyapunov function is designed. This is an important development in the direction of safety. Results shown in this Chapter are in [\[102\]](#).

4.1 Introduction

Deep Reinforcement Learning (DRL) is a branch of Machine Learning (ML) that is specifically designed to solve control problems. In particular, DRL methods aim at learning control policies through interaction with an environment. However, one of the prominent reasons that hinders the adoption of such learning-based approaches in real control applications is safety [9]. Although DRL control policies can achieve promising performance and provide excellent results, most of the systems employed in reality are safety-critical, usually because of their interaction with human beings and/or with equipment that could be damaged when unsuitable control actions are taken by the control policy.

Therefore, it is important to ensure the learning process produces *safe control policies*. In general, a state is considered safe if system trajectories are bounded within a region and eventually converge asymptotically to the equilibrium point under a given control policy. When DRL methods are applied to nonlinear continuous time systems, the disadvantage is that there is no guarantee that the learned control policy always stabilises the system as prescribed. This is due to the fact that, during the training phase, the DRL algorithm cannot exhaustively explore all possible states since they are infinite. As a consequence, when the trained control policy is actually deployed, it could occur that some unexplored states are visited, in which case the system dynamics may diverge.

A common workaround that is adopted when dealing with nonlinear systems is to linearise the system dynamics around an equilibrium point and to use the theory of linear systems to compute a linear feedback controller (e.g. using a Linear Quadratic Regulator) that guarantees stability in a small neighbourhood of the equilibrium point where the linear approximation remains valid. However, such

controllers are no longer valid outside the neighbourhood, hence they prove of little use in several applications. To overcome this issue, Lyapunov methods are adopted to synthesise controllers valid outside the small neighbourhood of the equilibrium point. A common method relies on polynomial approximations of the system dynamics and seeks sum-of-squares (SOS) polynomials as Lyapunov functions using semidefinite programming (SDP) [123], [77]. However, also in this case, one would work with just an approximation of the system and therefore a restriction on the control.

The aim of the work presented in this Chapter is to check whether the control policy obtained with a DRL algorithm makes the system asymptotically stable. To this purpose, the aim is to derive stability certificates to prove the asymptotic stability of nonlinear systems at an equilibrium point in the presence of *deterministic control policies* obtained with DRL. DRL techniques are particularly useful when it comes to deriving a control policy needed to control a continuous-time system without any approximation of its dynamics. However, there is no guarantee that such a policy is always able to successfully control the system. This controller can be plugged into the nonlinear model of the system and then a *Lyapunov Neural Network* (LNN) can be implemented to synthesise a Lyapunov function, which provides a stability certificate for the system. In our framework, a DRL module derives the best feedback control possible given the system dynamics and a reward function. To synthesise a Lyapunov function, a recently proposed *Learner-Verifier* approach is adopted [29], [3]. This approach consists of a *learner* that trains an LNN over a set of sample states with stochastic gradient descent to tune proper parameters for a Lyapunov candidate function in such a way that a specific cost—representative of the violation of the Lyapunov conditions—is minimised. The *verifier* retrieves the DRL control policy and the Lyapunov candidate function from the learner and checks whether the candidate is actually a Lyapunov function. If

it is not the case, the verifier provides a certain number of counterexamples that are added to the set of sample states and feeds them back to the learner so that the procedure can iterate until a Lyapunov function is found or a maximum number of iterations is reached. It is important to point out that in general it is not easy to verify the Lyapunov conditions and that for this reason we employ a sound decision procedure using Satisfiability Modulo Theory (SMT) [12] called δ -complete decision procedure [60]. Such a procedure guarantees correctness of the Lyapunov function found in terms of fulfilment of the Lyapunov conditions. In the following chapters we provide a learning-based methodology to prove asymptotic—and not only practical—stability at the equilibrium point of a nonlinear system with a deterministic DRL control policy. Then, the region of attraction of a system at an equilibrium point with a deterministic DRL feedback control policy is computed, thus bounding the trajectories of the system when the initial state lies inside the region. With our approach, it is possible to show that a DRL feedback controller can ensure wider regions of attraction compared to other state-of-the-art controllers.

4.2 Related Work

In [133], the authors present an approach to learn safety certificates for nonlinear discrete-time closed-loop dynamical systems. The procedure is based on sampling methods, which are used to verify the Lyapunov conditions, and on a manual design of the neural network to derive the Lyapunov function given a controller. This method, along with [109], [118], [126], does not provide formal numerical soundness. The approach proposed in this Chapter first derives a control policy using a DRL algorithm and then relies on a generic feed-forward network to obtain a Lyapunov function with a *Learner-Verifier* method, which is formally sound because it is based on SMT solvers. Mehrjou *et al.* [108] provide an improvement

of the algorithm in [133] to enlarge the region of attraction only of a specific class of systems.

In [29] a relevant approach is shown. In particular, the authors leverage feed-forward neural networks and SMT solvers to synthesise both the control and the Lyapunov function. However, only Lagrange–or practical–stability is guaranteed, i.e. stability is not guaranteed in a neighbourhood of the equilibrium point. Conversely, the aim of the approach described in the following sections is to guarantee full asymptotic stability and compare our results with the aforementioned work to show that we are still able to synthesise Lyapunov functions and that the controller obtained with DRL techniques ensures even larger regions of attraction. Abate *et al.* [3] propose a method for the formal synthesis of LNNs to ensure full asymptotic stability of autonomous nonlinear systems. A polynomial Lyapunov function is derived and comparisons with other approaches prove the efficiency of the method in terms of computation time needed to synthesise a Lyapunov function. The same authors have developed a tool called FOSSIL [2] for Lyapunov function and barrier certificate synthesis employing the aforementioned approach. The focus of this work is mainly computation time efficiency for autonomous nonlinear systems without analysing the regions of attraction. Our method is based on the computation of a DRL control policy to prove its suitability in terms of asymptotic stability and on the analysis of the regions of attraction compared to other methodologies.

4.3 Preliminaries

Before describing the procedure to obtain a deterministic DRL controller and to synthesise a Lyapunov function that certifies its stability, relevant preliminary background and notation used in this Chapter is provided.

Lyapunov stability theory. Consider the continuous-time time-invariant nonlinear system:

$$\dot{x} = f(x, u) \quad (4.1)$$

where $f: \mathcal{D} \rightarrow \mathbb{R}^n$ is a Lipschitz-continuous vector field and $\mathcal{D} \subseteq \mathbb{R}^n$ is the domain of the system containing all states $x(t)$ and such that $0 \in \mathcal{D}$. The continuous function $u: \mathcal{D} \rightarrow \mathbb{R}^p$ is the static state-feedback control policy to be learned, thus we can write $\dot{x} = f(x, u(x))$. Throughout this Chapter, we will consider w.l.o.g. that $x = 0$ is an equilibrium point for system (4.1).

According to a well-known theorem in [81], let $f(x, u(x))$ be a locally Lipschitz continuous vector field with an equilibrium point at the origin. Let $V: \mathcal{D} \rightarrow \mathbb{R}$ be a continuously differentiable function such that $V(0) = 0$, $V(x) > 0 \forall x \in \mathcal{D} \setminus \{0\}$ and $\dot{V}(x) < 0 \forall x \in \mathcal{D} \setminus \{0\}$. Then, the origin is an asymptotically stable equilibrium point of the system and V is called a Lyapunov function.

Notice that $\dot{V}(x)$ represents the Lie derivative of $V(x)$ over the vector field $f(x, u(x))$ and is defined as follows:

$$\dot{V}(x) = \nabla_x V(x) \cdot f(x, u(x)) = \sum_{i=1}^n \frac{\partial V}{\partial x_i} f_i(x, u(x)). \quad (4.2)$$

Despite its relevance in determining the asymptotic stability of a system, finding a Lyapunov function is in general not an easy task and, up to now, there exists no general method to construct Lyapunov functions [65]. These functions are also important tools to estimate the regions of attraction at an equilibrium point of a general nonlinear dynamical system [81]. Suppose system (4.1) is asymptotically stable at the origin under the control law $u(x)$ and let $V(x)$ be a Lyapunov function

for system (4.1) in \mathcal{D} . A region of attraction \mathcal{S} is an invariant subset of \mathcal{D} that contains the origin, i.e. if the initial state of the system belongs to \mathcal{S} , then the system trajectories always stay inside \mathcal{S} . All level surfaces of $V(x)$ contained in \mathcal{D} are regions of attraction for the system, i.e. for some $c > 0$, $\mathcal{S} = \{x \in \mathbb{R}^n | V(x) \leq c\} \subseteq \mathcal{D}$.

Neural Networks. Neural networks find their application in several fields. They have been employed also to enhance Reinforcement Learning (RL) agents to derive policies for the control of continuous-time systems. Hence, the birth of Deep Reinforcement Learning (DRL), which allows us to scale to decision-making problems that were previously intractable, i.e. settings with high-dimensional state and action spaces. Two DRL algorithms with a deterministic control policy will be considered throughout this work: *Deep Deterministic Policy Gradient* (DDPG) [93] and *Twin Delayed Deep Deterministic Policy Gradient* (TD3) [58]. The DDPG algorithm is one of the most widely used learning techniques applied to control systems. It is based on an off-policy Actor-Critic architecture to learn control policies in continuous action spaces. In a nutshell, the actor chooses an action to perform on the environment and observes the reward and the new state. Such a transition is stored in the *replay buffer*. The critic measures how good the action chosen by the actor was through the Q-function. Then, it samples a number of transitions from the replay buffer to minimise a loss function and to update the actor policy, i.e. the control policy employed to control the environment. The TD3 algorithm improves the DDPG by introducing two Q-functions instead of just one and uses the smallest of the two Q-values to compute the loss function. In addition, TD3 does not update the policy at the same frequency as the Q-function but, usually, it is updated every two Q-function updates.

Neural networks are also efficient regressors. This property is important in that it

allows us to approximate Lyapunov functions. Such neural networks are usually called Lyapunov Neural Networks (LNNs). Several works have employed LNNs for different purposes [109], [118], [141]. In this work, we implement a feed-forward neural network as an LNN whose input has the dimension of the state space of the system, say n , the output layer has dimension 1, and in between, there are m hidden layers with h_1, h_2, \dots, h_m neurons. Supposing fully connected layers, the weights are contained in $m + 1$ matrices as follows: $W_1 \in \mathbb{R}^{h_1 \times n}$ for those weights from the input to the first hidden layer, $W_2 \in \mathbb{R}^{h_2 \times h_1}$ for the weights from the first hidden layer to the second and so on until W_{m+1} . Each hidden neuron may also present an additive bias, therefore $B_1 \in \mathbb{R}^{h_1}$, $B_2 \in \mathbb{R}^{h_2}$, and so on. Every layer can have an activation function $\sigma : \mathbb{R} \rightarrow \mathbb{R}$ applied to each neuron. Hence, if x_0 is the input state, it follows that:

$$l_1 = \sigma_0(W_1 x_0), \quad (4.3)$$

$$l_i = \sigma_i(W_i l_{i-1}), \quad i = 2, \dots, m, \quad (4.4)$$

where σ_i is computed element-wise and l_i is the output of the i -th layer. The output of the feed-forward network represents the value of the Lyapunov function in the input state:

$$V(x_0) = \sigma_{m+1}(W_{m+1} l_m). \quad (4.5)$$

Once this network is trained to meet the Lyapunov conditions, we end up with an LNN that represents a Lyapunov function $V : \mathcal{D} \rightarrow \mathbb{R}$ proving the asymptotic stability of the controlled system at the equilibrium point in the domain \mathcal{D} .

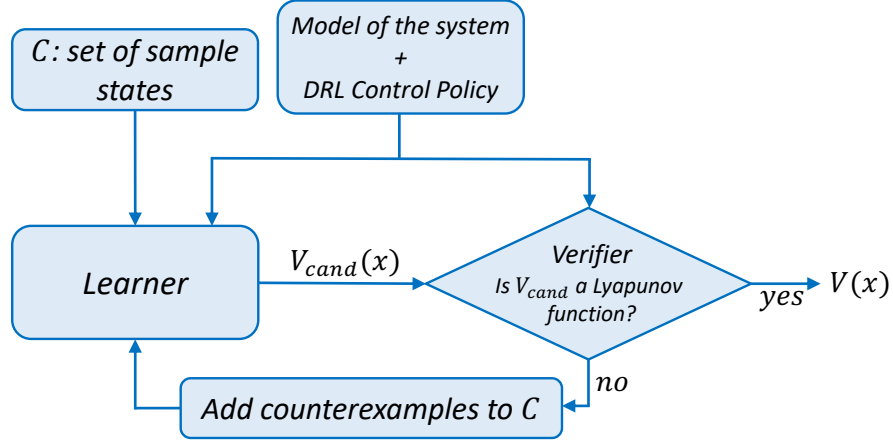


Figure 4.1: Proposed architecture

4.4 The Proposed Architecture

In this Section, we present an architecture that extracts a DRL control policy and proves—if it is the case—full asymptotic stability of the closed-loop feedback system through the synthesis of a Lyapunov function and a corresponding region of attraction in the domain of the system.

On the basis of the counterexample-guided approach illustrated in [3], Fig. 4.1 shows the proposed architecture composed of three main modules: the *model of the system with the DRL Control Policy*, the *Learner* and the *Verifier*. Below we provide a more detailed description of each component.

- *DRL Control Policy*: it employs the model of the system and is in general based on a DRL algorithm that outputs the control policy $u(x)$ after a training phase. Since we aim at extracting the control policy as a function of the states, the DRL algorithm needs to be deterministic, i.e. the output of

the algorithm must provide an action when an input state is presented rather than an average value and a standard deviation of the action, which would imply stochasticity.

- *Learner*: a neural network is trained on the basis of an input set \mathcal{C} containing a certain number of sample states. The trained network represents a candidate LNN whose output, given a state, is the value of the candidate Lyapunov function in that state. The neural network is trained by minimising a cost function using the steepest gradient descent. The cost function must necessarily take into account the Lyapunov conditions that have to be necessarily satisfied to obtain a Lyapunov function. Let us formulate such a cost as in [29]:

$$L(x) = \frac{1}{N} \sum_1^N (\max(-V(x_i), 0) + \max(0, \dot{V}(x_i))) + V(0)^2 \quad (4.6)$$

where x_i is the i -th sample of the state and is such that $x_i \in \mathcal{C}$ with $|\mathcal{C}| = N$. Notice that for each sample, the cost is 0 when $V(x) > 0$, i.e. when the first Lyapunov condition is met, and $\dot{V}(x) < 0$, i.e. when the second Lyapunov condition is met. If $V(x) < 0$, then the first condition is violated and such a violation is penalised of a term equal to $-V(x)$. The same can be said for the condition $\dot{V}(x) < 0$. The average over all samples plus $V(0)^2$ represents our cost, which is minimised using the steepest gradient descent. Obviously, when $V(x)$ is a Lyapunov function, $L(x) = 0$, although the contrary is not true. In fact, it could happen that there are some states not included in \mathcal{C} in which the candidate function violates the conditions. It is at this point that the *Verifier* checks if there are some states that violate the Lyapunov conditions and adds them to \mathcal{C} .

- *Verifier*: it makes use of a solver to find—if it is the case—states that violate the Lyapunov conditions w.r.t. a candidate Lyapunov function. Satisfiability Modulo Theory (SMT) solvers are powerful tools to check the satisfiability of first-order logic formulae. Such solvers are formally sound and therefore provide guarantees that are equivalent to giving analytical proofs. For this reason, SMT solvers are employed to check candidate Lyapunov functions. As already said, a Lyapunov function has to meet the following conditions:

$$V(0) = 0 \wedge V(x) > 0 \wedge \dot{V}(x) < 0, \forall x \in \mathcal{D} \setminus \{0\}. \quad (4.7)$$

To exploit the nice properties of SMT solvers, the so-called *dual falsification problem* is derived. Such a problem is needed in formal verification and is simply the negation of (4.7):

$$\exists x \in \mathcal{D} \setminus \{0\} : V(x) \leq 0 \vee \dot{V}(x) \geq 0. \quad (4.8)$$

If the falsification condition is true for some nonzero x in \mathcal{D} , then $V(x)$ is not a Lyapunov function since there is at least one state in the domain that violates the Lyapunov conditions. On the contrary, if it is false, it means that $V(x)$ is a Lyapunov function in \mathcal{D} . To solve problem (4.8), it is necessary to globally minimise non-convex functions such as Lie derivatives, which is an NP-hard problem. This is the reason why the *Verifier* relies on an open-source SMT solver for nonlinear formulas over the reals called dReal [61]. This solver provides formally sound solutions to problem (4.8), i.e. if a solution exists, it is always found. Such a guarantee comes with the δ -completeness property. For the sake of clarity, let us provide the following definitions [60].

Definition 1 (SMT problem). *An SMT problem is the problem of determining whether an SMT formula, which is a first-order logic formula, is satisfiable.*

Definition 2 (δ -completeness). *Let φ be an SMT formula and δ a positive rational number, a decision procedure \mathcal{P} is δ -complete if it either determines that φ is not satisfiable or that the δ -weakening of φ is satisfiable.*

The δ -weakening of φ is a numerical relaxation of the original formula. In our case, a δ -complete procedure \mathcal{P} that verifies the formula in (4.8) is what we need because if a formula is satisfiable, then its δ -relaxation is always satisfiable. On the contrary, if the δ -relaxation of the formula is satisfiable, then it could be that either the formula itself is satisfiable or it is not satisfiable. Therefore, we can guarantee the check on the Lyapunov conditions because if the δ -relaxation of (4.8) is not satisfiable, then also (4.8) itself is not satisfiable, thus implying that $V(x)$ satisfies the Lyapunov conditions and is a Lyapunov function. However, it could happen that the δ -relaxation of (4.8) is satisfiable. This provides no guarantee on (4.8) and the procedure gives counterexamples that are solutions of the relaxed formula. If the not-relaxed formula is not satisfiable, then the counterexamples obtained are just spurious solutions that do not pose any issue since this could translate into the synthesis of a more conservative candidate Lyapunov function.

4.5 Results

Experimental results of the proposed approach are provided in this Section. We picked two test cases and, as already said, we considered two of the most com-

mon deterministic DRL algorithms: DDPG and TD3. In the *Learner* we set up a feed-forward neural network with one hidden layer while in the *Verifier* we use dReal [61] as the SMT solver.

Case 1: Inverted Pendulum. The first test case consists of controlling an inverted pendulum whose control goal is to balance it in the upright position with zero angle and zero angular velocity. The torque that can be applied to the hinge of the pendulum represents the control variable whereas the state is represented by the angle, measured in radians, and the angular velocity, measured in radians per second. Finally, a domain $\mathcal{D} = \{x = (\theta, \dot{\theta}) \in \mathbb{R}^2 \mid \|x\|_2 \leq 6\}$ is imposed.

Let us start by considering a DDPG control policy. As a first step, the Actor and Critic are trained until the cost function is minimised and the control goal is achieved. In particular, an Actor with the two states as inputs, no hidden layers, and one output with linear activation is set. In other words, a controller of the form $u(x) = Kx$ is considered, thus obtaining: $u(\theta, \dot{\theta}) = -1.2616117 \theta - 0.29633152 \dot{\theta}$.

The next step is to create a set \mathcal{C} of 500 random states sampled from the domain and, on the basis of such a set and of the model of the system, the *Learner* derives a candidate Lyapunov function that is passed to the *Verifier*. The latter uses the SMT solver to check if the Lyapunov conditions are verified. In the case they are not, the solver augments \mathcal{C} with some states that violate the Lyapunov conditions in such a way that, at the next iteration, the *Learner* can adjust the Lyapunov candidate function to satisfy the conditions. Notice that, according to what we have said in Section 4.4, it could happen that the *Learner* finds a Lyapunov function but the *Verifier* does not recognise it due to the δ -completeness property of the SMT solver. In this case, the counterexamples added to \mathcal{C} are spurious data that are not detrimental to the *Learner*: they can help it find a more suitable Lyapunov

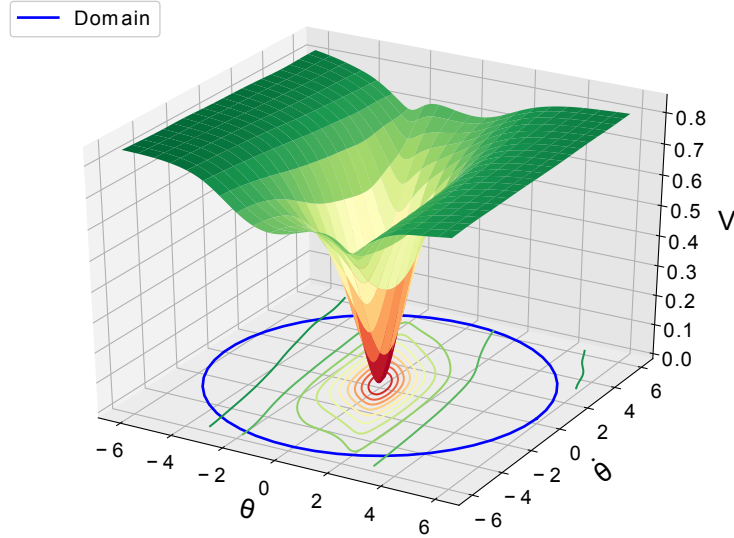


Figure 4.2: Lyapunov function for Inverted Pendulum with DDPG control

function that, eventually, will pass the *Verifier*'s check. It is worth stressing that the contrary can never happen, i.e. the *Verifier* is always able to detect candidate Lyapunov functions that violate the conditions.

Fig. 4.2 shows the Lyapunov function $V(\theta, \dot{\theta})$ found by the *Learner* and valid inside \mathcal{D} . Qualitatively, one can see that this function is 0 only when $(\theta, \dot{\theta}) = (0, 0)$ and is strictly positive elsewhere in the domain (blue circle in the figure). The *Verifier* formally ensures this along with the condition on the gradient. Only now do we have the guarantee that the DDPG control policy asymptotically stabilises the nonlinear system also when unexplored states are visited.

As already explained, any level set of the Lyapunov function entirely lying within the domain is a region of attraction for the system. Therefore, we can consider the widest region of attraction in the domain to have an idea of the stabilising performance guaranteed by the controller and, consequently, of its quality. Fig. 4.3 outlines the region of attraction obtained with the synthesised Lyapunov function (green curve) and the phase portrait of the closed-loop system (grey arrows)

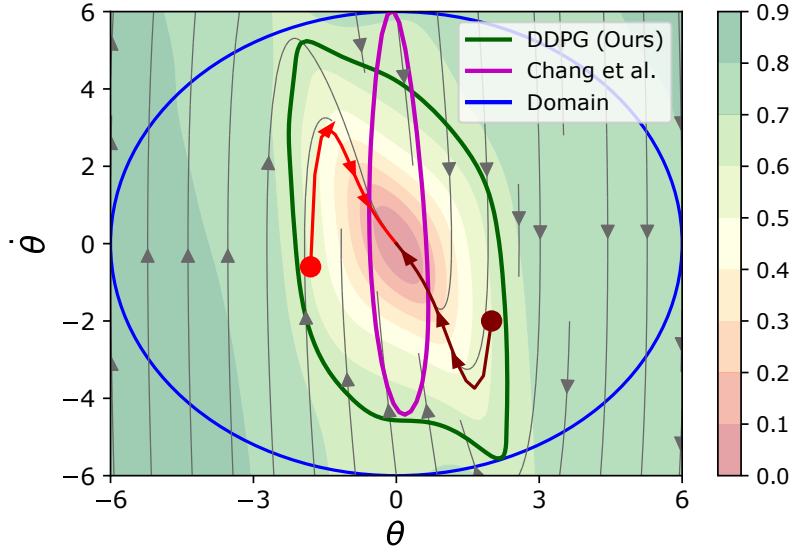


Figure 4.3: Region of Attraction for Inverted Pendulum with DDPG control

showing that the equilibrium point in zero is a stable point. To provide a better grasp of the concepts, if we pick two random initial states (large dots in the figure) within the region of attraction and let the system evolve, the trajectories will never leave the region until convergence. As the figure shows, it is possible to see that the trajectories evolve but never leave the region until convergence. By looking at Fig. 4.3, it is possible to observe that if the initial state lies outside the highlighted green region, then the system trajectories can evolve arbitrarily far before converging to zero. This is an important insight into tackling safety issues concerning DRL control strategies. In fact, safety is the main reason why such approaches meet few practical applications. Therefore, synthesising a Lyapunov function for the DDPG controller in the case of an inverted pendulum not only guarantees asymptotic stability of the controlled system, but also allows us to find a region of attraction, which—provided that the initial state is in it—ensures that no state outside it is visited, thus preventing the system from reaching possible unsafe states.

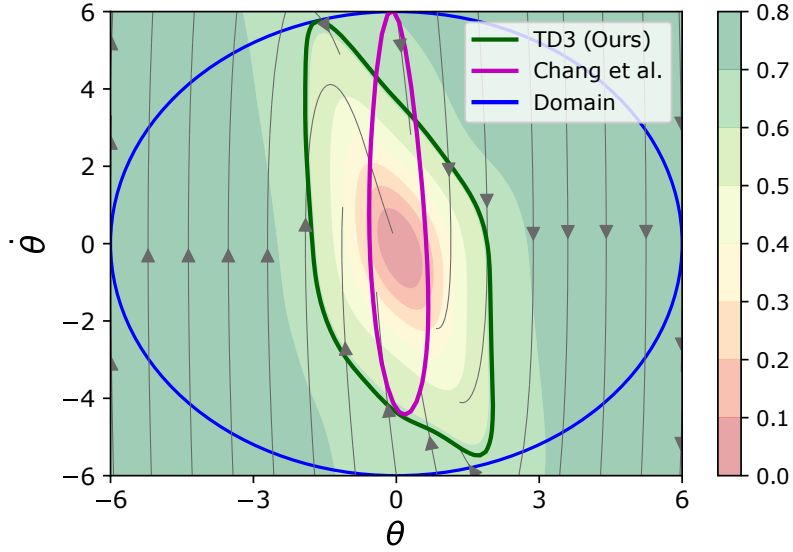


Figure 4.4: Region of Attraction for Inverted Pendulum with TD3 control

In Fig. 4.3 it is also shown the region of attraction found with the controller developed by Chang *et al.* [29] for the same example (magenta curve). This linear controller is synthesised by the same neural network that generates the candidate Lyapunov function. Even if the goal of the present work is different, let us make some considerations on the kind of controllers found. From the figure, it is clear that the DDPG controller guarantees a much wider region of attraction w.r.t. to the other controller. In other words, it would prove a more performing control. Moreover, the control in [29] guarantees in practice Lagrange stability, thus excluding a neighbourhood of the equilibrium point when synthesising the Lyapunov function. On the other hand, our analysis ensures full stability at the equilibrium point yet with a wider region of attraction.

Similarly, the TD3 control policy was evaluated, obtaining essentially similar results. In particular, after finding the Lyapunov function, which has a similar shape as in Fig. 4.2, we show the corresponding region of attraction in Fig. 4.4. Also in this case, the TD3 controller is able to guarantee full stability and still a wider

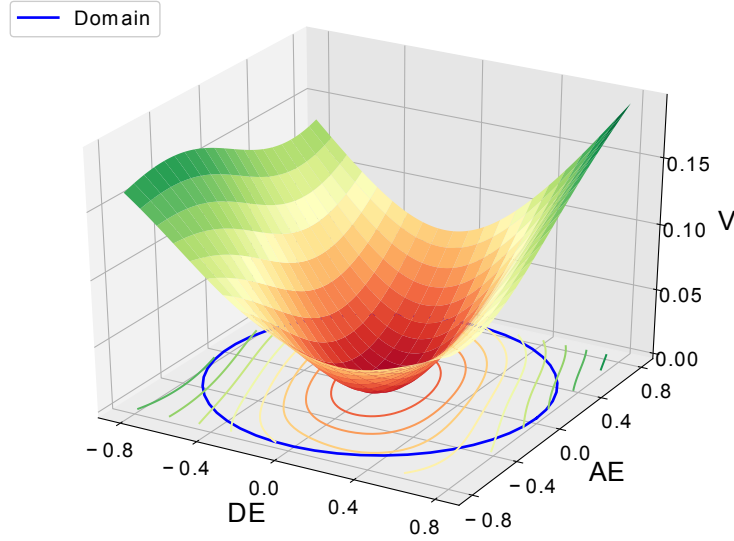


Figure 4.5: Lyapunov function for Bicycle circle tracking with DDPG control

region of attraction compared to the case described in the literature.

Case 2: Bicycle circle tracking. In this example, we consider the case of a bicycle, running at a constant speed, that has to track a unit circle. The control variable is the steering angle of the front wheel whereas the state variables are represented by the Distance Error (DE), i.e. the distance measured in meters between the rear wheel and the circle, and the Angle Error (AE), i.e. the difference measured in radians between the angle of the rear wheel and the angle of the tangent to the circle in the closest point to the wheel (see [29] for more details). The control goal is to steer the state to zero. In this case, a domain $\mathcal{D} = \{x = (DE, AE) \in \mathbb{R}^2 \mid \|x\|_2 \leq 0.8\}$ is set.

Let us first evaluate the DDPG algorithm. Notice that in this case we set up the Actor with an input layer of dimension 2, two hidden layers with 256 neurons, and an output layer of dimension 1, all with linear activation, thus resulting in a linear state feedback control. After training a control policy able to stabilise the system in the testing phase, stability is evaluated. To this end, the framework displayed

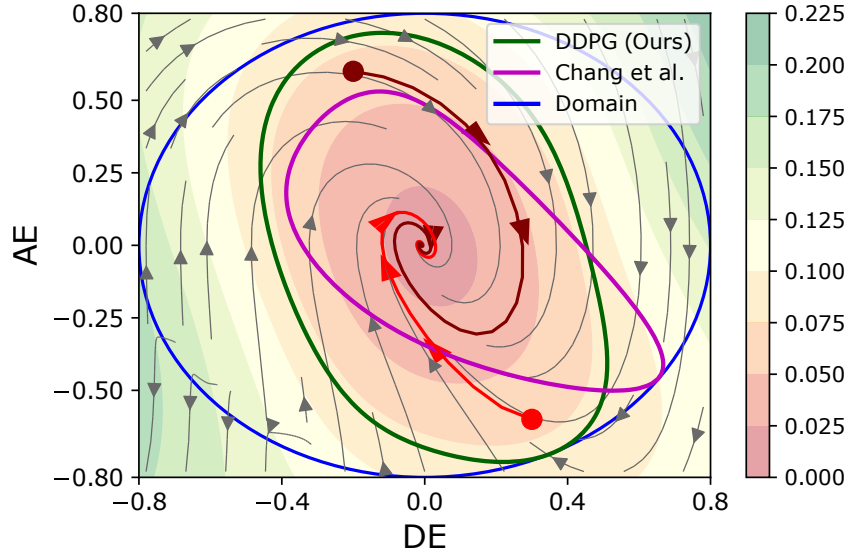


Figure 4.6: Region of attraction for Bicycle circle tracking with DDPG control

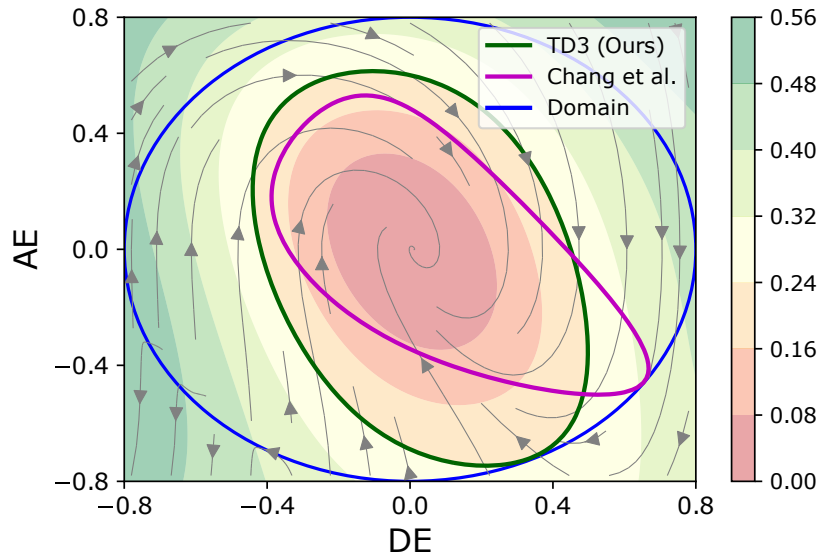


Figure 4.7: Region of attraction for Bicycle circle tracking with TD3 control

in Fig. 4.1 is employed to try to synthesise a Lyapunov function that proves the asymptotic stability of the controlled system. The *Learner* managed to find a Lyapunov function, which is shown in Fig. 4.5. From its level sets, it is possible to find a region of attraction for the system (green curve) and the phase portraits (grey arrows) as shown in Fig. 4.6. The region is considerably wide in the domain of the system, ensuring that the trajectory of the state never leaves it during its evolution until convergence every time the initial state lies in the region. Also, we show two trajectories for clarity and the smaller region of attraction found by Chang *et al.*

Finally, we carried out the same analysis for the TD3 algorithm. A Lyapunov function—similar to the DDPG case—was successfully found. Fig. 4.7 shows the region of attraction from which the same conclusions as the DDPG control can be drawn.

In each of the considered benchmark control systems, the region of attraction obtained with a DRL controller is also way wider than the region derived with classical controllers such as LQR or SOS (see [29] for a visual comparison).

As already explained, in general, if no Lyapunov function is found, then one cannot establish any conclusion about the system's stability. In addition, there is no way of proving the non-existence of a Lyapunov function for a system. Therefore, it could happen that the LNN either finds a Lyapunov function or reaches a maximum number of iterations. The latter case simply implies the network was not able to find a Lyapunov function that could exist. Moreover, the more the system is complex, the wider the networks employed in the DDPG or TD3 to control it. However, as long as activation functions on neurons are linear, the resulting controller will be linear or affine in the state and therefore the LNN can be easily deployed. In the case of nonlinear activation functions, the time complexity of

the proposed procedure can considerably increase. Time complexity also depends on the complexity of the Lyapunov function we try to synthesise. In this work, we considered linear or affine controllers and an LNN with one hidden layer, 64 neurons, and a *tanh* activation function. It is important to notice that non-smooth networks cannot be employed since, in order to analytically check the Lyapunov conditions, Lie derivatives must exist. For this reason, activation functions like ReLU cannot be considered. In the simulations, the time needed for the synthesis ranges from 132 to 431 seconds when using Google Colab. In any case, this is an offline procedure, whose goal is to certify the asymptotic stability of the controlled system without taking into account the time required to accomplish such a task.

Finally, the approach just described relies on the availability of a good model of the control system and assumes no uncertainty is present. In other words, the trained DRL control policy is not robust. To apply the proposed framework to real-world control systems, the DRL algorithm has to be trained on a simulator, on which usually only the nominal model is available. However, to apply the trained control policy to the real-world system, we need stability guarantees on a robust controller that takes into account parametric uncertainties. The approach proposed in [37] could be leveraged to synthesise a learning-based function able to provide both stability and safety guarantees for uncertain control-affine dynamical systems while generating a robust controller. Although some important rigorous verification is still missing, we believe that such a strategy can be developed further and integrated into our approach to address also the problem of robustness for practical applications.

4.6 Concluding Remarks

This Chapter proposes a methodology that can be employed to prove the asymptotic stability of nonlinear systems when deterministic DRL control policies are used. Such an analysis has been carried out on two different DRL algorithms: DDPG and TD3. After a training phase, in which the DRL algorithm learns how to achieve the control goal, the corresponding control policy is extracted and a *Learner-Verifier* approach is implemented to synthesise a Lyapunov function with the help of SMT solvers and LNNs. The results obtained on two benchmark control systems show the effectiveness of the proposed approach, ensuring not only the asymptotic stability of DRL algorithms but also important considerations on safety.

Chapter 5

Conclusions and Future Research Directions

In this thesis, the first and main part is devoted to video streaming and Internet media delivery services, whereas the second part represents the beginning of a study concerning the synthesis of stability certificates when Deep Reinforcement Learning controllers are employed.

After providing an overview of video streaming protocols, Quality of Service and Quality of Experience have been defined and analysed. Finally, basic concepts of Machine Learning and, more precisely, Reinforcement Learning have been introduced. Regarding the first part, a Multi-Commodity Flow Problem (MCFP) has been described to address the issue of designing a QoE-fair optimal allocation strategy. The motivation lies in the high user engagement and Quality of Experience video service providers have to keep when delivering videos to massive audiences in order to avoid service abandonment. In the context of the MCFP optimisation framework, a Proportional Fair (PF) resource allocation strategy has

been adopted to choose the appropriate path or paths for each demand in order to equalise the QoE obtained by concurrent heterogeneous users for video delivery networks.

Subsequently, a clustering of video sessions has been proposed with the purpose of making the number of variables involved in the optimisation problem manageable. The performance of the proposed PF allocation strategy has been compared to the case of a QoE-unaware allocation strategy, which is representative of the currently deployed video delivery networks. Simulations have shown that the proposed PF allocation strategy is able to considerably improve fairness among heterogeneous clients while still keeping the same average visual quality. An interesting future direction would be carrying out an in-depth analysis of the video clustering procedure to further optimise the network resource allocation. Another development could be implementing a decentralised technique to solve the optimisation problem in order to improve scalability-related issues as well as resiliency to computing failures, which could be caused by malicious users attacking the node where the optimiser is located. Furthermore, another future perspective is to conduct the analysis so far presented with a more complicated QoE function, which not only depends on the visual quality but also on rebuffering events and several other features that are still object of research. One last aspect worth being investigated is to see how the QoE changes according to the screen size of the device, which is independent of its video resolution and could be the cause of further QoE unfairness.

In addition to optimal network resource distribution, live video streaming synchronisation issues have been investigated. Live streaming events are generally enjoyed together by users even if they are not physically in the same place. However, synchronisation of video playback among geographically distributed users

is fundamental to prevent users' service abandonment. When comments and reactions on social networks are left, a not synchronised video playback can be easily noticed and be detrimental to users' feelings of togetherness.

To this end, a distributed control approach to achieve synchronisation among users has been proposed. In particular, the playback time of each user has been modelled as a first-order integrator with saturated input. Then, the well-known consensus problem of simple integrators with saturated inputs has been deployed to design a distributed playback synchronisation framework. In this context, a consensus protocol acting on the draining rate of the video content playing on the user's device has been proposed. This way, we have shown that consensus is globally and asymptotically achieved. Furthermore, a leader-follower approach has been adopted with the aim of ensuring a controlled synchronisation among users in order to obtain the least possible delay with respect to the video content provider. Finally, an event-triggered control is introduced as an enhancement to the previously developed control to reduce the information exchanged among users. With this approach, clients are only required to share the value of their playback time, which is readily available to the player, only when a triggering time occurs. Simulations on different network topologies show that the proposed approach is effective and guarantees asymptotic synchronisation along with the satisfaction of playback rate saturation constraints independently of the protocols adopted. A future investigation could focus on the experimental evaluation of the proposed approach in real live streaming scenarios.

In the second and last part of this work, a methodology that proves the asymptotic stability of nonlinear systems controlled via deterministic DRL algorithms has been illustrated. The framework proposed to take a step forward in this direction consists of extracting the DRL control policy that is able to achieve the control

goal for a given system. Two DRL algorithms have been considered: DDPG and TD3. Then, a *Learner-Verifier* approach is adopted to synthesise a Lyapunov function with the help of SMT solvers. Finding a Lyapunov function is crucial in that it certifies the asymptotic stability of the system controlled with the policy extracted. This framework also provides useful insights into safety guarantees that are often necessary when it comes to real applications. Such considerations can be made through the analysis of regions of attraction obtained from those level curves of Lyapunov functions that are fully contained in the domain of the system. Experimental results obtained on two benchmark control systems show the effectiveness of the proposed approach, ensuring not only the asymptotic stability of the DRL algorithms implemented but also important considerations on safety. An interesting future research direction could be testing the framework on real control systems, but also developing an equivalent approach for stochastic DRL algorithms.

Bibliography

- [1] ISO/IEC 23009-1. Information technology—dynamic adaptive streaming over http (dash)—part 1: Media presentation description and segment formats. *Tech. Rep., ISO/IEC JTC1/SC29/WG11*, 2014.
- [2] Alessandro Abate, Daniele Ahmed, Alec Edwards, Mirco Giacobbe, and Andrea Peruffo. Fossil: a software tool for the formal synthesis of lyapunov functions and barrier certificates using neural networks. In *Proceedings of the 24th International Conference on Hybrid Systems: Computation and Control*, pages 1–11, 2021.
- [3] Alessandro Abate, Daniele Ahmed, Mirco Giacobbe, and Andrea Peruffo. Formal synthesis of lyapunov neural networks. *IEEE Control Systems Letters*, 5(3):773–778, 2020.
- [4] Florence Agboma and Antonio Liotta. Qoe-aware qos management. In *Proceedings of the 6th international conference on advances in mobile computing and multimedia*, pages 111–116, 2008.
- [5] Amir Ali Ahmadi, Miroslav Krstic, and Pablo A Parrilo. A globally asymptotically stable polynomial vector field with no polynomial lyapunov function. In *2011 50th IEEE Conference on Decision and Control and European Control Conference*, pages 7579–7580. IEEE, 2011.

- [6] Qing Ai, Yusheng Ji, Lei Zhong, Ping Wang, and Fuqiang Liu. Qoe-based cross-layer resource allocation for video streaming in high speed downlink access. In *2012 8th International Wireless Communications and Mobile Computing Conference (IWCMC)*, pages 1011–1016. IEEE, 2012.
- [7] Francesca Albertini and Domenico D’Alessandro. Asymptotic stability of continuous-time systems with saturation nonlinearities. *Systems & Control Letters*, 29(3):175–180, 1996.
- [8] Sa’di Altamimi and Shervin Shirmohammadi. Qoe-fair dash video streaming using server-side reinforcement learning. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 16(2s):1–21, 2020.
- [9] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.
- [10] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. Technical report, Stanford, 2006.
- [11] Thomas Barnett, Shruti Jain, Usha Andra, and Taru Khurana. Cisco visual networking index (vni) complete forecast update, 2017–2022. *Americas/EMEAR Cisco Knowledge Network (CKN) Presentation*, pages 1–30, 2018.
- [12] Clark Barrett and Cesare Tinelli. Satisfiability modulo theories. In *Handbook of model checking*, pages 305–343. Springer, 2018.
- [13] Abdelhak Bentaleb, Ali C Begen, and Roger Zimmermann. Snddash: Improving qoe of http adaptive streaming using software defined networking.

- In *Proceedings of the 24th ACM international conference on Multimedia*, pages 1296–1305, 2016.
- [14] Abdelhak Bentaleb, Ali C Begen, Roger Zimmermann, and Saad Harous. Sdnhas: An sdn-enabled architecture to optimize qoe in http adaptive streaming. *IEEE Transactions on Multimedia*, 19(10):2136–2151, 2017.
- [15] Abdelhak Bentaleb, Bayan Taani, Ali C Begen, Christian Timmerer, and Roger Zimmermann. A survey on bitrate adaptation schemes for streaming media over http. *IEEE Communications Surveys & Tutorials*, 21(1):562–585, 2018.
- [16] Felix Berkenkamp, Riccardo Moriconi, Angela P Schoellig, and Andreas Krause. Safe learning of regions of attraction for uncertain, nonlinear systems with gaussian processes. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 4661–4666. IEEE, 2016.
- [17] Felix Berkenkamp, Matteo Turchetta, Angela P Schoellig, and Andreas Krause. Safe model-based reinforcement learning with stability guarantees. *arXiv preprint arXiv:1705.08551*, 2017.
- [18] Dimitri Bertsekas and Robert Gallager. *Data networks*. Athena Scientific, 2021.
- [19] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.
- [20] Steven Blake, David Black, Mark Carlson, Elwyn Davies, Zheng Wang, and Walter Weiss. An architecture for differentiated services. Technical report, 1998.

- [21] Robert Braden, David Clark, and Scott Shenker. Integrated services in the internet architecture: an overview. 1994.
- [22] Tomás Brandão and Maria Paula Queluz. No-reference image quality assessment based on dct domain statistics. *Signal processing*, 88(4):822–833, 2008.
- [23] Kjell Brunnström, Sergio Ariel Beker, Katrien De Moor, Ann Dooms, Sebastian Egger, Marie-Neige Garcia, Tobias Hossfeld, Satu Jumisko-Pyykkö, Christian Keimel, Mohamed-Chaker Larabi, et al. Qualinet white paper on definitions of quality of experience. 2013.
- [24] S Canale, F Delli Priscoli, S Monaco, L Palagi, and V Suraci. A reinforcement learning approach for qos/qoe model identification. In *2015 34th Chinese Control Conference (CCC)*, pages 2019–2023. IEEE, 2015.
- [25] Silvia Canale, Federico Cimorelli, Francisco Facchinei, Raffaele Gambuti, Laura Palagi, and Vincenzo Suraci. Profiled qoe based network controller. In *2015 23rd Mediterranean Conference on Control and Automation (MED)*, pages 1085–1091. IEEE, 2015.
- [26] Gaetano Carlucci, Luca De Cicco, and Saverio Mascolo. Http over udp: an experimental investigation of quic. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, pages 609–614, 2015.
- [27] Pedro Casas and Sarah Wassermann. Improving qoe prediction in mobile video through machine learning. In *2017 8th International Conference on the Network of the Future (NOF)*, pages 1–7. IEEE, 2017.
- [28] Ray-I Chang, Yu-Chi Liu, Jan-Ming Ho, Yu-Hsien Chu, Wei-Chun Chung, and Chi-Jen Wu. Optimal scheduling of qoe-aware http adaptive streaming.

- In *TENCON 2015-2015 IEEE Region 10 Conference*, pages 1–4. IEEE, 2015.
- [29] Ya-Chien Chang, Nima Roohi, and Sicun Gao. Neural lyapunov control. *arXiv preprint arXiv:2005.00611*, 2020.
- [30] Federico Chiariotti, Stefano D’Aronco, Laura Toni, and Pascal Frossard. Online learning adaptation strategy for dash clients. In *Proceedings of the 7th International Conference on Multimedia Systems*, pages 1–12, 2016.
- [31] Yinlam Chow, Ofir Nachum, Edgar Duenez-Guzman, and Mohammad Ghavamzadeh. A lyapunov-based approach to safe reinforcement learning. *arXiv preprint arXiv:1805.07708*, 2018.
- [32] Giuseppe Cofano, Luca De Cicco, and Saverio Mascolo. A hybrid model of adaptive video streaming control systems. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 6601–6606. IEEE, 2016.
- [33] Giuseppe Cofano, Luca De Cicco, and Saverio Mascolo. Modeling and design of adaptive video streaming control systems. *IEEE Transactions on Control of Network Systems*, 5(1):548–559, 2016.
- [34] Giuseppe Cofano, Luca De Cicco, Thomas Zinner, Anh Nguyen-Ngoc, Phuoc Tran-Gia, and Saverio Mascolo. Design and experimental evaluation of network-assisted strategies for http adaptive streaming. In *Proceedings of the 7th international conference on multimedia systems*, pages 1–12, 2016.
- [35] Andrew M Colman. *A dictionary of psychology*. Oxford quick reference, 2015.

- [36] Laura Dal Col, Sophie Tarbouriech, Luca Zaccarian, and Michel Kieffer. A linear consensus approach to quality-fair video delivery. In *53rd IEEE Conference on Decision and Control*, pages 5296–5301. IEEE, 2014.
- [37] Charles Dawson, Zengyi Qin, Sicun Gao, and Chuchu Fan. Safe nonlinear control using robust neural lyapunov-barrier functions. In *Conference on Robot Learning*, pages 1724–1735. PMLR, 2022.
- [38] Luca De Cicco, Vito Caldaralo, Vittorio Palmisano, and Saverio Mascolo. ELASTIC: a client-side controller for dynamic adaptive streaming over HTTP (DASH). In *Proc. of Packet Video Workshop*, pages 1–8, 2013.
- [39] Luca De Cicco, Vito Caldaralo, Vittorio Palmisano, and Saverio Mascolo. Tapas: a tool for rapid prototyping of adaptive streaming algorithms. In *Proceedings of the 2014 Workshop on Design, Quality and Deployment of Adaptive Video Streaming*, pages 1–6, 2014.
- [40] Luca De Cicco, Gioacchino Manfredi, Saverio Mascolo, and Vittorio Palmisano. Qoe-fair resource allocation for dash video delivery systems. In *Proceedings of the 1st International Workshop on Fairness, Accountability, and Transparency in MultiMedia*, pages 33–39, 2019.
- [41] Luca De Cicco, Gioacchino Manfredi, Vittorio Palmisano, and Saverio Mascolo. A multi-commodity flow problem for fair resource allocation in multi-path video delivery networks. *IFAC-PapersOnLine*, 53(2):7386–7391, 2020.
- [42] James Deese and Roger A Kaufman. Serial effects in recall of unorganized and sequentially organized verbal material. *Journal of experimental psychology*, 54(3):180, 1957.

- [43] Nameirakpam Dhanachandra and Yambem Jina Chanu. A new approach of image segmentation method using k-means and kernel based subtractive clustering methods. *International Journal of Applied Engineering Research*, 12(20):10458–10464, 2017.
- [44] Steven Diamond and Stephen Boyd. Cvxpy: A python-embedded modeling language for convex optimization. *The Journal of Machine Learning Research*, 17(1):2909–2913, 2016.
- [45] Lei Ding, Qing-Long Han, Xiaohua Ge, and Xian-Ming Zhang. An overview of recent advances in event-triggered consensus of multiagent systems. *IEEE transactions on cybernetics*, 48(4):1110–1123, 2017.
- [46] Yan Dong and Jin-guo Xian. Finite-time event-triggered consensus for nonlinear multi-agent networks under directed network topology. *IET Control Theory & Applications*, 11(15):2458–2464, 2017.
- [47] Hermann Ebbinghaus. Memory: A contribution to experimental psychology. *Annals of neurosciences*, 20(4):155, 2013.
- [48] Alireza Erfanian, Farzad Tashtarian, Anatoliy Zabrovskiy, Christian Timmerer, and Hermann Hellwagner. Oscar: On optimizing resource utilization in live video streaming. *IEEE Transactions on Network and Service Management*, 18(1):552–569, 2021.
- [49] Jeffrey Ertman, Martin Arlitt, and Anirban Mahanti. Traffic classification using clustering algorithms. In *Proceedings of the 2006 SIGCOMM workshop on Mining network data*, pages 281–286, 2006.
- [50] ITUT Estimating end-to end. performance in ip networks for data applications, 2005.

- [51] Nagabhushan Eswara, Soumen Chakraborty, Hemanth P Sethuram, Kiran Kuchi, Abhinav Kumar, and Sumohana S Channappayya. Perceptual qoe-optimal resource allocation for adaptive video streaming. *IEEE Transactions on Broadcasting*, 66(2):346–358, 2019.
- [52] Nagabhushan Eswara, K Manasa, Avinash Kommineni, Soumen Chakraborty, Hemanth P Sethuram, Kiran Kuchi, Abhinav Kumar, and Sumohana S Channappayya. A continuous qoe evaluation framework for video streaming over http. *IEEE Transactions on Circuits and Systems for Video Technology*, 28(11):3236–3250, 2017.
- [53] Markus Fiedler, Tobias Hossfeld, and Phuoc Tran-Gia. A generic quantitative relationship between quality of experience and quality of service. *IEEE Network*, 24(2):36–41, 2010.
- [54] Alessandro Floris, Luigi Atzori, Giaime Ginesu, and Daniele D Giusto. Qoe assessment of multimedia video consumption on tablet devices. In *2012 IEEE Globecom Workshops*, pages 1329–1334. IEEE, 2012.
- [55] Timur Friedman, Ramon Caceres, and Alan Clark. Rtp control protocol extended reports (rtcp xr). Technical report, 2003.
- [56] Junjie Fu, Ying Wan, and Tingwen Huang. Event-triggered finite-time practical consensus of multiagent systems with general directed communication graphs. *International Journal of Robust and Nonlinear Control*, 30(17):7255–7277, 2020.
- [57] Junjie Fu, Guanghui Wen, Tingwen Huang, and Zhisheng Duan. Consensus of multi-agent systems with heterogeneous input saturation levels. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 66(6):1053–1057, 2018.

- [58] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, pages 1587–1596. PMLR, 2018.
- [59] Matteo Gadaleta, Federico Chiariotti, Michele Rossi, and Andrea Zanella. D-dash: A deep q-learning framework for dash video streaming. *IEEE Transactions on Cognitive Communications and Networking*, 3(4):703–718, 2017.
- [60] Sicun Gao, Jeremy Avigad, and Edmund M Clarke. δ -complete decision procedures for satisfiability over the reals. In *International Joint Conference on Automated Reasoning*, pages 286–300. Springer, 2012.
- [61] Sicun Gao, Soonho Kong, and Edmund M Clarke. dreal: An smt solver for nonlinear theories over the reals. In *International conference on automated deduction*, pages 208–214. Springer, 2013.
- [62] David Geerts, Ishan Vaishnavi, Rufael Mekuria, Oskar Van Deventer, and Pablo Cesar. Are we in sync? synchronization requirements for watching online video together. In *Proc. of the SIGCHI Conference on Human Factors in Computing Systems*, pages 311–314, 2011.
- [63] Panagiotis Georgopoulos, Yehia Elkhatib, Matthew Broadbent, Mu Mu, and Nicholas Race. Towards network-wide qoe fairness using openflow-assisted adaptive video streaming. In *Proceedings of the 2013 ACM SIGCOMM workshop on Future human-centric multimedia networking*, pages 15–20, 2013.
- [64] Deepti Ghadiyaram, Janice Pan, and Alan C Bovik. Learning a continuous-time streaming video qoe model. *IEEE Transactions on Image Processing*, 27(5):2257–2271, 2018.

- [65] Peter Giesl and Sigurdur Hafstein. Review on computational methods for lyapunov functions. *Discrete & Continuous Dynamical Systems-B*, 20(8):2291, 2015.
- [66] Bernd Girod. What’s wrong with mean-squared error? *Digital images and human vision*, pages 207–220, 1993.
- [67] Leana Golubchik, John CS Lui, and Richard R Muntz. Adaptive piggy-backing: A novel technique for data sharing in video-on-demand storage servers. *Multimedia systems*, 4(3):140–155, 1996.
- [68] Yan Gong, Fangchun Yang, Lin Huang, and Sen Su. Model-based approach to measuring quality of experience. In *2009 first international conference on emerging network intelligence*, pages 29–32. IEEE, 2009.
- [69] Abdelwahab Hamam, Mohamad Eid, Abdulmotaleb El Saddik, and Nicolas D Georganas. A quality of experience model for haptic user interfaces. In *Proceedings of the 2008 Ambi-Sys workshop on Haptic user interfaces in ambient media systems*, pages 1–6, 2008.
- [70] David W Hosmer Jr, Stanley Lemeshow, and Rodney X Sturdivant. *Applied logistic regression*, volume 398. John Wiley & Sons, 2013.
- [71] Kazuki Hosoya, Yutaka Ishibashi, Shinji Sugawara, and Kostas E Psannis. Group synchronization control considering difference of conversation roles. In *Proc. of IEEE International Symposium on Consumer Electronics (ICME)*, pages 948–952, 2009.
- [72] Tobias Hoßfeld, Sebastian Egger, Raimund Schatz, Markus Fiedler, Kathrin Masuch, and Charlott Lorentzen. Initial delay vs. interruptions:

- Between the devil and the deep blue sea. In *2012 Fourth International Workshop on Quality of Multimedia Experience*, pages 1–6. IEEE, 2012.
- [73] Tobias Hoßfeld, Michael Seufert, Matthias Hirth, Thomas Zinner, Phuoc Tran-Gia, and Raimund Schatz. Quantification of youtube qoe via crowdsourcing. In *2011 IEEE International Symposium on Multimedia*, pages 494–499. IEEE, 2011.
- [74] Tobias Hoßfeld, Lea Skorin-Kapov, Poul E Heegaard, and Martin Varela. Definition of qoe fairness in shared systems. *IEEE Communications Letters*, 21(1):184–187, 2016.
- [75] Tobias Hossfeld, Dominik Strohmeier, Alexander Raake, and Raimund Schatz. Pippi longstocking calculus for temporal stimuli pattern on youtube qoe: $1 + 1 = 3$ and $1 \cdot 4 \neq 4 \cdot 1$. In *Proceedings of the 5th Workshop on Mobile Video*, pages 37–42, 2013.
- [76] Tianchi Huang, Xin Yao, Chenglei Wu, Rui-Xiao Zhang, Zhengyuan Pang, and Lifeng Sun. Tiyuntsong: A self-play reinforcement learning approach for abr video streaming. In *2019 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1678–1683. IEEE, 2019.
- [77] Zachary Jarvis-Wloszek, Ryan Feeley, Weehong Tan, Kunpeng Sun, and Andrew Packard. Control applications of sum of squares programming. In *Positive Polynomials in Control*, pages 3–22. Springer, 2005.
- [78] Jingyan Jiang, Liang Hu, Pingting Hao, Rui Sun, Jiejun Hu, and Hongtu Li. Q-fdba: improving qoe fairness for video streaming. *Multimedia Tools and Applications*, 77(9):10787–10806, 2018.

- [79] Volker Jung, Stefan Pham, and Stefan Kaiser. A web-based media synchronization framework for mpeg-dash. In *2014 IEEE International Conference on Multimedia and Expo Workshops (ICMEW)*, pages 1–2. IEEE, 2014.
- [80] Mark Kalman, Eckehard Steinbach, and Bernd Girod. Adaptive media playout for low-delay video streaming over error-prone channels. *IEEE Transactions on Circuits and Systems for Video Technology*, 14(6):841–851, 2004.
- [81] Hassan K Khalil. *Nonlinear systems; 3rd ed.* Prentice-Hall, Upper Saddle River, NJ, 2002. The book can be consulted by contacting: PH-AID: Wallet, Lionel.
- [82] Hassan K Khalil and Jessy W Grizzle. *Nonlinear systems*, volume 3. Prentice hall Upper Saddle River, NJ, 2002.
- [83] Stas Khirman and Peter Henriksen. Relationship between quality-of-service and quality-of-experience for public internet service. In *In Proc. of the 3rd Workshop on Passive and Active Measurement*, volume 1, 2002.
- [84] Hyun Jong Kim, Dong Hyeon Lee, Jong Min Lee, Kyoung Hee Lee, Won Lyu, and Seong Gon Choi. The qoe evaluation method through the qos-qoe correlation model. In *2008 Fourth International Conference on Networked Computing and Advanced Information Management*, volume 2, pages 719–725. IEEE, 2008.
- [85] Jan Willem Kleinrouweler, Sergio Cabrero, and Pablo Cesar. Delivering stable high-quality video: An sdn architecture with dash assisting network elements. In *Proceedings of the 7th International Conference on Multimedia Systems*, pages 1–10, 2016.

- [86] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.
- [87] Praveen Kumar, Yang Yuan, Chris Yu, Nate Foster, Robert Kleinberg, Petr Lapukhov, Chiun Lin Lim, and Robert Soulé. {Semi-Oblivious} traffic engineering: The road not taken. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*, pages 157–170, 2018.
- [88] Eric Cooper Larson and Damon Michael Chandler. Most apparent distortion: full-reference image quality assessment and the role of strategy. *Journal of electronic imaging*, 19(1):011006, 2010.
- [89] Chaofeng Li and Alan C Bovik. Three-component weighted structural similarity index. In *Image quality and system performance VI*, volume 7242, pages 252–260. SPIE, 2009.
- [90] Songnan Li, Fan Zhang, Lin Ma, and King Ngi Ngan. Image quality assessment by separately evaluating detail losses and additive impairments. *IEEE Transactions on Multimedia*, 13(5):935–949, 2011.
- [91] Zhi Li, Anne Aaron, Ioannis Katsavounidis, Anush Moorthy, and Megha Manohara. Toward a practical perceptual video quality metric. *The Netflix Tech Blog*, 6(2), 2016.
- [92] Luhong Liang, Shiqi Wang, Jianhua Chen, Siwei Ma, Debin Zhao, and Wen Gao. No-reference perceptual image quality metric using gradient profiles for jpeg2000. *Signal Processing: Image Communication*, 25(7):502–516, 2010.
- [93] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous con-

- trol with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.
- [94] Jie Liu, Xiaoming Tao, and Jianhua Lu. Qoe-oriented rate adaptation for dash with enhanced deep q-learning. *IEEE Access*, 7:8454–8469, 2018.
- [95] Tsung-Jung Liu, Weisi Lin, and C-C Jay Kuo. Image quality assessment using multi-method fusion. *IEEE Transactions on image processing*, 22(5):1793–1807, 2012.
- [96] Tsung-Jung Liu, Yu-Chieh Lin, Weisi Lin, and C-C Jay Kuo. Visual quality assessment: recent developments, coding applications and future trends. *APSIPA Transactions on Signal and Information Processing*, 2, 2013.
- [97] Zhengyi Luo, Yan Huang, Xiangwen Wang, Rong Xie, and Li Song. Vmaf oriented perceptual optimization for video coding. In *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5. IEEE, 2019.
- [98] J MacQueen. Classification and analysis of multivariate observations. In *5th Berkeley Symp. Math. Statist. Probability*, pages 281–297, 1967.
- [99] Pavan C Madhusudana, Neil Birkbeck, Yilin Wang, Balu Adsumilli, and Alan C Bovik. High frame rate video quality assessment using vmaf and entropic differences. In *2021 Picture Coding Symposium (PCS)*, pages 1–5. IEEE, 2021.
- [100] Divyani Sanjay Mane and Balasaheb B Gite. Brain tumor segmentation using fuzzy c-means and k-means clustering and its area calculation and disease prediction using naive-bayes algorithm. *Brain*, 6(11):21342–21347, 2017.

- [101] Gioacchino Manfredi, Luca De Cicco, and Saverio Mascolo. Synchronizing live video streaming players via consensus. In *2021 European Control Conference (ECC)*, pages 1062–1067. IEEE, 2021.
- [102] Gioacchino Manfredi, Luca De Cicco, and Saverio Mascolo. On asymptotic stability of nonlinear systems with deep reinforcement learning controllers. In *2022 30th Mediterranean Conference on Control and Automation (MED)*, pages 306–311. IEEE, 2022.
- [103] Gioacchino Manfredi, Luca De Cicco, and Saverio Mascolo. Optimal queue-fair resource allocation in multi-path video delivery networks. *IEEE Transactions on Network and Service Management*, 2022.
- [104] Gioacchino Manfredi, Vito Andrea Racanelli, Luca De Cicco, and Saverio Mascolo. Live streaming synchronisation using event-triggered consensus control. Submitted to *IEEE Transactions on Networking*, 2022.
- [105] Hongzi Mao, Ravi Netravali, and Mohammad Alizadeh. Neural adaptive video streaming with pensieve. In *Proceedings of the conference of the ACM special interest group on data communication*, pages 197–210, 2017.
- [106] Dani Marfil, Fernando Boronat, Almanzor Sapena, and Anna Vidal. Synchronization mechanisms for multi-user and multi-device hybrid broadcast and broadband distributed scenarios. *IEEE Access*, 7:605–624, 2018.
- [107] Dylan McNamee, Charles Krasice, Kang Li, Ashvin Goel, Erik Walthinssen, David Steere, and Jonathan Walpole. Control challenges in multi-level adaptive video streaming. In *Proceedings of the 39th IEEE Conference on Decision and Control (Cat. No. 00CH37187)*, volume 3, pages 2228–2233. IEEE, 2000.

- [108] Arash Mehrjou, Mohammad Ghavamzadeh, and Bernhard Schölkopf. Neural lyapunov redesign. *arXiv preprint arXiv:2006.03947*, 2020.
- [109] Mayank Mittal, Marco Gallieri, Alessio Quaglino, Seyed Sina Mirrazavi Salehian, and Jan Koutník. Neural lyapunov model predictive control. *arXiv preprint arXiv:2002.10451*, 2020.
- [110] Jeonghoon Mo and Jean Walrand. Fair end-to-end window-based congestion control. *IEEE/ACM Transactions on networking*, 8(5):556–567, 2000.
- [111] Sabine A Moebs. A learner, is a learner, is a user, is a customer: Qos-based experience-aware adaptation. In *Proceedings of the 16th ACM international conference on Multimedia*, pages 1035–1038, 2008.
- [112] Mario Montagud, Fernando Boronat, Bernardino Roig, and Almanzor Sapena. How to perform amp? cubic adjustments for improving the qoe. *Computer Communications*, 103:61–73, 2017.
- [113] Mario Montagud, Fernando Boronat, and Hans Stokking. Design and simulation of a distributed control scheme for inter-destination media synchronization. In *2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA)*, pages 937–944. IEEE, 2013.
- [114] Mario Montagud, Fernando Boronat, Hans Stokking, and Ray van Brandenburg. Inter-destination multimedia synchronization: schemes, use cases and standardization. *Multimedia systems*, 18(6):459–482, 2012.
- [115] Sue B Moon, Jim Kurose, and Don Towsley. Packet audio playout delay adjustment: performance bounds and algorithms. *Multimedia systems*, 6(1):17–28, 1998.

- [116] Bennet B Murdock Jr. The serial position effect of free recall. *Journal of experimental psychology*, 64(5):482, 1962.
- [117] John F Nash Jr. The bargaining problem. *Econometrica: Journal of the econometric society*, pages 155–162, 1950.
- [118] Navid Noroozi, Paknoosh Karimaghaee, Fatemeh Safaei, and Hamed Javadi. Generation of lyapunov functions by neural networks. In *Proceedings of the World Congress on Engineering*, volume 2008, 2008.
- [119] Reza Olfati-Saber and Richard M Murray. Consensus problems in networks of agents with switching topology and time-delays. *IEEE Transactions on automatic control*, 49(9):1520–1533, 2004.
- [120] Brendan O’donoghue, Eric Chu, Neal Parikh, and Stephen Boyd. Conic optimization via operator splitting and homogeneous self-dual embedding. *Journal of Optimization Theory and Applications*, 169(3):1042–1068, 2016.
- [121] Antonis Papachristodoulou, James Anderson, Giorgio Valmorbida, Stephen Prajna, Pete Seiler, and Pablo Parrilo. Sostools version 3.00 sum of squares optimization toolbox for matlab. *arXiv preprint arXiv:1310.4716*, 2013.
- [122] SangHoon Park and JongWon Kim. An adaptive media playout for intra-media synchronization of networked-video applications. *Journal of Visual Communication and Image Representation*, 19(2):106–120, 2008.
- [123] Pablo A Parrilo. *Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization*. California Institute of Technology, 2000.

- [124] Dries Pauwels, Jeroen van der Hooft, Stefano Petrangeli, Tim Wauters, Danny De Vleeschauwer, and Filip De Turck. A web-based framework for fast synchronization of live video players. In *Proc. of IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pages 524–530, 2017.
- [125] Pablo Pérez, Narciso García, and Álvaro Villegas. Subjective assessment of adaptive media playout for video streaming. In *2019 Eleventh International Conference on Quality of Multimedia Experience (QoMEX)*, pages 1–6. IEEE, 2019.
- [126] Vassilios Petridis and Stavros Petridis. Construction of neural network based lyapunov functions. In *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, pages 5059–5065. IEEE, 2006.
- [127] Michal Pióro and Deep Medhi. *Routing, flow, and capacity design in communication and computer networks*. Elsevier, 2004.
- [128] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.
- [129] Benjamin Rainer and Christian Timmerer. Adaptive media playout for inter-destination media synchronization. In *Proc. International Workshop on Quality of Multimedia Experience (QoMEX)*, pages 44–45, 2013.
- [130] Benjamin Rainer and Christian Timmerer. Self-organized inter-destination multimedia synchronization for adaptive media streaming. In *Proc. of ACM Multimedia (MM)*, pages 327–336, 2014.
- [131] Sangeeta Ramakrishnan, Xiaoqing Zhu, Frank Chan, and Kashyap Kambhatla. Sdn based qoe optimization for http-based adaptive video streaming.

- In *2015 IEEE International Symposium on Multimedia (ISM)*, pages 120–123. IEEE, 2015.
- [132] Reza Rejaie, Mark Handley, and Deborah Estrin. Layered quality adaptation for internet video streaming. *IEEE Journal on Selected Areas in Communications*, 18(12):2530–2543, 2000.
- [133] Spencer M Richards, Felix Berkenkamp, and Andreas Krause. The lyapunov neural network: Adaptive stability certification for safe learning of dynamical systems. In *Conference on Robot Learning*, pages 466–476, 2018.
- [134] Marco Roccetti, Vittorio Ghini, Giovanni Pau, Paola Salomoni, and Maria Elena Bonfigli. Design and experimental evaluation of an adaptive playout delay control mechanism for packetized audio for use over the internet. *Multimedia Tools and Applications*, 14(1):23–53, 2001.
- [135] Lopamudra Roychoudhuri, Ehab Al-Shaer, and Gregory B Brewster. On the impact of loss and delay variation on internet packet audio transmission. *Computer Communications*, 29(10):1578–1589, 2006.
- [136] Gavin A Rummery and Mahesan Niranjan. *On-line Q-learning using connectionist systems*, volume 37. University of Cambridge, Department of Engineering Cambridge, UK, 1994.
- [137] Abolfazl Samani and Mea Wang. Maxstream: Sdn-based flow maximization for video streaming with qos enhancement. In *2018 IEEE 43rd Conference on Local Computer Networks (LCN)*, pages 287–290. IEEE, 2018.
- [138] Henning Schulzrinne, Stephen Casner, Ron Frederick, and Van Jacobson. Rtp: A transport protocol for real-time applications. Technical report, 2003.

- [139] Henning Schulzrinne, Anup Rao, and Robert Lanphier. Real time streaming protocol (rtsp). Technical report, 1998.
- [140] George AF Seber and Alan J Lee. *Linear regression analysis*. John Wiley & Sons, 2012.
- [141] Gursel Serpen. Empirical approximation for lyapunov functions with artificial neural nets. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pages 735–740, 2005.
- [142] Michael Seufert, Sebastian Egger, Martin Slanina, Thomas Zinner, Tobias Hoßfeld, and Phuoc Tran-Gia. A survey on quality of experience of http adaptive streaming. *IEEE Communications Surveys & Tutorials*, 17(1):469–492, 2014.
- [143] Hamid R Sheikh and Alan C Bovik. Image information and visual quality. *IEEE Transactions on image processing*, 15(2):430–444, 2006.
- [144] Hamid R Sheikh, Alan C Bovik, and Lawrence Cormack. No-reference quality assessment using natural scene statistics: Jpeg2000. *IEEE Transactions on image processing*, 14(11):1918–1927, 2005.
- [145] Iraj Sodagar. The mpeg-dash standard for multimedia streaming over the internet. *IEEE multimedia*, 18(4):62–67, 2011.
- [146] Thomas Stockhammer. Dynamic adaptive streaming over http– standards and design principles. In *Proc. of ACM Multimedia Systems (MMSys)*, pages 133–144, 2011.
- [147] Ya-Fan Su, Yi-Hsuan Yang, Meng-Ting Lu, and Homer H Chen. Smooth control of adaptive media payout for video streaming. *IEEE transactions on multimedia*, 11(7):1331–1339, 2009.

- [148] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.
- [149] Zhao Tian, Laiping Zhao, Lihai Nie, Peiqi Chen, and Shuyu Chen. Deeplive: Qoe optimization for live video streaming through deep reinforcement learning. In *2019 IEEE 25th international conference on parallel and distributed systems (ICPADS)*, pages 827–831. IEEE, 2019.
- [150] Christian Timmerer and Christopher Müller. Http streaming of mpeg media. *Streaming Day*, 2010.
- [151] Fouad A Tobagi and Joseph Pang. Starworks-a video applications server. In *Digest of Papers. Compcon Spring*, pages 4–11. IEEE, 1993.
- [152] M Oskar van Deventer, Hans Stokking, Matt Hammond, Jean Le Feuvre, and Pablo Cesar. Standards for multi-stream and multi-device media synchronization. *IEEE Communications Magazine*, 54(3):16–21, 2016.
- [153] Anthony Vannelli and Mathukumalli Vidyasagar. Maximal lyapunov functions and domains of attraction for autonomous nonlinear systems. *Automatica*, 21(1):69–80, 1985.
- [154] Vladislav Vasilev, Jérémie Leguay, Stefano Paris, Lorenzo Maggi, and Mérouane Debbah. Predicting qoe factors with machine learning. In *2018 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, 2018.
- [155] Maria Torres Vega, Vittorio Sguazzo, Decebal Constantin Mocanu, and Antonio Liotta. Accuracy of no-reference quality metrics in network-impaired video streams. In *Proceedings of the 13th International Conference on Advances in Mobile Computing and Multimedia*, pages 326–333, 2015.

- [156] Qingyong Wang, Hong-Ning Dai, Di Wu, and Hong Xiao. Data analysis on video streaming qoe over mobile networks. *EURASIP Journal on Wireless Communications and Networking*, 2018(1):1–10, 2018.
- [157] Zhou Wang and Alan C Bovik. Mean squared error: Love it or leave it? a new look at signal fidelity measures. *IEEE signal processing magazine*, 26(1):98–117, 2009.
- [158] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.
- [159] Zhou Wang and Qiang Li. Information content weighting for perceptual image quality assessment. *IEEE Transactions on image processing*, 20(5):1185–1198, 2010.
- [160] Zhou Wang, Hamid R Sheikh, and Alan C Bovik. No-reference perceptual quality assessment of jpeg compressed images. In *Proceedings. International conference on image processing*, volume 1, pages I–I. IEEE, 2002.
- [161] Zhou Wang and Eero P Simoncelli. Translation insensitive image similarity in complex wavelet domain. In *Proceedings.(ICASSP’05). IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005.*, volume 2, pages ii–573. IEEE, 2005.
- [162] Zhou Wang, Eero P Simoncelli, and Alan C Bovik. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, volume 2, pages 1398–1402. Ieee, 2003.

- [163] Christopher John Cornish Hellaby Watkins. Learning from delayed rewards. 1989.
- [164] Yijing Xie and Zongli Lin. Global optimal consensus for multi-agent systems with bounded controls. *Systems & Control Letters*, 102:104–111, 2017.
- [165] Yijing Xie and Zongli Lin. Global leader-following consensus of a group of discrete-time neutrally stable linear systems by event-triggered bounded controls. *Information Sciences*, 459:302–316, 2018.
- [166] Yuedong Xu, Eitan Altman, Rachid El-Azouzi, Majed Haddad, Salaheddine Elayoubi, and Tania Jimenez. Probabilistic analysis of buffer starvation in markovian queues. In *2012 Proceedings IEEE INFOCOM*, pages 1826–1834. IEEE, 2012.
- [167] Abid Yaqoob, Ting Bi, and Gabriel-Miro Muntean. A survey on adaptive 360 video streaming: Solutions, challenges and opportunities. *IEEE Communications Surveys & Tutorials*, 22(4):2801–2838, 2020.
- [168] Xinlei Yi, Tao Yang, Junfeng Wu, and Karl H Johansson. Distributed event-triggered control for global consensus of multi-agent systems with input saturation. *Automatica*, 100:1–9, 2019.
- [169] Xiaoqi Yin, Abhishek Jindal, Vyas Sekar, and Bruno Sinopoli. A control-theoretic approach for dynamic adaptive video streaming over http. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, pages 325–338, 2015.
- [170] Hongwei Zhang, Zhongkui Li, Zhihua Qu, and Frank L Lewis. On con-

- structing Lyapunov functions for multi-agent systems. *Automatica*, 58:39–42, 2015.
- [171] Huaipin Zhang, Dong Yue, Xiuxia Yin, Songlin Hu, and Chun xia Dou. Finite-time distributed event-triggered consensus control for multi-agent systems. *Information Sciences*, 339:132–142, 2016.
- [172] Kanghua Zheng, Huijin Fan, Lei Liu, and Zhongtao Cheng. Triggered finite-time consensus of first-order multi-agent systems with input saturation. *IET Control Theory & Applications*, 16(4):464–474, 2022.