

AUS Repository

Rate Adaptation in Dynamic Adaptive Video Streaming Over HTTP

Item Type	Thesis
Authors	Abdelhafez, Nada
Download date	2026-03-16 05:14:04
Link to Item	http://hdl.handle.net/11073/21611

RATE ADAPTATION IN DYNAMIC ADAPTIVE
VIDEO STREAMING OVER HTTP

by

Nada Abdelhafez

A Thesis Presented to the Faculty of the
American University of Sharjah College
of Engineering
in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science in
Electrical Engineering

Sharjah, United Arab Emirates

November 2021

Declaration of Authorship

I declare that this thesis is my own work and, to the best of my knowledge and belief, it does not contain material published or written by a third party, except where permission has been obtained and/or appropriately cited through full and accurate referencing.

Signed: Nada Abdelhafez

Date: December 5, 2021

The Author controls copyright for this report.
Material should not be reused without the consent of the author.
Due acknowledgment should be made where appropriate.

© Year 2021
Nada Abdelhafez
ALL RIGHTS RESERVED

Approval Signatures

We, the undersigned, approve the Master's Thesis of Nada Abdelhafez

Thesis Title: Rate Adaptation in Dynamic Adaptive Video Streaming Over HTTP

Date of Defense: 30/11/2021

Name, Title and Affiliation

Signature

Dr. Mohamed S. Hassan
Professor, Department of Electrical Engineering
Thesis Advisor

Dr. Taha Landolsi
Professor, Department of Computer Science and Engineering
Thesis Co-Advisor

Dr. Mahmoud H. Ismail Ibrahim
Professor, Department of Electrical Engineering
Thesis Committee Member

Dr. Raafat Abu-Rukba
Assistant Professor, Department of Computer Science and
Engineering
Thesis Committee Member

Dr. Oualid Hammi
Acting Head
Department of Electrical Engineering

Dr. Lotfi Romdhane
Associate Dean for Graduate Affairs and Research
College of Engineering

Dr. Sameer Al-Asheh
Interim Dean
College of Engineering

Dr. Mohamed El-Tarhuni
Vice Provost for Research and Graduate Studies

Acknowledgments

First and foremost, I thank Almighty Allah for giving me strength and showering me with all the blessings.

I would like to thank the American University of Sharjah for providing me with a graduate teaching assistantship during my graduate studies.

I would like to express my most profound gratitude to my two advisors Dr. Mohamed S. Hassan and Dr. Taha Landolsi for their support, flexibility and trust. Throughout this journey they have always offered me the guidance necessary for me to succeed as a researcher.

Also, I would like to thank my beloved husband and dear parents for their unfailing support and continuous encouragement throughout the process of researching and writing this thesis. This accomplishment would not have been possible without them.

Abstract

Video streaming stands out as the most significant traffic type consumed by mobile devices. This increased demand has been a major driver for research on bitrate adaptation algorithms. Bitrate adaptation ensures high user-perceived quality, which, in turn, correlates with higher profits for content providers and delivery systems. Dynamic Adaptive Streaming over HTTP (DASH) is a widely adopted video streaming standard utilized by service providers to provide competitive Quality of Experience (QoE). It is capable of providing seamless streaming via uncertain network conditions by switching across different video qualities and their corresponding video segment bitrates. The complexity of the video streaming environment makes it a good candidate for different learning-based approaches. Accordingly, this thesis proposes and assesses different learning-based adaptation approaches. The first proposed approach is the hybrid QoE-based algorithm, which manages the trade-off between video quality, re-buffering and quality switching events that impact the user-perceived quality. The objective is to optimize the QoE metric which is constrained by the network throughput, segment size, and buffer occupancy to continuously select the optimum bitrate levels with low complexity. The second proposed approach is the state-of-the-art DQNReg, a reinforcement learning based technique that enhances the classical deep Q-learning method. A segment-wise QoE-based reward function is established so that the learning strategy can converge towards maximizing the QoE outcome. The proposed adaptation approaches have been thoroughly evaluated using trace-based simulation for fixed and mobile networks. The first hybrid QoE-based approach performance surpasses that of the benchmark algorithm and the DQNReg algorithm outperforms other existing methods. The evaluations show that the DQNReg significantly reduces the re-buffering duration while maintaining higher video quality and relatively lower quality variations.

Keywords – Video Streaming, DASH, Bitrate Adaptation, Optimization, Reinforcement Learning, Deep Q-learning

Table of Contents

Abstract	5
List of Figures	10
List of Tables	12
List of Algorithms	13
List of Abbreviations	14
Chapter 1. Introduction	15
1.1 Motivation and Overview	15
1.2 Problem Statement	16
1.3 Objective	17
1.4 Thesis Outline	17
Chapter 2. Background	18
2.1 Internet-based Video Streaming	18
2.1.1 UDP-based video streaming	18
2.1.2 TCP-based video streaming	19
2.1.3 HTTP-based video streaming	19
2.1.4 HTTP adaptive streaming	20
2.1.5 MPEG dynamic adaptive streaming over HTTP	20
2.2 Adaptive Bitrate Streaming	21
2.2.1 Quality switching	21
2.2.2 Playback buffering	22
2.2.3 Video bitrate and quality	23

2.2.4	Video coding and segmentation	24
2.3	Reinforcement Learning	24
2.3.1	Markov decision process	25
2.3.2	Learning algorithms	27
2.3.3	Exploration strategies	29
2.3.4	Deep learning	30
2.3.5	Q-learning	30
2.3.6	Deep Q networks	32
Chapter 3.	Literature Review	34
3.1	Traditional Adaptation Bitrate Techniques	34
3.2	Reinforcement Learning Based Adaptation	36
Chapter 4.	Proposed Hybrid System Model	40
4.1	Video Streaming Model	40
4.2	Classical QoE Problem Formulation	41
4.3	Rate-Based Adaptation	44
4.4	QoE-Based Adaptation	44
4.5	Hybrid-QoE-Based Adaptation Approach	47
4.6	Simulation of the Hybrid Model	47
4.6.1	Video parameters	48
4.6.2	Network traces	48
4.7	Hybrid Model Simulation Results	49
4.7.1	Fixed channel environment	49

4.7.2	Mobile channel environment	50
4.8	Hybrid Model Results Summary	52
Chapter 5.	Proposed Reinforced Learning Model	55
5.1	DQNReg Algorithm	55
5.2	Methodology	56
5.2.1	Reward function	56
5.2.2	Video streaming environment	57
5.2.3	DQNReg rate adaptation	57
5.3	Simulation of the RL-based Model	59
5.3.1	Simulation setup	59
5.3.2	Implementation and training algorithm	59
5.3.3	Simulation results	60
5.4	Results Summary	63
Chapter 6.	Results and Discussion	66
6.1	Evaluation Metrics	66
6.2	Performance Comparison	67
6.2.1	Average QoE	67
6.2.2	Rebuffering times	67
6.2.3	Rebuffering duration	68
6.2.4	Inter-starvation length	68
6.2.5	Quality switching times	69
6.3	Analysis and Discussion	70

Chapter 7. Conclusion and Future Work	72
7.1 Conclusion	72
7.2 Future Work	72
References	73
Vita	77

List of Figures

Figure 2.1: Illustration of segment requests and bitrate adaptation in ABS.	22
Figure 2.2: The interaction between the agent and the environment in MDP.	26
Figure 4.1: Proposed hybrid system model.	42
Figure 4.2: Rate-based model implementation.	45
Figure 4.3: QoE-Based model implementation.	46
Figure 4.4: Illustration of segment requests in DASH in 5G network.	49
Figure 4.5: RB simulation in WLAN network.	50
Figure 4.6: Hybrid QoE-based simulation in WLAN network.	51
Figure 4.7: RB simulation in static 5G network.	52
Figure 4.8: Hybrid QoE-Based simulation in a static 5G network.	53
Figure 4.9: RB simulation in a moving 5G network.	53
Figure 4.10: Hybrid simulation in a moving 5G network.	54
Figure 5.1: Basic network architecture for the proposed DQNReg network.	58
Figure 5.2: DQNReg learning convergence curve.	60
Figure 5.3: DQN simulation in a WLAN network.	61
Figure 5.4: DQNReg simulation in a WLAN network.	62
Figure 5.5: DQN simulation in a 5G network with stationary users.	63
Figure 5.6: DQNReg simulation in a 5G network with stationary users.	64
Figure 5.7: DQN simulation in a 5G network with mobile users.	64
Figure 5.8: DQNReg simulation in a 5G network with mobile users.	65
Figure 6.1: Average QoE for RB, HB, DQN, DQNReg methods.	67

Figure 6.2: Rebuffering instances for RB, HB, DQN, DQNReg methods.	68
Figure 6.3: Rebuffering lengths for RB, HB, DQN, DQNReg methods.	69
Figure 6.4: Inter-starvation lengths for RB, HB, DQN, DQNReg methods.	69
Figure 6.5: QL switching instances for RB, HB, DQN, DQNReg methods.	70

List of Tables

Table 4.1: Description of symbols used in the hybrid system model.	41
--	----

List of Algorithms

2.1	Q-Learning Algorithm	31
4.1	Hybrid QoE-based Adaptation Workflow	47

List of Abbreviations

ABR Adaptive Bit Rate

ABS Adaptive Bitrate Streaming

CDN Content Delivery Network

DASH Dynamic Adaptive Streaming over HTTP

GOP Group of Pictures

HAS HTTP Adaptive Streaming

HTTP Hypertext Transfer Protocol

ISP Internet Service Provider

ML Machine Learning

MPD Media Presentation Description

QL Quality Level

QoE Quality of Experience

QoS Quality of Service

RB Rate Based

RL Reinforcement Learning

RTCP Real-Time Control Protocol

RTP Real-time Transport Protocol

RTSP Real-Time Streaming Protocol

TCP Transmission Control Protocol

UDP User Datagram Protocol

URL Uniform Resource Locator

Chapter 1: Introduction

In this chapter, a short introduction is provided to describe, in brief, dynamic adaptive streaming over HTTP and explain the need for adaptation techniques to cater to the desired user-perceived experience. Then, the problem investigated in this study is presented followed by the thesis objective. Finally, general organization of the thesis is outlined.

1.1. Motivation and Overview

With the gradual roll out of 5G mobile network, 5G wireless technology is expected to deliver high multi-Gbps peak data speeds of more than 10 Gbps, ultra low latency, increased reliability and decreased network management complexity [1]. Enhanced mobile broadband with more uniform data rates and increased efficiency empower new and improved user experiences. At the moment video streaming stands out as the most significant traffic type consumed by mobile devices accounting for an average of 60% of total traffic. This percentage is anticipated to increase to 74% by the end of 2024. Consumer behavior is shifting from low-definition and standard-definition formats (360p and 480p respectively) to high-definition (HD) video (720p and 1080p) as network capabilities improve. Not only that but viewer behaviors are expected to change more dramatically as 5G services are made available [2]. The quality of experience (QoE) perceived by the users is affected by several factors like video quality, quality switching and re-buffering duration [3] which are influenced by video streaming strategies. Considering the intricate web-based video delivery ecosystem and its various bottlenecks, adaptive bitrate (ABR) algorithms become essential to content providers to optimize video quality to enhance video delivery and customer satisfaction.

Dynamic adaptive streaming over HTTP (DASH) technology utilizes HTTP as its delivery protocol along with TCP as its transport layer protocol. DASH allows the bitrate and consequently the quality of the video to adjust according to the available resources on a sensible timescale. Relevant resources are typically the network throughput and the availability of the playback buffer [3]. The client initiates a streaming session with

the server and gets the desired video's manifest file. The media presentation description (MPD) provides required information such as uniform resource locators (URLS), bitrates, resolutions, size and availability of the video segments. Typically the video file is divided into short segments (1-10 seconds long) and encoded at several quality levels (QL). DASH enables seamless switching between quality levels for each segment based on streaming client's local information such as network quality of service (QoS) (like available throughput and network delay), playback buffer occupancy and server workload).

1.2. Problem Statement

Presently, DASH is one of the predominant and compelling technologies in internet video streaming. A significant number of techniques, algorithms and systems have been positioned around this standard which aims to offer users the best possible viewing experience. Nevertheless, the user-perceived experience still suffers from frequent stalling events and varying image quality [3,4]. The client's video player makes decisions on the next selected QL by examining different inputs such as buffer status, and network throughput. The player needs to take into account possible conflicting Quality of Experience (QoE) matters [5]:

1. Provide high bitrate video segments within the throughput limits.
2. Minimize rebuffering occurrences in which the buffer is empty and does not have content to play.
3. Reduce quality switching to smooth out the playback of the video.

The following example illustrates the possible conflicts of the aforementioned objectives: one solution to minimize rebuffering would be to select the lowest bitrate, however, it contradicts with providing high video quality which means selecting high bitrates. On the other hand, selecting high bitrates will cause rebuffering events. Likewise, lowering quality switching may also conflict with the other two objectives if the optimal solution is to minimize rebuffering event while maximizing average bitrate through frequently switching bitrates.

1.3. Objective

The ultimate objective of this work is to maximize the QoE for multimedia streaming consumers by proposing new enhanced algorithms to DASH. In order to achieve high QoE this thesis attempts to provide high bitrate video segments within the throughput limits, minimize rebuffering events and reduce quality switching to smooth out the playback of the video using different adaptation approaches in various network settings.

1.4. Thesis Outline

The thesis is organized into six chapters. Following this chapter is Chapter 2, which explores the background. It starts with a historical account of Internet-based video streaming service, then followed by an overview of the HTTP based streaming services then briefs on adaptive bitrate switching. Relevant reinforcement learning theory is also introduced in chapter 2. Chapter 3 starts by exploring different traditional adaptation methods and looks into trendy reinforcement learning techniques discussed in the literature. This is followed by the Chapter 4, which describes the methodology of the first proposed hybrid QoE-based approach. Simulation setup is described and the simulation results are discussed. Chapter 5, describes the methodology of the second proposed RL-based approach, called DQNReg. The DQNReg simulation setup is described and the simulation results are discussed. A performance evaluation and comparison of different simulated approaches is presented in Chapter 6. Finally, the thesis proposal concludes in Chapter 7 with a summary, and a discussion of planned future work.

Chapter 2: Background

This chapter will go through the required information on the work presented in this thesis report. A brief on the technology of Internet-based video streaming is introduced in Section 2.1. This is followed by the introduction of adaptive video streaming in Section 2.2, including quality switching, playback buffer, video bitrate, coding and segmentation, and adaptation controller. Next, the MPEG-Dynamic Adaptive Streaming over HTTP (DASH) is discussed in Section 2.1.5. This chapter ends with introducing the theoretical concepts of RL algorithm in Section 2.4.

2.1. Internet-based Video Streaming

Video streaming is a continuous transmission of data from a server to a client video player which plays the video/audio before receiving the entire media files. The enhancement of the quality of service of Internet Service Providers (ISP), the optimization of video compression and encoding standards and the variety of transport protocols boosted the supply and demand for video streaming services over the internet. Video streaming technologies can be classified based on the deployed transport protocol. The two main protocols are User Datagram Protocol (UDP), and Transmission Control Protocol (TCP) [6].

2.1.1. UDP-based video streaming

UDP is a connectionless transmission protocol that is utilized in various video streaming services. It does not require a handshaking dialog, hence it does not guarantee delivery, ordering, or retransmission [6]. This minimizes the delay and simplifies the transmission process. Saying that, UDP may be appropriate to delay-sensitive, real-time applications like Live video streaming services. Since UDP is susceptible to packet loss, it requires additional protocols to augment it. Real-time Transport Protocol (RTP) and The Real-Time Control Protocol (RTCP) [7] are built on UDP. RTP provides timestamps and sequence numbering to the transported stream, then RTCP gathers statistics about the state of media transfer during a streaming session. These statics are com-

municated by the client to the server which utilizes this information uses it to adapt its response.

Next, Real Time Streaming Protocol (RTSP) [7] is deployed at the application layer. It establishes and controls the streaming session, as well as monitors the QoS so that streaming applications are able to deal with delay fluctuations/jitters and packet reordering/losses [8]. Not only that, it also provides operations, such as playback, pause, and record, for audio or video streaming clients [6]. These supplementary protocols demand a specialized media server. Hence, deployment and maintenance of UDP based video streaming service are quite complex and expensive.

2.1.2. TCP-based video streaming

Unlike UDP, TCP is a connection based, end-to-end reliable protocol that offers ordered, and error-checked delivery of data packets [6]. Compared to UDP, TCP is easily deployed on existing Content Delivery Network (CDN) architecture for web services, hence, it is favorable in commercial streaming services [8].

TCP suffers from two drawbacks that makes it unfitting for multimedia streaming. Firstly, the TCP acknowledgment technique does not scale properly when the number of destinations increases for the same server. Secondly, TCP's retransmission technique may result in delays, which means it breaks strict requirements for stringent time requirements for streamed live video. Hypertext Transfer Protocol (HTTP) has been usually employed upon TCP and the video player is embodied in a web browser [6].

2.1.3. HTTP-based video streaming

HTTP is the most widely utilized protocol for content delivery over the Internet. HTTP is well established and easy to set-up. Over time, the Internet infrastructure has been adapted to efficiently handle HTTP traffic. Existing content delivery technologies developed for regular web usage can be reused by combining HTTP with a streaming technology [7].

2.1.4. HTTP adaptive streaming

HTTP adaptive streaming (HAS) is a common video quality adaptation technology in commercial streaming systems. It deploys HTTP on top of TCP as its transport technology. Leading standardization bodies in media delivery have either independently or jointly standardized it. For example, IETF, 3GPP, and the Open IPTV Forum (OIPF). These contributions led to the most commonly adopted standard MPEG Dynamic Adaptive Streaming over HTTP (MPEG-DASH) [7].

2.1.5. MPEG dynamic adaptive streaming over HTTP

MPEG dynamic adaptive streaming over HTTP (DASH) is the outcome of HAS standardization efforts. It is one of the most prominent applications of HAS that runs on top of HTTP/TCP. To facilitate the interoperability, the first specification of an HAS standard was initiated by the Third Generation Partnership Project (3GPP) and was published in TS 26.234 release 9 [8] in 2009. In collaboration with 3GPP, MPEG working group published an international standard finally published as ISO/IEC 23009 - 1:2012 in 2012 [7] called DASH, also known as MPEG-DASH. Being an international ISO standard, DASH is not owned by a single entity and thus can not be altered easily. Not only that, but also legally DASH is easily accessible for various interested companies. DASH is a flexible framework that can be implemented in versatile ways to fit multiple media delivery settings from the mobile live to video on demand on desktop computers. DASH defines the structure of the media presentation description (MPD) file and extensions for formats that allow a client player to get video streams from any server, hence, facilitating seamless playback and unification of servers and clients of diverse vendors. DASH supports any type of codec and does not dictate the segment length. Mainly, DASH specification defines two formats [8]:

- MPD which provides adequate details such as the program timing, segment availability, bitrates, and available resolutions for a DASH client to initiate a streaming session.

- The segment format which specifies the formats of the HTTP response to the DASH client's request. Segments contain video data to decode and render the included video streams.

Information on content provisioning, the delivery of the MPD and the segments, as well as adaptation algorithms are outside of the scope of the DASH standard. Ergo, DASH supports high flexibility for different use cases and delivery settings. DASH-compatible players like VLC Media Player and DASH-JS are prominent and wide spread. YouTube and Netflix streaming services are also based on MPEG-DASH [3]. The DASH Industry Forum (DASH-IF) is a working group of affluent streaming establishments that promote adoption and research in and around MPEG-DASH. DASH-IF aims for interoperability by offering publicly available guidelines, datasets, and software.

2.2. Adaptive Bitrate Streaming

In adaptive bitrate streaming (ABS), the bitrate and effectively the quality of the video stream is allowed to vary according to the available resources on a sensible timescale. Relevant resources are mainly the network throughput/bandwidth and the client playback buffer occupancy [3]. Figure 2.1 shows an example of ABS, which improves the QoE by dynamically adjusting the QLs according to the network QoS. It is further explained in the following subsections.

2.2.1. Quality switching

The client initiates a streaming session with the server and fetches the desired video's MPD file, which provides necessary information such as uniform resource locators (URLs), bitrates, resolutions, size and availability of the video segments. The client sends HTTP GET requests to fetch the video segments then plays the received video segments while the subsequent segments are being downloaded. The video file is divided into short segments (typically 2–10 seconds long) [8] which are processed individually. These video segments are encoded at multiple QL with respect to multiple video attributes such as the resolution, frame rate, and video-coding format. The QL reflect the segments' bitrate which is basically the average number of bits processed in

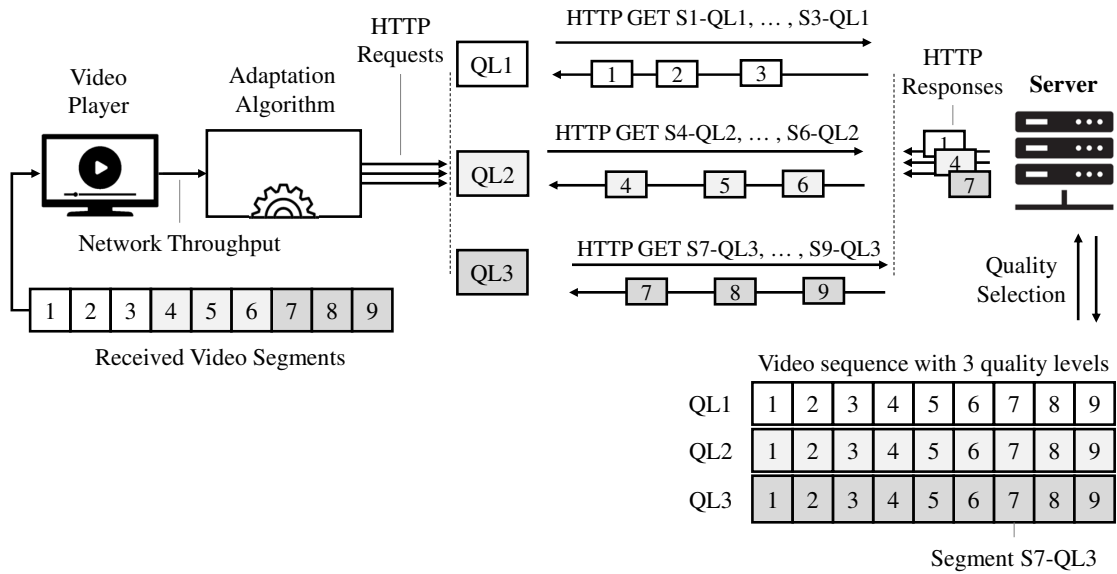


Figure 2.1: Illustration of segment requests and bitrate adaptation in ABS.

a unit of time over the entire video. The client may seamlessly switch between QLS for each segment. The streaming system performs video QL selection based on the network QoS information such as the available throughput and the network delays at the client side, in addition to the client playback buffer occupancy and the server workload information.

2.2.2. Playback buffering

The playback of the media file is played once the client receives the data. Due to both undesired packet delay and mismatch between the bitrate of the network throughput and the selected video bitrate, the arrival time of media data is occasionally unable to meet its playback deadline, preventing continuous playback of the media data. To assimilate the effect of the delay and the video/network throughput bitrate contrast, the client video player deploys a playback buffer. The buffer reserves the received data and tries to maintain its occupancy at a certain level.

The buffer size (measured in seconds) depends on two main elements; service provider settings and storage limitations on the player's device [3]. The buffer occupancy however is shaped by the video bitrate as dynamically selected by the client. The playback buffer size varies as the video is being played. It is filled with a length of video equal

to the video duration of the received segment and it is drained as the video is played back. Rebuffering events happen when the playback buffer has been depleted and the consecutive video segment does not arrive before its scheduled playback time, thus the video playback temporarily stops. This is commonly referred to as a buffer underflow or buffer starvation.

If the network throughput is unable to maintain the minimum video bitrate, the ABS system cannot avoid rebuffering, except for buffering sufficient video prior to such an insufficient throughput. Typically, internet service providers should secure satisfactory network QoS; not less than the minimum video bitrate should be sustained while streaming. However, at some times this may not be realized due network degradation events such as hardware failures, network congestion, signal degradation on wireless networks,...etc. Various studies [8] show that the QoE is affected mostly by the frequency and the duration of the rebuffering events. Thus, a proficient ABS system should maintain the buffer occupancy should be at a high level to reduce the effect of network degradation and should be able to minimize rebuffering events.

2.2.3. Video bitrate and quality

As previously touched on, the video files are encoded at multiple quality levels, which are represented by the nominal video bitrate. Since the video bitrate is the average rate at which a video is processed—typically measured in kilobits per second (kbps) and/or megabits per second (Mbps)—, the nominal bitrate is the mean video bitrate over the entire video. Being a subjective term, the quality of the video is shaped by multiple factors such as firstly, the video compression/coding technique. Video compression refers to reducing the amount of data in a video by decreasing and removing unwanted data for efficient video storage [8]. Secondly, the video resolution, which refers to the number of distinct pixels that could be displayed in each dimension (width \times height), for example 1280×720 (720p) or 1920×1080 (1080p, Full HD). Finally, the frame rate, which is defined by the number of successive images called frames that are displayed per second on the screen, that is frames-per-second, for example: 24 fps, 30 fps, and 60 fps. Typically the higher video bitrate the better the image quality in the video.

Other factors may also affect the image quality. For instance, at the same video bitrate, advanced codecs will result in improved image quality compared to old ones.

2.2.4. Video coding and segmentation

Segmentation is crucial in video streaming. A video stream is divided into segments with a fixed length (measured in seconds). For smooth and uninterrupted playback, it is required that each segment is decoded separately. The streaming format of the video must attain independent segment decoding. Advanced video encoding deploys inter-frame compression, in which one or more adjacent frames are utilized to reconstruct a current frame where only the differences are transmitted due to the temporal redundancy. Ergo, it is essential to guarantee that there is an Intra-coded frame (I-frame) at the start of each segment, so that clients can smoothly play back segments and transition across quality levels between various segments. The client can start the decoding process immediately upon receiving the segment, (independently) as I-frames do not reference any other frames or segments. Aligning the group-of-picture (GOP) size of the encoding to the number of frames in a segment can attain fixed I-frame positions. For instance, for a video with a frame rate of 24 fps, a segment with two second video duration may have a GOP size of 24 frames or 48 frames [8]. As a result, shorter segment durations have a lower compression ratio as I-frames have more bits, and hence the final video quality is inferior to the one of longer segments encoded at the same bitrate. Nevertheless, short segments can attain faster adaptability. Saying that, it is important that this trade-off is balanced in content generation for adaptive streaming. Generally, the segment duration is between between 2 seconds and 10 seconds [7].

2.3. Reinforcement Learning

Reinforcement learning (RL) is a machine learning paradigm that aims to maximize a reward signal by learning how to map certain states to specific actions. A key component in RL that is responsible for learning and making decisions is the learning agent. The learning agent is not told what to do and does not have the complete environment

model. By trial and error, the learning agent discovers which actions yield the highest reward and memorizes the consequences of its actions.

Unlike supervised machine learning, in RL, the learning agent is not fed with a dataset of state-action pairs, but only partial feedback is given to the learning agent about its actions. Not only that but the actions taken by the agent influence the future state of the environment which may lead to long term effects. The relationship between the environment and the learning agent must be mathematically modeled to solve the problem. Typically the model should represent the notions of states and transition between states over time. Additionally, it is important to carefully design an algorithm that sets the learning agent's strategy or policy and then optimizes the strategy to obtain the highest rewards. Finally, the RL problem is said to be solved once the learning agent learns the optimal policy to pursue in all states. RL problems are commonly formulated as Markov decision processes (MDPs). This formulation is explained in the following subsection.

2.3.1. Markov decision process

MDP is a mathematical tool for sequential decision making under uncertainty [9]. Actions taken not only impact subsequent situations but also instant rewards. MDP is a Markov process in a way that only the current state and action impact the transition probability from the current state to a subsequent state, no matter what all past and future states or actions taken. An MDP is defined as a tuple $(S, A, R, P_a, R, \gamma)$ where:

- S is the finite set of states of the environment.
- A is the finite set of actions available to the agent in state $s \in S$.
- P_a is the transition probability from state s_t to state s_{t+1} when the agent chooses action $a \in A$.
- R is the instant reward that the agent receives when it selects action $a \in A$ and the state transitions from s_t to s_{t+1} .
- γ is an exponential discount factor on future rewards. It must satisfy $\gamma \in [0, 1]$ to prevent divergence.
- The long-term reward is given by: $\sum_{\tau=t}^{\infty} r_{\tau} \gamma^{\tau-t}$.

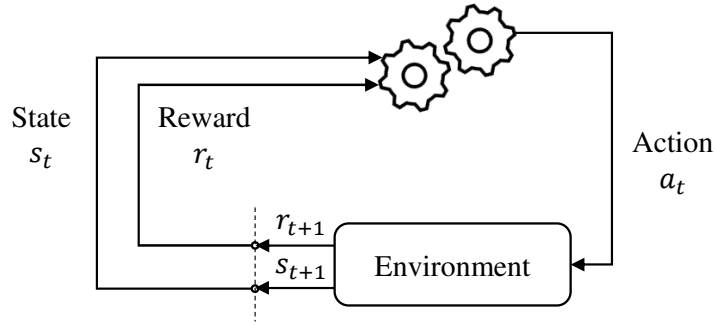


Figure 2.2: The interaction between the agent and the environment in MDP.

Figure 2.2 illustrates the interaction between the learning agent and the environment in an MDP. The agent does not have the full knowledge of the environment's model or the impact of all of its actions. Therefore, it needs to learn how to maximize the rewards and this is done by exploring various actions at different states.

A policy $\pi(s_t, a_t) : S \times A \rightarrow [0, 1]$ is the function that links every state to an action, it depends on the long term rewards acquired by the agent's learning experience from past experiments. The policy's value function $V_\pi(s_t)$ is calculated to evaluate the agent's decision of being in a certain state. The value function also represents the expected total reward achieved through the agent's actions. Finding a solution to the MDP means finding an optimal policy $\pi^*(s_t)$ that maximizes the long-term reward in each state, which is the objective of the learning agent. The learning agent's reward depends on the future evolution of the MDP, so a selection that maximizes immediate reward but puts the agent in a low-yielding state may not be the optimum action.

Other approaches can be used to solve MDPs, such as dynamic programming techniques. Nevertheless, when the state-space is too large it becomes unmanageable and hence RL algorithms can be utilized by virtue of sampling and function approximation. Sampling represent the dynamics of the environment and allows the agent to deal with varying or complex environment dynamics. Where as function approximation represents the value functions and facilitates dealing with high dimensional state-spaces and actions. RL algorithms do not provide an exact solution, but they do not need complete knowledge of the MDP to converge and can work for very large MDPs.

2.3.2. Learning algorithms

Aforementioned in last section, the objective of the learning agent is to learn the optimal policy $\pi^*(s_t)$. However, this optimal policy doesn't have to be unique [10]. Practically, the learning agent begins with a random policy, then keeps improving it by observing rewards till it converges to the optimal policy. The value functions are calculated to compare and evaluate the performance of various policies. The principal three value functions are illustrated in the following:

- Action-value function or Q Value $Q(s_t, a_t)$: This is a measure of the total expected reward. It shows the learning agent's belief of how good it is to take action a at state s at time t following some policy π in the future.

$$Q(s_t, a_t) = \mathbb{E}_\pi[R_t | S = s_t, A = a_t]. \quad (2.1)$$

As shown in equation (2.1), $Q(s_t, a_t)$ is the expected reward following a specific policy π given that the learning agent is in state s_t and have taken action a_t .

- State-value function $V_\pi(s_t)$: This is a measure of the total expected reward when the agent is in a certain state at a particular time. It reflects the agent's learned experience of how good to be at state s at time t if it follows some policy π . It is represented by:

$$V_\pi(s_t) = \sum_{a \in A} \pi(a_t | s_t) Q(s_t, a_t). \quad (2.2)$$

Equation (2.2) shows that the value function is the total reward, since $Q(s_t, a_t)$ measures the expected reward, while following a policy $\pi(a_t, s_t)$.

While this looks similar to the action-value function, there is a significant difference to be clarified. The reward of the value function is just from being in state s before any action is taken. However, the expected reward of the action-value /Q value is after a certain action is taken. This difference also yields the advantage function.

- Advantage function, $A_\pi(s_t, a_t)$: This is a measure of how much is a specific action a is good in a particular state s . It reflects the advantage of picking a certain action from a state compared to picking an action in random. Meaning $A_\pi(s_t, a_t)$

reflects relative (rather than absolute) state action values which is easier to learn since it only needs to learn one action that yields better rewards. The advantage function is given by:

$$A_{\pi}(s_t, a_t) = Q(s_t, a_t) - V_{\pi}(s_t) \approx (r_t + \gamma V_{\pi}(s_{t+1})) - V_{\pi}(s_t). \quad (2.3)$$

Equation (2.3) describes the advantage function as the difference between the Q value, $Q(s_t, a_t)$, and the value function $V_{\pi}(s_t)$. It is approximated by the instantaneous reward r_t in addition to the discounted value function at state s_{t+1} $V_{\pi}(s_{t+1})$ minus the value function at state s_t which is $V_{\pi}(s_t)$.

Typically, value functions are stored as tables, where the states and actions are represented as rows and columns, respectively. In the case of large state-action spaces, where the states or actions have continuous values, the size of value function becomes unmanageable. Henceforth, function approximation techniques are utilized. For instance, the weights of the neural networks (NN) parametrize the value function $V_{\pi}(s_t)$ to obtain an approximated value $V_{\pi}(s_t, w)$.

Policy evaluation and policy improvement are two steps that aid in learning the optimal policy. The former approximates the value function while the latter improves the policy. These two steps can be implemented through various methods. These methods are grouped into three main categories of algorithms:

- Value-based methods learn the state or state-action value by choosing the best action. Utilize the Bellman equation and keeps approximating the value function. Hence, improving the value function improves the derived policy. Q-learning and SARSA [10] algorithms are value-based.
- Policy-based methods learn the stochastic policy function that maps state to action. Utilize learned experience and/or samples to enhance the policy without considering value functions. They are typically used when the a stochastic policy is required or if action space is continuous.
- Actor-critic methods merge the benefits of value-based and policy-based algorithms. The critic utilizes an estimation architecture to learn a value function which

then updates the actor's policy parameters. On the other hand, the actor aims to optimize its parameters and policy to obtain enhanced value function. Once the optimal policy is found, the actor network employs the algorithm.

2.3.3. Exploration strategies

During the process of learning the optimal policy, the learning agent tries to explore the state-action space to learn more about the environment. However, pure exploration may increase the learning time and the computing resources without helping the learning agent achieve its target.

It is important that the agent tries to find the right combination of exploration and exploitation. This allows the agent to utilize its current learned knowledge to achieve its target while minimizing the learning time and maximizing the learning agent's performance.

An important parameter of exploration is the exploration rate which tries to allocate a certain threshold for exploration so as to avoid unnecessary knowledge. At the same time it allows for the agent to exploit from its experiences to faster achieve its goals.

To manage a reasonable combination of exploration and exploitation, three behavior policies have been proposed in the literature, namely, ϵ -greedy [11], Softmax [12], and VDBE-Softmax [13]. These strategies are used to find suitable actions, and they have different balances of exploration versus exploitation. They can be defined as follows:

- **ϵ -greedy:** It is a common method to have a reasonable balance between exploration and exploitation. The learning agent selects actions randomly. Non-greedy actions (explore) are selected with probability ϵ , while greedy (exploit) actions are selected with probability $1 - \epsilon$. Through managing the value of ϵ , a certain level of exploration can be guaranteed. It is defined as:

$$\pi(s) = \begin{cases} \text{Random action } a \in A & \text{if } \zeta \leq \epsilon, \\ \arg \max Q_{\pi}(s, a) & \text{otherwise,} \end{cases} \quad (2.4)$$

where $0 \leq \zeta \leq 1$ is a uniform random number, $0 \leq \epsilon \leq 1$ and $Q_\pi(s, a)$ the Q-value when the action a is taken (Q-value is a mapping between the states and actions following a policy π , i.e., the state-action value which is the expected discounted revenue for following π).

- **Softmax:** The learning agent selects an action based on probabilistic function, in particular, a Boltzmann distribution is used to rank the learned Q-values. Actions with high Q-values will be selected with high probabilities, so the exploration is directed towards achieving the desired goal.
- **VDBE-Softmax:** It combines a policy called value-difference-based exploration with Softmax. This allows Softmax to carry out the exploration but with a state-dependent exploration probability $\epsilon(s)$, unlike the tuning in Softmax. This makes the learning agent more exploitative in uncertain conditions. This is reflected by the fluctuating Q values during the learning process.

2.3.4. Deep learning

Deep-learning algorithms are representation learning methods that utilize several consecutive layers of artificial neurons, in which each layer consists of simple modules that reshape the input data into an abstract form then uses it as an input for the successive layer.

Deep Q Network (DQN) algorithm is a revolutionary combination of deep neural networks and RL that reignited the interest in deep RL. In the following, Q-learning is explained, then DQN along with the innovative experience replay technique are discussed.

2.3.5. Q-learning

Q-learning is based on the Q-function. As stated earlier, the Q-function or action-value function of a policy measures the expected reward from a state by taking an action first then following a policy in the future. The optimal action-value function follows the prominent identity the *Bellman equation*. The Bellman equation states that if the optimal Q-value of the next state is known for all possible actions, then the optimal strategy

is to select the action that maximizes the expected reward. Q-learning is a powerful technique to solve decision making problem in RL. It is an off-policy temporal difference (TD) that can deal with delayed rewards. The algorithm maintains a table of the expected long-term rewards. This table is known as the Q-table, it sets the Q-values for state-action pairs. The Q-table rows are all states and each column is related to a possible action from the set of actions. The Q-values are initialized with zeros or sometimes with high values to motivate exploration till reasonable values are approximated. The learning agent improves its estimates by adjusting them to its experience. It updates the Q-table and ensures the Q-values converge to the optimal long-term expected reward. For every $Q_t(s, a)$ in Q-table, the update is given by:

$$Q_t(s_t, a_t) = Q_{t-1}(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_a Q_t(s_{t+1}, a) - Q_t(s_t, a_t)], \quad (2.5)$$

where α is the learning rate ($0 \leq \alpha \leq 1$) that sets the aggressiveness of the update and is typically decremented with time as the agent approaches convergence and r_{t+1} is the immediate reward. The long-term rewards are estimated using the next state's Q-values, typically with a certain behavior policy. The learned Q-value estimate the optimal Q-value regardless of the policy followed. The Q-learning method is illustrated in Algorithm 2.1. Q-learning can use different exploration policies to select the future

Algorithm 2.1 Q-Learning Algorithm

- 1: $Q_t(S, A) \rightarrow$ Initialize (0), for $s \in S, a \in A$
- 2: **for** each episode **do**
- 3: Start state $s_t \rightarrow$ Initialize (0)
- 4: **repeat**
- 5: $a \rightarrow$ Policy (s_t)
- 6: Take action a , Observe (r_{t+1}, s_{t+1})
- 7: Update the Q-value:

$$Q_t(s_t, a_t) = Q_{t-1}(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_a Q_t(s_{t+1}, a) - Q_t(s_t, a_t)]$$

- 8: Update $s_t \rightarrow s_{t+1}$
 - 9: **until** s is in terminal state
 - 10: **end for**
-

action based on the estimated Q-values. The most common ones are the ϵ -greedy policy and Softmax.

The Q-learning approach is quite effective, however it has some limitations: the algorithm converges to the optimal policy if its parameters are selected accurately [14]. Not only that but the convergence speed is a matter of concern in complex environments. Not to forget the curse of dimensionality, where the number of states in the MDP can be quite large for most problems, forcing classical Q-learning technique to take significant numbers of training samples to converge and obtain a fair trade-off between accuracy and adaptability. In this thesis the objective is to create RL algorithms that converge rapidly, generalize well and approximate the optimal policies that would be derived by solving the MDP. Deep Q-learning can fulfill this as it delivers natural and powerful means to generalize the knowledge attained during certain transitions and re-utilize it for future states.

2.3.6. Deep Q networks

DQN was introduced by Mnih et al. [15] to solve various Atari games and beat human experts. The algorithm was developed by enhancing the traditional Q-learning algorithm with deep NN and the state-of-the-art technique called experience replay.

The main difference between DQN and Q-learning is the implementation of the Q-table. DQN utilizes a neural network to map input states to action/Q-value pairs. Typically, a neural network is made up of several basic units called neurons that are connected in various ways based on a certain architecture. The neurons in the input layer get activated by the environment state variables, while neurons in the other layers get activated through weighted connections from neurons in the preceding layers. The architecture and the activation functions (how the weighted input is altered by each neuron) define the connection weights and biases, which in turn are included in a single vector w . A *mini-batch method* that updates the weights after accumulating gradient information over a small batch of samples is also proposed in [15]. This method yielded smoother sample gradients in contrast to updating the weights after each action [10].

Different network architectures can be used such as feed-forward neural networks (FFNN), convolutional neural networks (CNN) and recurrent neural networks (RNN). The DQN loss function is defined below [16]:

$$L_{DQN} = (Q(s_t, a_t) - (r_t + \gamma * \max Q_{targ}(s_{t+1}, a)))^2, \quad (2.6)$$

The DQN architecture has two deep neural networks, the Q network and the target network. The first network updates the weight vector w_t at every time step t , which is then used to make up the Q-value map $Q(s_t, a_t)$. The second neural network, known as the *target network*, is meant to increase the stability of the learning agent. Equation (2.6) shows the DQN loss function. In the learning process, DQN minimizes the squared error between the predicted value $Q(s_t, a_t)$ and the target Q value which is computed the adding the reward r_t to the discounted Q value and maximized $Q_{targ}(s_{t+1}, a)$ provided by the target network.

Experience replay is the act of storing past experiences that the RL algorithm can learn from. Instead of using a single recent experience, it allows utilizing a random subset of the learned experiences to update the Q-network. It can be used with off-policy methods to update the RL algorithm's parameters. DQN uses experience replay to learn in small batches. Not only it avoids bad correlations it also allows learning more from individual experiences multiple times. Overall it utilizes the stored past experiences better.

Chapter 3: Literature Review

Adaptation methods and ABR algorithms are still vital research areas regardless of general deployment [17]. This is owed to the continuously updated internet video delivery ecosystem. This section illustrates various classic adaptation approaches. The existing approaches are classified into two main categories; Traditional adaptation techniques and RL based adaptation methods. In most of traditional approaches solutions are customized for different scenarios. On the other hand, RL based methods learn to adapt in unforeseen environments. Typically, network QoS and buffer occupancy information are used in these methods.

3.1. Traditional Adaptation Bitrate Techniques

Several ABR approaches have been proposed to tackle the challenges associated with video streaming. Generally, bitrate adaptation methods utilize some information about the resources (e.g., typically available network throughput and/or buffer occupancy) as inputs, and then based on a specific algorithm the bitrate is selected for the next segments. Sometimes the information of the available throughput is unknown, hence, an estimation technique is typically adopted to predict the throughput for the delivery of the next segments. The simple way is to use a smoothing method such as averaging based on measured throughput history. Since most information is the local information of the client and the network QoS involves client's network states.

Most of the time bitrate adaptation take place at the client side. The earliest proposed solutions are mainly divided into three main categories: rate-based (RB), buffer-based (BB) and rate and buffer based (RBB) and [7, 18]. RB algorithms predict the network bandwidth from previous video segment downloads, then request segments at the highest bitrate that can be supported by the network. These approaches are impacted by the network throughput prediction bias. Different ways are suggested to improve throughput predictions but accurate estimates remain a challenge.

On the other hand, BB take into account the client's playback buffer availability when selecting the bitrates for upcoming video segments. The main aim of these methods

is to maintain the buffer occupancy at a preset level that balance rebuffering and video quality. Finally, RBB methods combine both throughput predictions and clients playback buffer availability information to select bitrates of subsequent segments. A classical RBB approach model predictive control (MPC) proposed by [3] introduced an improved throughput predictor and utilized playback buffer information to maximize the QoE over future video segments. However, MPC depends on throughput predictions which are still not always accurate. Not only that, but it is still challenging to tune several fixed heuristics to perform adequately every time in a varying environment.

Based on the data extracted to obtain strategies, the heuristic based techniques are further sub categorized into two [17]. Firstly, throughput-based approach, that only use the network throughput is used to make the decision for future segment bitrate [18]. Secondly, the buffer-based approach [19], that utilizes the available buffer length updates.

The authors in [20] presented a prediction model to accurately approximate the trend of buffer level variation in the client side. The quality switching in the adaptation is reduced based on the approximation. It can be inferred from the results that a steady video quality is achieved when buffer underflows are reduced. Nonetheless, the main drawback of the heuristic based techniques is that they cannot be generalized since they are deterministically customized to particular network conditions. It is hard to use heuristics to design an algorithm that is predictable and mathematically describable. Thus, attempts to design an adaptation logic that is not only performing well but also based on descriptive and predictable models that offer more dynamic solutions.

Control theory is used to model dynamical systems that are stable, accurate and settle quickly into a steady state [3]. The work of Cicco et al. [21] proposes an adaptation logic based on feedback control. The quality adaptation controller takes a target buffer as an input and returns the video rate of the segment to be downloaded. The goal of the controller is to ensure the buffer is always maintained at the target level. It achieves this by calculating the error between the target buffer and the measured buffer level. The error is then passed to the Proportional Integral (PI) controller that outputs a video rate that matches the estimated available bandwidth.

Optimization-based approaches are widely used for bitrate adaptation in DASH. Essentially, the optimization problem is solved based on the prediction of buffer dynamics and network throughput. However, stochastic segment size deviates observably the buffer occupancy from the expected value, making the evolution hard to predict. In order to get rid of this effect and improve the prediction accuracy for buffer occupancy, the work in [22] proposes an algorithm based on Markov decision process (MDP) with incorporating segment size information so that only the network capacity variation need to be considered in the decision-making process.

A buffer-based adaptation approach that does not rely on throughput prediction methods is presented by Huang et al. [23]. The algorithm solves optimization problem based on the buffer length reservation. It downloads the lowest bitrate and tries to maintain the buffer occupancy within a certain threshold. Another adaptation logic based on optimization techniques is presented in [24] which introduces a novel probing based network measurement technique to advance the video quality selection. It also presents a QoE-aware DASH system (QDASH) with mixed-integer linear programming.

Sobhani et al. in [4] tried a different approach. They use an optimization mechanism with two objective functions; the first function maximizes the overall average QoE among DASH clients while the second function minimizes the negative impact of temporal video quality changes, that is the up and down switching between different representation during playback.

At the network provider side, there is a need to arbitrate the allocation of network resources among the competing clients. Hence, according to Bouten et al. [25] the support for coordinated management, and global optimization is essential. They employ an integer linear programming (ILP) model to manage policies to either maximize the QoE of all users or minimize the penalties incurred for violating the subscriptions contract.

3.2. Reinforcement Learning Based Adaptation

Existing classic ABR algorithms mostly rely on fixed heuristics that have been excessively tuned to assumptions about the video streaming environment. Also, some of

these algorithms tend to optimize a certain QoE metric. Thus, fixed ABR methods tend to fail to generalize to new or unforeseen conditions. They would require manual tuning to be able to perform well in new environments.

To overcome these issues, learning-based approach for generating ABR algorithms becomes the trend. With machine learning (ML) techniques a client can learn to adapt its video quality to the changing context without the need for any human intervention.

Chien et al. in [26] use a decision tree-based random forest classification to map network related features onto the video rate. The scheme trains the classification model using a dataset. The classifier is then used to predict the current request or any future video request. Unlike all previous works that implement fixed heuristics, RL in adaptive video streaming was used in [27]. In this case, an agent gradually improves its bitrate adaptation policy by interacting with the video streaming environment.

The promising results lead to more efforts in applying RL in more realistic scenarios. RL allows an agent to discover the right action to take, within a particular context, based on feedback from its environment. To do this, an adaptation module interacts with its environment by sensing factors that are expected to influence its decision. For example, Pensieve, a system that generates ABR algorithms using RL, is introduced in [28]. Pensieve trains a neural network model that selects bitrates for future video segments based on observations collected by client video players. Pensieve does not rely on pre-programmed models or assumptions about the environment, it learns to make ABR decisions solely through observations of the resulting performance of past decisions. Furthermore, its asynchronous parallel learning framework was also compatible with the scenarios of multiple users watching DASH video at the same time, thus suitable for real-time online learning.

The work in [29] also incorporated a reinforcement learning method with the addition of Q-Learning. The action is set as the segment request with a particular bitrate and the reward is defined as the QoE approximation. The study maximizes the QoE through adjusting the adaptation behavior as per the existing network conditions.

Another ML-based approach [30] proposes to combine the nearest neighbor (KNN) with Q-learning algorithm. In the bitrate adaptive scenario, the KNN-Q learning can achieve higher QoE and faster convergence speed than Q-learning algorithm alone.

The work in [31] introduces Comyco which trains the policy via imitating expert trajectories given by the instant solver. It attempts to pick the segment with higher perceptual video qualities rather than video bitrates by constructing Comyco's neural network architecture, video datasets and QoE metrics with video quality features.

An online learning adaptation strategy for DASH based on MDP optimization was proposed in [32] based on an MDP optimization. The authors introduced a penalty function into the reward function to penalize the system for re-buffering events as well as moving away from a safe buffer level.

The authors in [33] proposed D-DASH, a framework based on deep learning and RL methods to optimize the QoE of DASH. They suggested combining feed forward and recurrent deep neural networks with improved strategies. They evaluated their proposed framework on real and simulated network traces. D-DASH framework yielded high QoE values at a fast convergence rate.

J. Liu et al. [34] proposed a framework based on an improved deep Q-learning approach and called it redirecting enhanced deep Q-learning (RDQ). They designed a segment-wise QoE model and utilized it as the reward function in the RL model. The learning strategy converges towards maximizing the QoE score. They also applied various effective enhancements of deep Q-learning to the proposed RDQ's learning agent's neural network architecture. They evaluated the RDQ's learning agent's performance using trace based simulations on LTE network trace. Their simulations showed that RDQ achieves higher average reward and faster convergence compared to other learning based methods.

In this thesis, I propose to assess different adaptation algorithms: hybrid and learning-based ones. The first proposed approach is the hybrid QoE-based algorithm, which manages the trade-off between video quality, re-buffering and quality switching events that impact the user-perceived quality. The second proposed approach is the state-of-the-art DQNReg, an RL-based technique that enhances the classical deep Q-learning

method that has fast convergence during the training and better QoE performance during the test.

Chapter 4: Proposed Hybrid System Model

This chapter describes the system model, formulates the classical QoE problem and describes the first proposed adaptation approach implemented in this study. The approach is a hybrid of the classical rate-based technique and a proposed QoE-based adaptation technique.

4.1. Video Streaming Model

The video streaming system model as illustrated in Figure 4.1 is described in this section. A video is modeled as a set of N_s consecutive segments in one video sequence. Each segment lasts for T_s seconds encoded at different R^v bitrates. The client video player requests a segment at bitrate R_i^v for the i th segment. The selected bitrate R_i^v is mapped to different quality levels $\{Q_1, Q_2, \dots, Q_N\}$ based on playing device specifications and the available video content. Suppose Q_j is requested for i -th segment, and its corresponding bitrate is R_i^v . The higher the bitrate selected, the higher video quality delivered to the viewer. Let L be the size of segment i encoded at bitrate R_i^v . As per [3] for a constant bitrate (CBR) case, $L = T_s \times R_i^v$, unlike the variable bitrate (VBR) case where L relationship can change across segments. The video segments are requested, received and downloaded into a buffer $T_i^B \in [0, T_{max}^B]$ by the client. The buffer size (in seconds) depends on two main elements; service provider settings and storage available on the player's device. Typically the buffer may contain tens of seconds of video segments.

Given the average network throughput be R_i^n during downloading the i -th segment, the needed download time will be L/R_i^n . As inferred from the calculation, the download duration depends on the size of the segment with bitrate R_i^v and the experienced average download speed. The buffer size varies as the video is being played. It increases by T_s as a segment is downloaded and decreases as the video is played back. Rebuffering events happen when the playback buffer of the client has been depleted and the consecutive video segment does not arrive before its scheduled playback time, thus the video playback temporarily stops.

Assuming the length of the buffer is T_i^B before segment download, the length of buffer occupancy changes with the next segment as follows:

$$T_{i+1}^B = T_i^B - \frac{L}{R_i^n} + T_s. \quad (4.1)$$

The buffer occupancy is continuously changing with the process of video segment playback and video segment reception. Video segment playback is represented in equation (4.1) by subtracting the total time it takes a segment to get downloaded (L/R_i^n). While video segment reception is represented by adding the T_s to the buffer update equation. Therefore, if sufficient buffer content is maintained before loading ($T_i^B > L/R_i^n$) re-buffering events will not take place. Table 4.1 summarizes the discussed notation.

Table 4.1: Description of symbols used in the hybrid system model.

Symbol	Definition
N_s	Total number of video segments
i	Index for video segments
N_r	Total number of available bitrate levels for the video
N_q	Total number of available quality levels
$Q(R_i^v)$	Quality level for the i -th segment
R_i^v	Bitrate of i -th segment (kbps)
T_s	Time duration of any video segment
T_{max}^B	Client buffer maximum occupancy (s)
R_i^n	Network throughput during download of i -th segment (kbps)
T_i^D	Download Time duration of i -th video segment
L	Length of any video segment in bits

4.2. Classical QoE Problem Formulation

QoE is an active area of research as it is subjective to perceived quality. It considers how users view the general quality of a service [35]. Hence improving the QoE is the main objective of the proposed study. While users may differ in their distinct QoE functions,

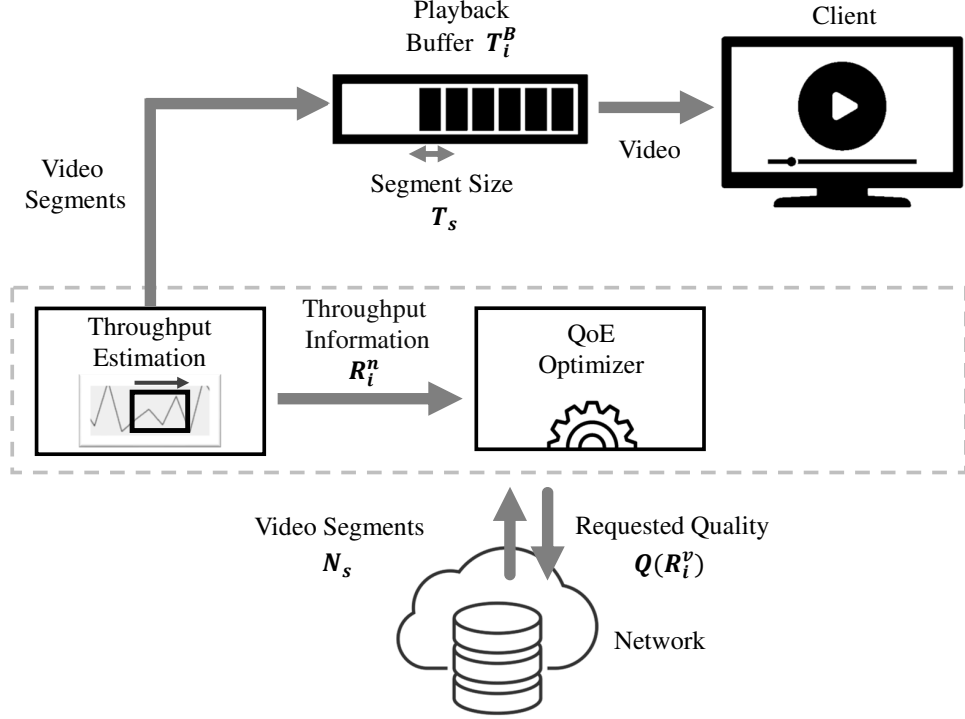


Figure 4.1: Proposed hybrid system model.

video QoE, the key elements as indicated in [3, 4, 28] and [31, 36] are enumerated as follows:

1. Required Average Video Quality (VQ), which is a summation of all the requested video segment qualities $Q(R_i)$ over the total available video segments N_s . It indicates how overall quality requested by the adaptation bitrate algorithm. It is computed by:

$$VQ = \frac{1}{N_s} \sum_{i=1}^{N_s} Q(R_i^v). \quad (4.2)$$

2. Rebuffering Duration (RD), which is a summation of all the times the buffer starves. Starvation happens when the buffer is empty, this takes place when the segment download time, $T_s R_i^v / R_i^n$, is greater than available buffer playback time. RD is calculated by:

$$RD = \sum_{i=1}^{N_s} \frac{T_s R_i^v}{R_i^n} - T_i^B. \quad (4.3)$$

3. Average Quality Switching (QW) is a measure of the difference between the requested quality of video segment $i + 1$ and the quality in the received video seg-

ment i . The average of these quality differences across the video segments indicates how much quality fluctuates throughout the playback of the all the video segments N_s . The average quality switching is given by:

$$QW = \frac{1}{N_s - 1} \sum_{i=1}^{N_s-1} |Q(R_{i+1}^v) - Q(R_i^v)|. \quad (4.4)$$

Typically adaptation algorithms need to maximize the average video quality, minimize rebuffering occurrences in which the buffer is empty and does not have content to play and minimize the quality switching to smooth out the playback of the video.

The following example illustrates the possible conflicts of the objectives: One solution to minimize rebuffering would be to select the low bitrates, however, it contradicts with providing high video quality which means selecting high bitrates. On the other hand, selecting high bitrates will cause rebuffering events. Likewise, lowering quality switching may also conflict with the other two objectives if the optimal solution is to minimize rebuffering event while maximizing average bitrate through frequently changing requested bitrates.

It should be noted that the three objectives defined above are normalized to ensure impartial comparisons between video segments of dissimilar lengths. Since the QoE depend on perception, the precedence of one factor would vary from one viewer to another. The QoE of video segment no. 1 through segment no. s is defined by a weighted sum of the listed objectives. The formulation of the bitrate adaptation for the QoE maximization problem is denoted by QoE, where, given the throughput trace R_i^n as input, the optimization problem provides the bitrate decisions $R_1^v, \dots, R_{N_s}^v$ as output. The function QoE is given by:

$$QoE = \sum_{i=1}^{N_s} Q(R_i^v) - \lambda \sum_{i=1}^{N_s} (T_s R_i^v / R_i^n - T_i^B) - \mu \sum_{i=1}^{N_s-1} |Q(R_{i+1}^v) - Q(R_i^v)|. \quad (4.5)$$

Here, λ and μ are non-negative weight parameters corresponding to the rebuffering duration and quality switching, respectively. A relatively small λ indicates that the viewer is not particularly concerned about rebuffering duration. Increasing the λ reflects

more effort to achieve shorted rebuffering durations. Similarly a large μ , relatively to the other parameters, indicates that a user is more concerned about quality switching. This QoE maximization problem is subject to a set of constraints on the buffer size limit and the segment size and the segment set:

$$\max \quad \{\text{QoE}\}, \quad (4.6)$$

$$s.t \quad T_{i+1}^B = T_i^B + T_s \left[1 - \frac{R_i^v}{R_i^n} \right], \quad (4.7)$$

$$0 \leq T_i^B \leq T_{\max}^B, \quad (4.8)$$

$$R_i^v \in \mathbb{R}, \text{ for } i = 1, \dots, N_s. \quad (4.9)$$

The first constraint, stated in equation (4.7), shows the buffer update through the playback and the reception of a video segment. The second constraint, stated in equation (4.8), shows the buffer size limits. Finally, the third constraint, stated in equation (4.9), shows that all R_i^v are real.

4.3. Rate-Based Adaptation

Rate-based (RB) adaptation is a traditional adaptation approach that selects the highest bitrate that is smaller than the predicted throughput regardless of the previous selected bitrates. This is referred to as a stateless adaptation algorithm [37]. Figure 4.2 illustrates the RB model implementation. As illustrated in the figure, the adaptation method is a simple quality level selector based on the predicted adaptation network throughput \bar{R}_i^n which is estimated through a moving average window of N steps. The buffer status is not considered during the quality selection decision.

4.4. QoE-Based Adaptation

Following the QoE problem formulation described in Section 4.2, an optimization model for bitrate adaptation that considers network throughput and buffer occupancy as illustrated in Figure 4.3. The figure shows the main components of the QoE-based adaptation method. The client has a network throughput predictor and buffer that feed the

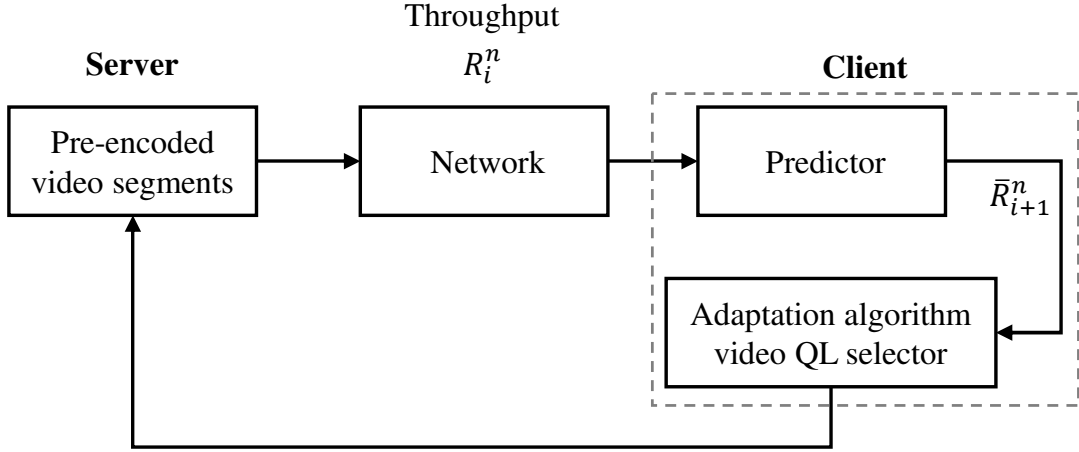


Figure 4.2: Rate-based model implementation.

adaptation algorithm video QL selector with the predicted network throughput \bar{R}_{i+1}^n and the buffer occupancy T_i^B . This study proposes a simple and fast algorithm which limits quality switching events and lowers the severity of quality jumps such that drastic drops or surges are avoided, so as to maintain smooth user-perceived experience. This is achieved by modifying the QoE function, formulated in equation (4.5), through re-representing the nonlinear term, caused by the absolute values and related to the quality variation QW by imposing two linear constraints such that quality jumps are limited by a specific threshold and ensure that switching follows a smooth step-like behavior. The adaptation method ensures that the selected bitrate of the segment $i + 1$ is not to be greater than one quality bitrate level defined as Q_{th} compared to segment i , which arguably can be better for user perception. These constraints are formulated in equations (4.11) and (4.12). Four QLs are introduced and mapped to the different selected bitrate ranges. For example bitrates from 500 kbps to 1000 kbps are mapped to Q_1 and so on. The quality level will better illustrate the quality switching later on. Network throughput \bar{R}_i^n is estimated through a moving average window of N steps. Rebuffering events are also taken into consideration. Starvation instances happen when the playback buffer of the client is emptied and the consecutive video segment does not arrive before the scheduled playback time, the video playback temporarily stops. This time is measured by subtracting the segment download from the time available in buffer. If the segment download time is greater than the occupancy, then a rebuffering event occurs.

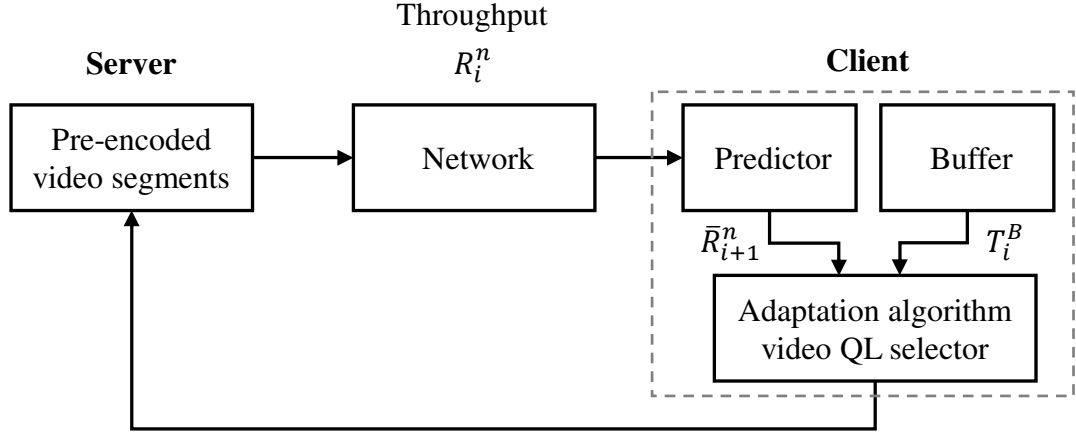


Figure 4.3: QoE-Based model implementation.

The implemented objective problem is defined below:

$$\max \quad \text{QoE} = \sum_{i=1}^{N_s} N_s Q(R_i^v) - \lambda \sum_{i=1}^{N_s} T_s R_i^v / R_i^n - T_i^B, \quad (4.10)$$

$$s.t \quad Q(R_{i+1}^v) - Q(R_i^v) \geq Q_{th}, \quad (4.11)$$

$$Q(R_{i+1}^v) - Q(R_i^v) \geq -Q_{th}, \quad (4.12)$$

$$(4.7) \text{ and } (4.8) \text{ and } (4.9).$$

The approach proposed maximizing the average video quality and minimizing the rebuffering duration as indicated in equation (4.10). Here, to give equal importance to the two objectives the average video quality and the rebuffering duration, the λ assigned to the rebuffering duration is set to 1. Equations (4.11) and (4.12) illustrate the quality variation inequality constraints which reflect maintaining the quality jump to a certain threshold, for example limiting the threshold to one or two levels.

Similar to Section 4.2, the constraint in equation (4.7) shows the buffer update through the playback and the reception of a video segment. The second constraint in equation (4.8) shows the buffer size limits. Finally, the third constraint, stated in equation (4.9), shows that all R_i^v are real.

4.5. Hybrid-QoE-Based Adaptation Approach

The hybrid adaptation algorithm combines both the traditional RB and the proposed QoE-based adaptation approaches. The additional constraint, $R_i^v \leq \bar{R}_i^n$, which limits the selected bitrate to be less than or equal to the predicted network throughput is added to the previously discussed QoE adaptation algorithm. Algorithm 4.1 summarizes the working of the hybrid-QoE based adaptation approach.

Given the current buffer occupancy T_i^B , previous bitrate R_{i-1} and throughput prediction \bar{R}_i^n , the optimal bitrate is found by solving the QoE maximization function defined in equation (4.10). The constraints are given in equations (4.7), (4.8), (4.9), (4.11), and (4.12). Finally the $R_i^v \leq \bar{R}_i^n$ constraints are also set as part of the optimization problem. Off-the-shelf solvers such as CPLEX can be used to solve the optimization problem. Since the problem is a linear one computational overhead may not be a significant concern.

Algorithm 4.1 Hybrid QoE-based Adaptation Workflow

- 1: Initialize system parameters: R^v , $Q(R^v)$, T^B , T_s , N_s , R^n , and Q_{th}
 - 2: Define QoE optimization problem
 - 3: Estimate network throughput \bar{R}_i^n
 - 4: **for** $i = 1$ to number of segment N_s **do**
 - 5: $T^B(i + 1) = T^B(i) + T_s - T_s R^v(i) / R^n(i)$
 - 6: **end for**
 - 7: Declare the constraints
 - 8: **for** $i = 1$ to $N_s - 1$ **do**
 - 9: $Q(R_{i+1}^v) - Q(R_i^v) \geq Q_{th}$
 - 10: $Q(R_{i+1}^v) - Q(R_i^v) \geq -Q_{th}$
 - 11: $R_i^v \leq \bar{R}_i^n$
 - 12: **end for**
 - 13: Download optimal segment with bit rate R_i^v
-

4.6. Simulation of the Hybrid Model

A simulation testbed based on the video streaming model for the proposed hybrid adaptation approach is implemented using MATLAB®. The testbed simulates the video

player buffer dynamics during the process of receiving and playing back video segments based on a bitrate range and network profiles. Once the optimization problem is defined in MATLAB®, the CPLEX solver is utilized to solve for the optimum bitrate. The optimum bitrate for each segment is then mapped to the closest bitrate value in the available video representation rates. The following section discusses the settings that were selected to conduct the simulations.

4.6.1. Video parameters

In the simulation the video used is Big Buck Bunny, which is a simple animation shortclip of 10 minutes and 34 seconds duration under the Peach open movie project. The video content consists of animated characters with a non-intricate background [38]. The video in the dataset is encoded by the H.264/MPEG-4 codecs to 13 different representation rates, ranging from 235 kbps to 40 Mbps. The 4-second segment group is selected from the full dash profile.

4.6.2. Network traces

The proposed approach is examined using realistic network environment conditions. Real network traces are used from a fixed WLAN network and a mobile 5G network. It is important to note the difference in the offered throughput and the mobility pattern in the two networks as it affects the quality and the performance of the approaches.

4.6.2.1 Fixed channel environment

Average network throughput trace from a house-WLAN is obtained from [39] and used in the deployed algorithm. The throughput ranges between 0-2 Mbps. The dataset contains client-side cellular key performance indicators such as throughput information and other context related metrics.

4.6.2.2 Mobile channel environment

5G network trace is taken from the first publicly available dataset [1] which is originally collected from an Irish mobile operator. The dataset is created from two mobility patterns; static and driving, in which throughput varies from 0 to 200 Mbps. It contains client-side throughput information and other channel context related metrics. These metrics are gener-

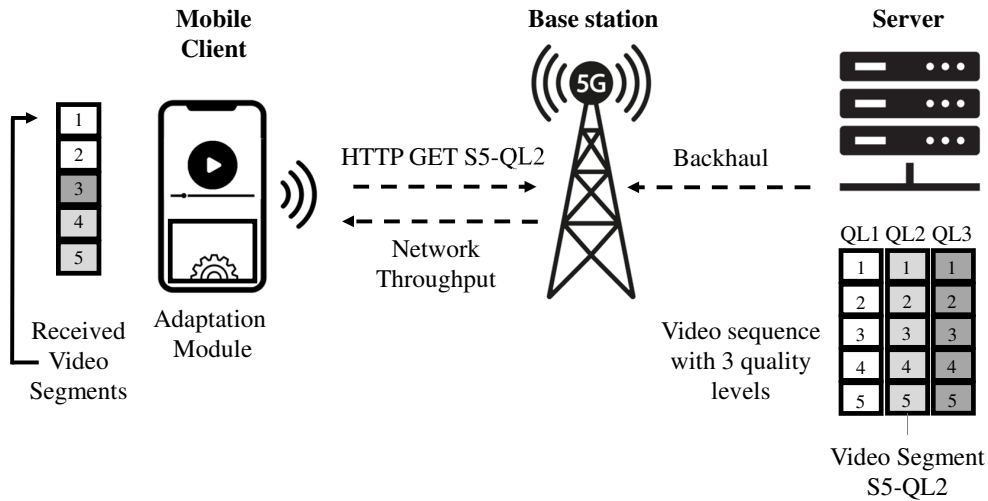


Figure 4.4: Illustration of segment requests in DASH in 5G network.

ated from a well established Android network monitoring application, G-NetTrack Pro. Figure 4.4 illustrates the simulated model.

4.7. Hybrid Model Simulation Results

The algorithm is evaluated on the testing datasets, both fixed and mobile . The algorithm selects the rate of the video segment to be downloaded.

The selected bitrate is mapped to one of four quality levels as described in Section 4.4 to better illustrate the quality changes. The performance of the proposed hybrid QoE-based for each channel environment type is illustrated in the following sections. The algorithm’s performance is compared to the classical RB algorithm. A window of 500 seconds is captured to view the network throughput, mapped quality level and buffer occupancy.

4.7.1. Fixed channel environment

The simulation results of the adaptation approach are analyzed and evaluated. Figures 4.5 and 4.6 show the RB and Hybrid QoE-based adaptation approaches respectively. The figures show the network throughput, the quality level changes and buffer occupancy simulation results in the fixed channel environment. Observing Figure 4.5 it is noted that the quality level tends to follow the throughput pattern with few quality

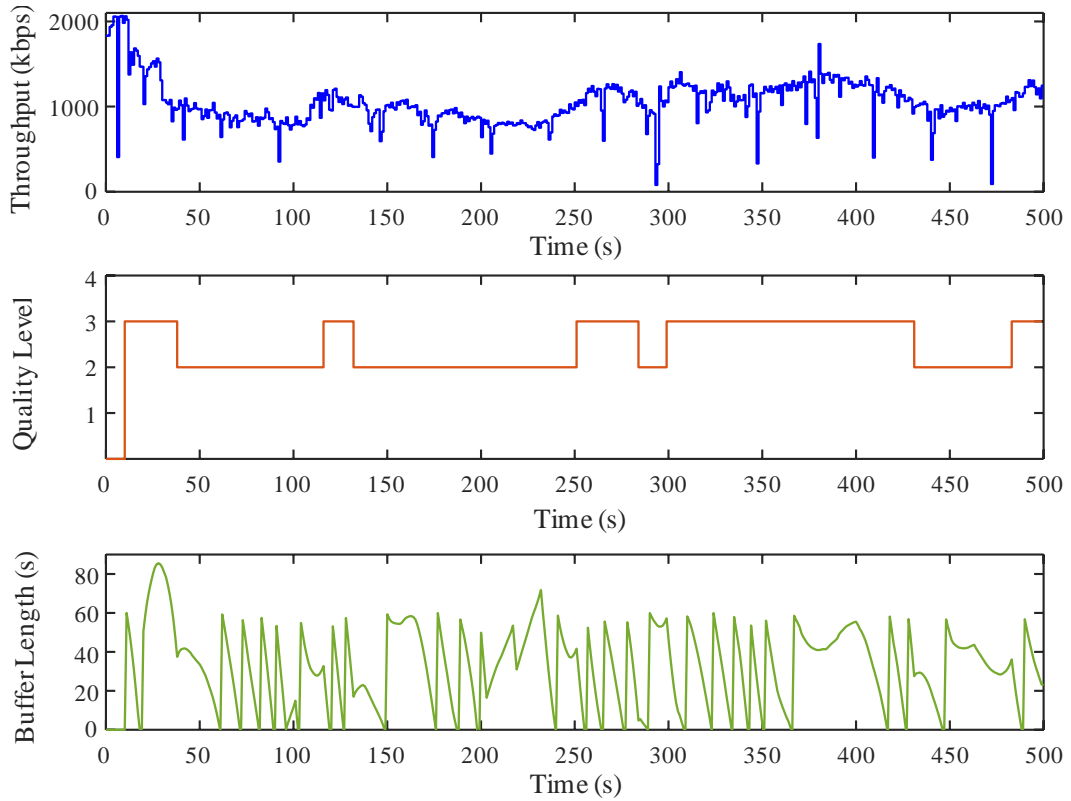


Figure 4.5: RB simulation in WLAN network.

jumps. The playback buffer on the other hand suffers from multiple starvation instances (as the buffer occupancy hits zero at multiple instances) and consequently endures several re-buffering events. Looking at Figure 4.6 it is observed that the quality selected by the Hybrid-QoE approach on average is higher than that of RB, but the quality changes are slightly higher. As for the buffer occupancy, the buffer loads new segments as the slope increases and plays video segments as slope decreases. The buffer suffers from starvation at certain instances which is caused by the drop in the network throughput level and the respective drop in quality levels. Clearly rebuffering is reduced drastically.

4.7.2. Mobile channel environment

Figures 4.7 and 4.8 show the RB and Hybrid QoE-based adaptation approaches respectively in a static mobile network. It is noted that the network throughput fluctuates in a regular pattern. Observing Figure 4.7 the quality level tends to follow the throughput pattern with few quality jumps. Also, the playback buffer on the undergoes multiple

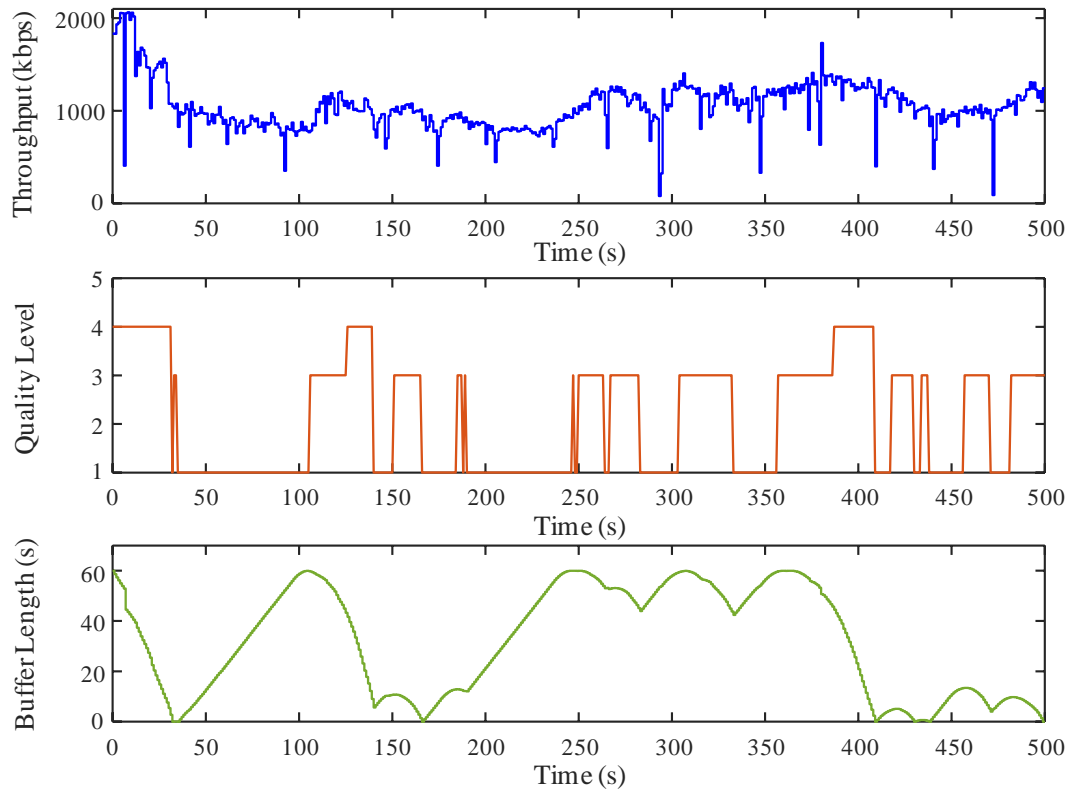


Figure 4.6: Hybrid QoE-based simulation in WLAN network.

starvation instances. Figure 4.8 shows that the quality selected by the Hybrid-QoE approach on average is slightly lower than that of RB, but the quality changes are lower. The buffer suffers from starvation at certain instances which is caused by the drop in network throughput level. Again rebuffering is improved greatly. Figures 4.9 and 4.10 show the RB and Hybrid QoE-based adaptation approaches respectively in a moving 5G network. It is noted that the network throughput fluctuates randomly and then suddenly drops after 300 seconds. Figure 4.9 the quality level fluctuates with the network pattern. Again, the rebuffering events became worse in the moving environment as the instances increased at a faster rate. Figure 4.10 shows the performance of the hybrid in a moving environment. Similar to RB, rebuffering instances increased due to the network fluctuation but it is still significantly lower than that in RB. However, the algorithm maintained a the quality level and quality jumps are reduced in comparison to RB.

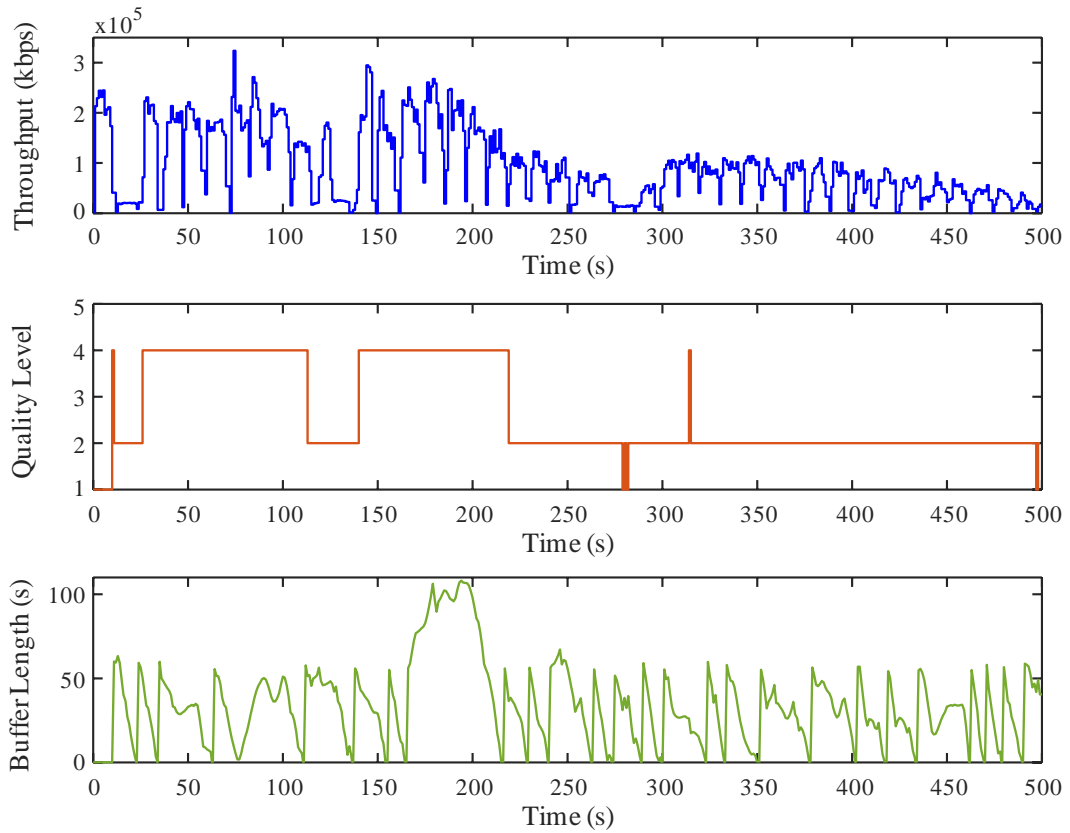


Figure 4.7: RB simulation in static 5G network.

4.8. Hybrid Model Results Summary

Observing the simulations of the RB and the hybrid-QoE approaches its concluded that the hybrid-QoE method outperforms RB. Although RB maintains a relatively low number of quality jumps in different environments, it greatly suffers from rebuffering events. It is evident that RB does not manage the buffer occupancy well which would greatly impact its performance. On the other hand, hybrid QoE significantly reduces rebuffering events and manages quality switching times at a number comparable to RB. Hybrid QoE maintains a fair trade-off between quality switching times and rebuffering events.

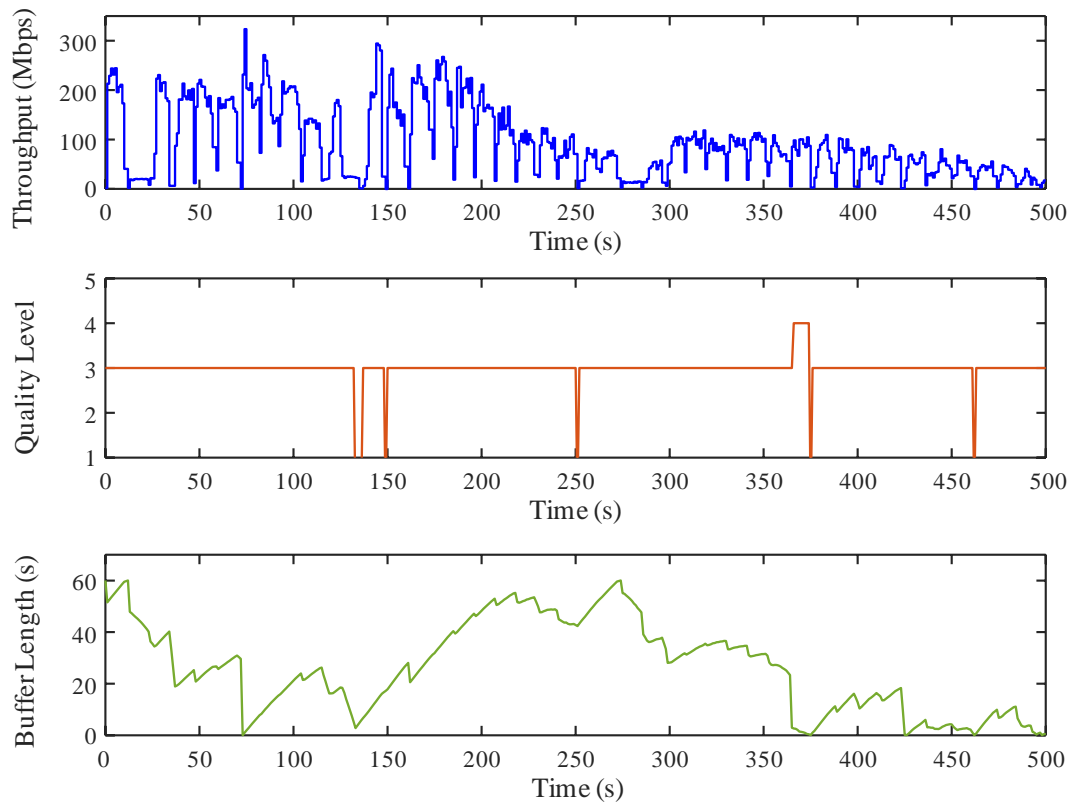


Figure 4.8: Hybrid QoE-Based simulation in a static 5G network.

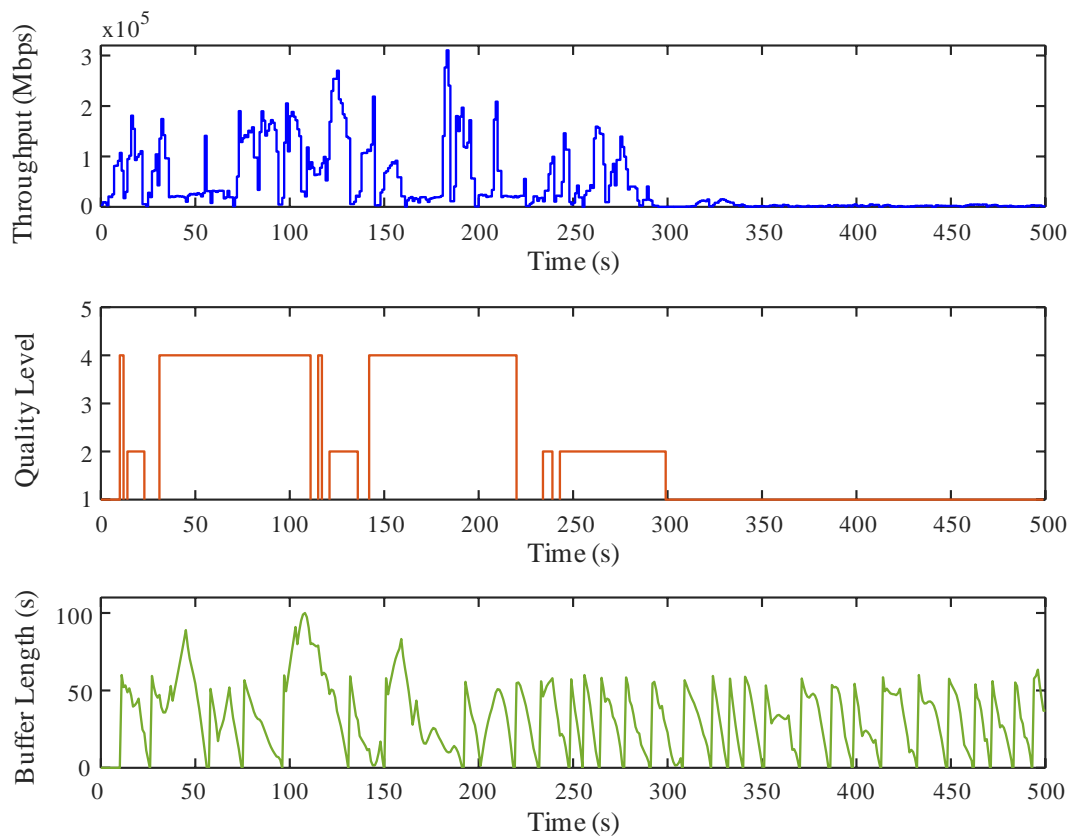


Figure 4.9: RB simulation in a moving 5G network.

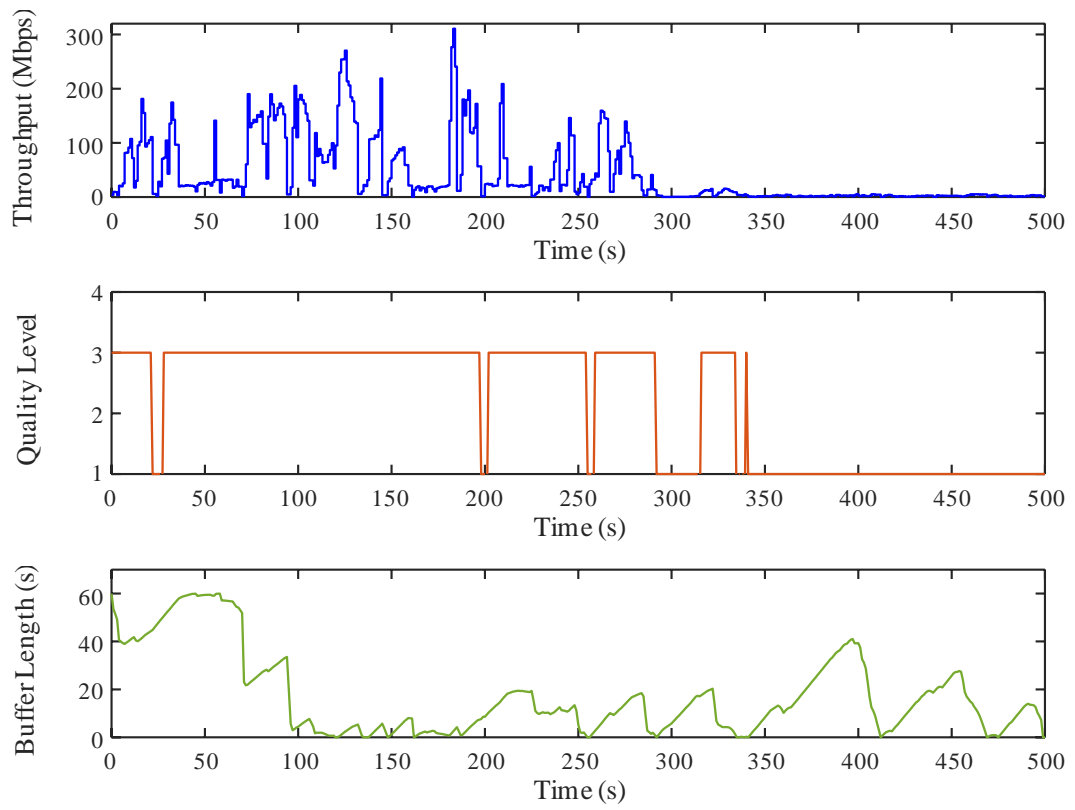


Figure 4.10: Hybrid simulation in a moving 5G network.

Chapter 5: Proposed Reinforced Learning Model

In this chapter, the rate adaptation for DASH is formulated into a traditional RL problem in which an agent interacts with the environment. In the following, the design and implementation of the proposed RL-based adaptation approach then the details of the DQNReg algorithm are explained. The methodology of the RL-based adaptation is highlighted. Finally the simulation setup is explained and the simulation results are presented.

5.1. DQNReg Algorithm

As the deep reinforcement learning research area thrives, many enhancements for deep Q-learning have been constantly proposed, most commonly variants of the DQN algorithm. The state-of-the-art method DQNReg is introduced by Co-Reyes et al. [40] which builds on the classical DQN algorithm by adding a weighted penalty on the Q-values to the normal squared Bellman error (DQN loss function is explained in Section 2.3.3.). The authors in [40] proposed method rediscovered the temporal difference (TD) and enabled interpretable alterations on existing algorithms like DQN. They utilize evolution strategies by exploring the space of computational graphs which calculates the loss function for the learning agent. Co-Reyes et al. highlight that the discovered DQNReg algorithm show improved generalization performance over DQN in new environments that are not experienced during training. The analysis in [40] shows that DQNReg outperforms the DQN algorithm and other well known variants such as double DQN (DDQN) on several Atari games.

The loss function for DQNReg is exhibited in the equations below:

$$\text{Let } Y_t = r_t + \gamma \max Q_{\text{targ}}(s_{t+1}, a), \quad (5.1)$$

$$\text{and } \delta = Q(s_t, a_t) - Y_t, \quad (5.2)$$

$$L_{DQNReg} = 0.1 * Q(s_t, a_t) + \delta^2. \quad (5.3)$$

Equation (5.1) shows the target Q-value which is the instantaneous reward r_t added to the discounted and maximized Q-value at state s_{t+1} . Equation (5.2) is similar to equation (2.6) but without the square. Finally equation (5.3) shows the DQNReg loss function which is the weighted Q-value penalty, $0.1 \times Q(s_t, a_t)$ added to the squared error.

Classical DQN algorithm tends to overestimate the Q-value [40], which might be a potential problem. It shows that the learned constraints starts early in the training. To address this issue and avoid overestimation, the weighted penalty is added to DQNReg loss function as shown in equation (5.3).

Considering the characteristics of the DASH rate adaptation, DQNReg is expected to enable the trained agent to obtain improved QoE performance gain. The QoE-based reward function is explained in the following section.

5.2. Methodology

5.2.1. Reward function

The QoE is impacted by the video quality of the viewed segment, the frequency of quality switching and the experienced re-buffering events. A reward function that represents these factors is introduced to issue policies that maximize the QoE perceived by the users. Similar to [3,28] and with reference to the video streaming model explained in Section 4.1, the reward function is then defined as follows:

$$QoE = R_i^v - \mu RD - |R_{i+1}^v - R_i^v|, \quad (5.4)$$

where R_i^v is the bitrate of the i -th video segment and RD is the rebuffering duration, which is experienced when the segment playout buffer level as a segment is downloaded is lower than the needed segment download time. It is calculated as:

$$RD = \frac{T_s R_i^v}{R_i^n} - T_i^B. \quad (5.5)$$

The parameter μ is a penalty coefficient for the experienced rebuffering event. Finally, the term $|R_{i+1}^v - R_i^v|$ reflects the quality variation between two consecutive segments.

5.2.2. Video streaming environment

The video streaming environment consists of a set of videos encoded at different rates. An internal representation of the client’s playback buffer is maintained. A download time is assigned based on the segment’s bitrate and available network throughput. To represent the video playback during the download, the playback buffer is drained by the current segment download time. In case the playback buffer is fully occupied, 500 ms delay is applied before fetching the other segment. The video streaming mathematical model used is similar to the one discussed in Section 4.1.

After each segment download, various state observations such as buffer occupancy and current segment bitrate are passed to the learning agent. The learning agent then tries to optimize the reward signal (QoE), which is impacted by the varying network conditions. To design a network that faithfully emulate real conditions, throughput from a corpus of real network traces were used to model varying conditions. The traces are used to shape the agent’s experience and help it predict the environment dynamics such as the anticipated network throughput. It is necessary to note that the quality of the dataset impacts the performance of the learning agent. These traces allow the agent to experience the magnitude and speed of throughput variations in real networks.

5.2.3. DQNReg rate adaptation

State-of-the-art DQNReg algorithm is used. The detailed functionality is explained as follows: Once the segment is downloaded, the learning agent receives the state inputs at segment i ,

$$s_i = (R_{i-1}^v, R_i^n, T_i^B, T_{i-1}^D, N_R), \quad (5.6)$$

where R_{i-1}^v is the bitrate at which the last video segment was downloaded, R_{i-1}^n is the past segment throughput measurement, T_i^B is the current buffer level, T_{i-1}^D is the download time of the past video segment, and N_R is the available bitrate for the next video segment.

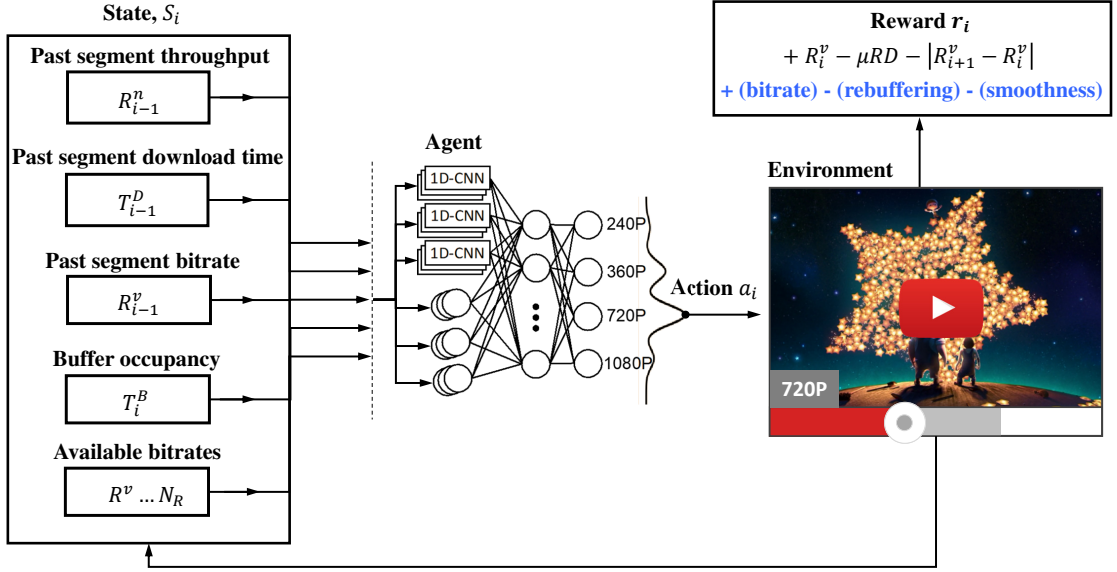


Figure 5.1: Basic network architecture for the proposed DQNReg network.

The learning agent observes the s_i from the environment then takes an action a_i which is selecting the bitrate for the upcoming video segment, in turn the agent receives the corresponding reward. It is to be noted that the state transition in the environment is also impacted by the action taken. The action selected depends on the *policy* π , which as defined earlier, the mapping of the action to state or the probability distribution over actions. $\pi(s_i, a_i)$ is the probability that action a_i is taken in s_i .

Practically, there are intractably several s_i, a_i pairs, to overcome this a neural network is utilized to represent the policy with manageable number of parameters θ , usually referred to as policy parameters. The advantage of neural networks is that they can deal with raw signals and do not need to have hand-crafted features.

In value based RL, the agent along side the neural network is expected to extract important features from the state, provide accurate estimation of the state-action value and finally derive the optimal policy. For a well-trained value network the optimal policy is derived as follows:

$$\pi_\theta = \arg \{ \max Q(s_i, a) \}. \quad (5.7)$$

The neural network architecture is not a priority of the research on the RL-based adaptation method, hence a default network architecture was designed for the simulation

experiment. The learning process of the agent is the training process of the state-action value network. By regularly observing the environment, the agent gathers tuples like previous state, new state, action and reward. The TD technique is then applied to perform gradient descent [34]. This allows the value network to estimate the real state-action value function with adequate accuracy.

5.3. Simulation of the RL-based Model

In this section the proposed DQNReg RL-based adaptation approach is simulated in fixed and wireless channel environments. The implementation algorithm and the training process are described then the experimental setup is exhibited.

5.3.1. Simulation setup

A simulation testbed based on the video streaming environment discussed in Section 5.2.2 and the mathematical model discussed in Section 4.1 is implemented in SimEvent® discrete event simulator in MATLAB®. The video dataset used is similar to the one discussed in Section 4.2.2. To train the agent a corpus of network traces is created through concatenating different excerpts of the network datasets discussed in Section 4.63 and 4.71. The size of the buffer is set $T_{max}^B = 60$ seconds which is common in a DASH video player. The segment-wise QoE reward is estimated as per the model described in Section 5.2.1 after each bitrate selection.

5.3.2. Implementation and training algorithm

The neural network architecture, similar to [28] and [34], is composed of 1D convolution layer composed of 128 filters. Output of these layers is then aggregated with other inputs in a hidden layer that uses 128 neurons to apply the ReLU activation function. The number of neurons in the output layer is equal to the adaptive bitrate set which is N_R .

The training is performed on the Big Buck Bunny video. Once the observation states are collected by the agent, the Q-value table or the weights in the network are updated until the policy converges. Certain hyper-parameters were set similar to [28, 33, 34].

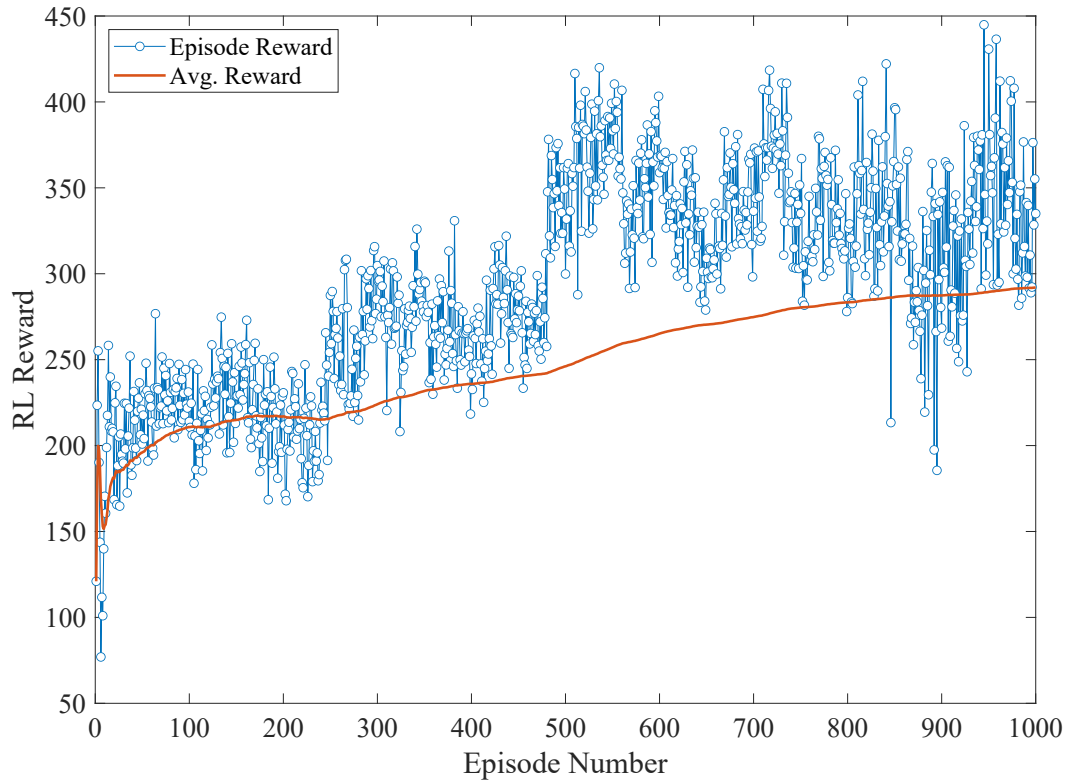


Figure 5.2: DQNReg learning convergence curve.

The discount factor used is $\gamma = 0.99$, the learning rate is 10^{-5} and the exploration adopted was the ϵ -greedy to explore many states and have a maintain a trade-off between exploration and exploitation.

The training algorithm takes the bitrate selection for a video segment as a step, it then takes the step experience and stores it into the replay memory. The average QoE reward on the training set is plotted against the number of training episodes as shown in Figure 5.2. The blue lines show the reward per episode.

5.3.3. Simulation results

The DQNReg model is evaluated on the testing datasets, both fixed and mobile as discussed in Section 4.6.2. The trained DQNReg agent is employed to pick the rate of the video segment to be downloaded. Once the bitrate is selected, the bitrate is mapped to one of four quality levels as described in section 4.4 to better illustrate the quality changes. The performance of the DQNReg agent for each channel environment type is illustrated in the following sections. The DQNReg performance is compared to the

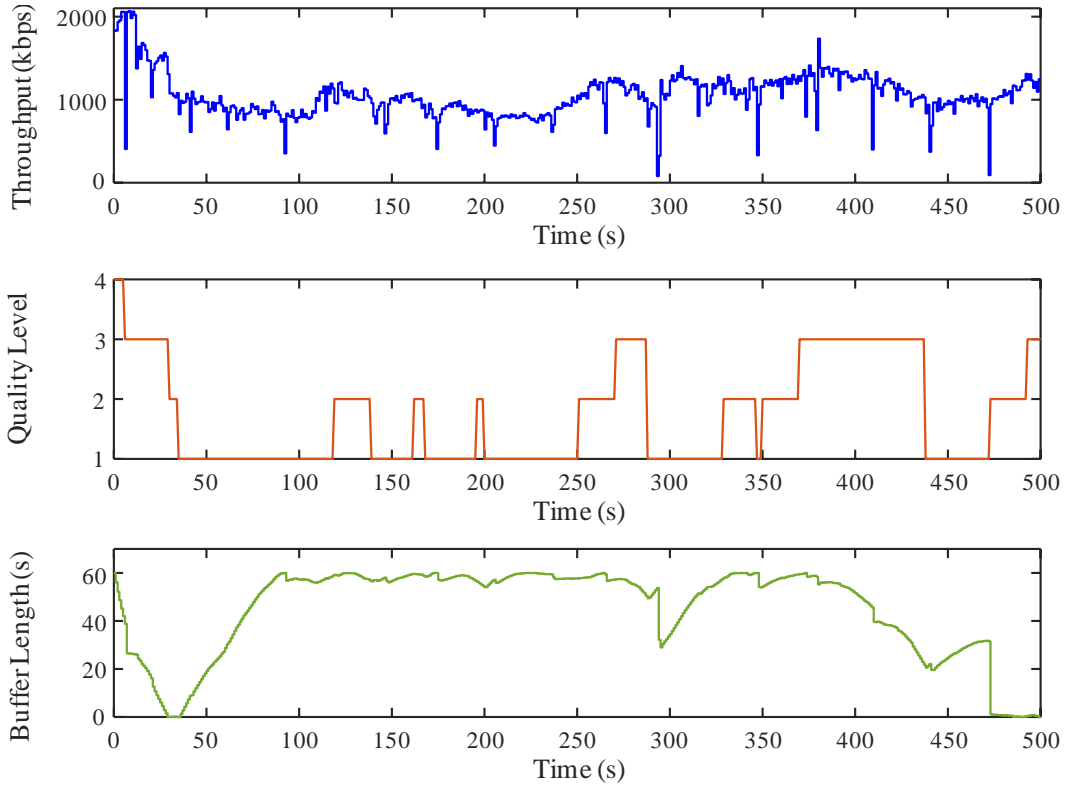


Figure 5.3: DQN simulation in a WLAN network.

classical DQN algorithm. A window of 500 seconds is captured to view the network throughput, mapped quality level and buffer occupancy.

5.3.3.1 Fixed channel environment

Simulation results of the adaptation approach are analyzed and evaluated. Figures 5.3 and 5.4 show the DQN and DQNReg adaptation approaches respectively. The Figures show the network throughput, the quality level changes and buffer occupancy simulation results in the fixed channel environment. Looking at Figure 5.3 it is noted that the quality level tends to fluctuate. The playback buffer on the other hand experiences few starvation instances. Looking at Figure 5.4 it is noted that the quality selected by DQNReg is more consistent and on average is higher than that of DQN, and quality switching is lower than that of DQN. The buffer plot shows significant improvement as the buffer does not starve and rebuffering does not take place.

5.3.3.2 Mobile channel environment

Figures 5.5 and 5.6 show the DQN and DQNReg algorithms respectively in a static

mobile network. Looking at Figure 5.5 the quality selection tends to be fluctuating between two levels. However, the frequency of the switching is quite high. In the third subplot the playback buffer shows few prolonged starvation instances. As for the DQNReg performance, Figure 5.6 show the improvement in quality switching compared to that of DQN. Also, the playback buffer undergoes starvation instances but for shorter lengths.

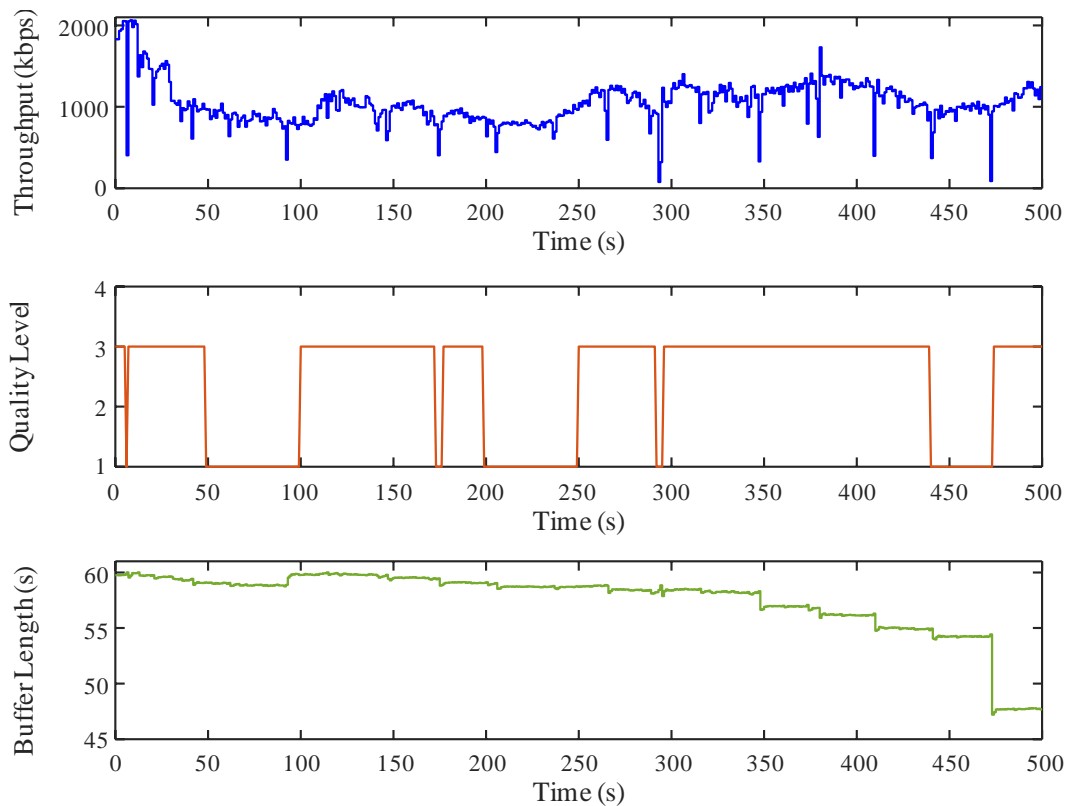


Figure 5.4: DQNReg simulation in a WLAN network.

In the last simulation, DQN and DQNReg algorithms are observed in a 5G cellular network while driving. Figures 5.7 and 5.8 show the DQN and DQNReg algorithms respectively. Looking at Figure 5.7 the quality selection tends to be fluctuating between 2 levels. Again, the frequency of the quality switching is high. In the third subplot the playback buffer shows starvation incidents taking place for a lengthy duration.

As for the DQNReg performance, Figure 5.8 shows remarkable improvement in quality switching compared to that of DQN. Still, the playback buffer undergoes starvation instances but for shorter periods of time.

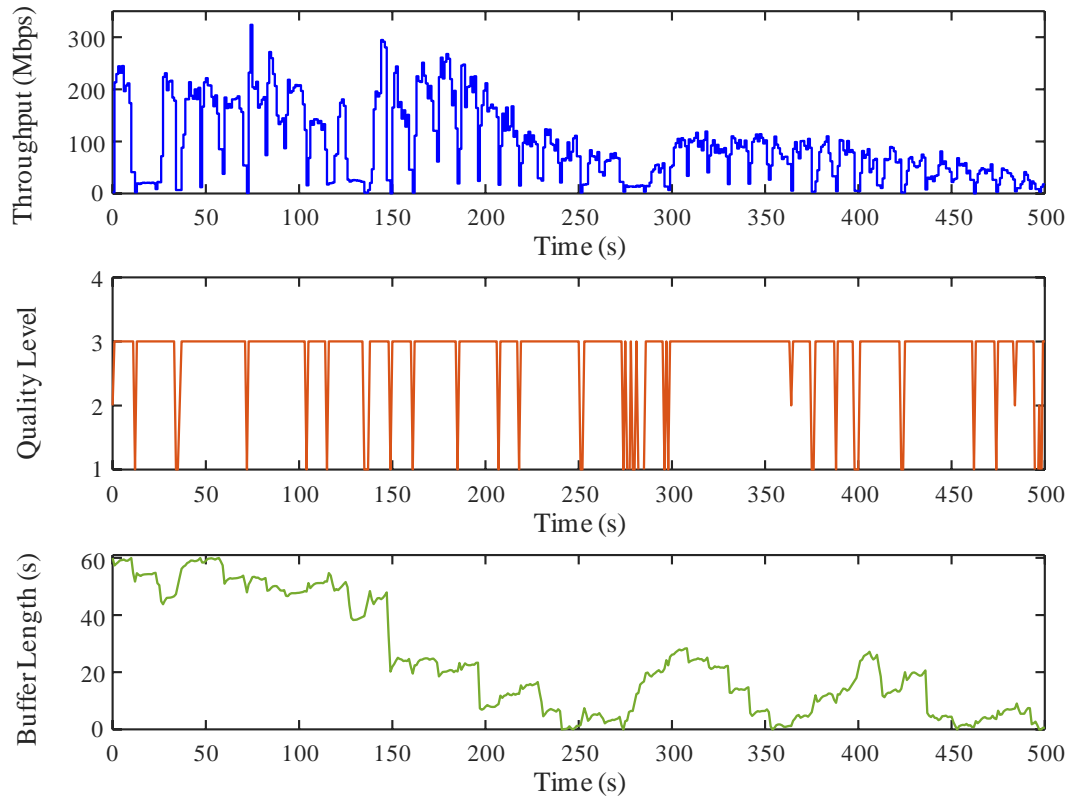


Figure 5.5: DQN simulation in a 5G network with stationary users.

5.4. Results Summary

Analyzing the simulation plots of DQN and DQNReg approaches, it is noted that both algorithms tend to maintain the same quality level. DQNReg out performs DQN with respect to quality switching while reducing the number of rebuffering events.

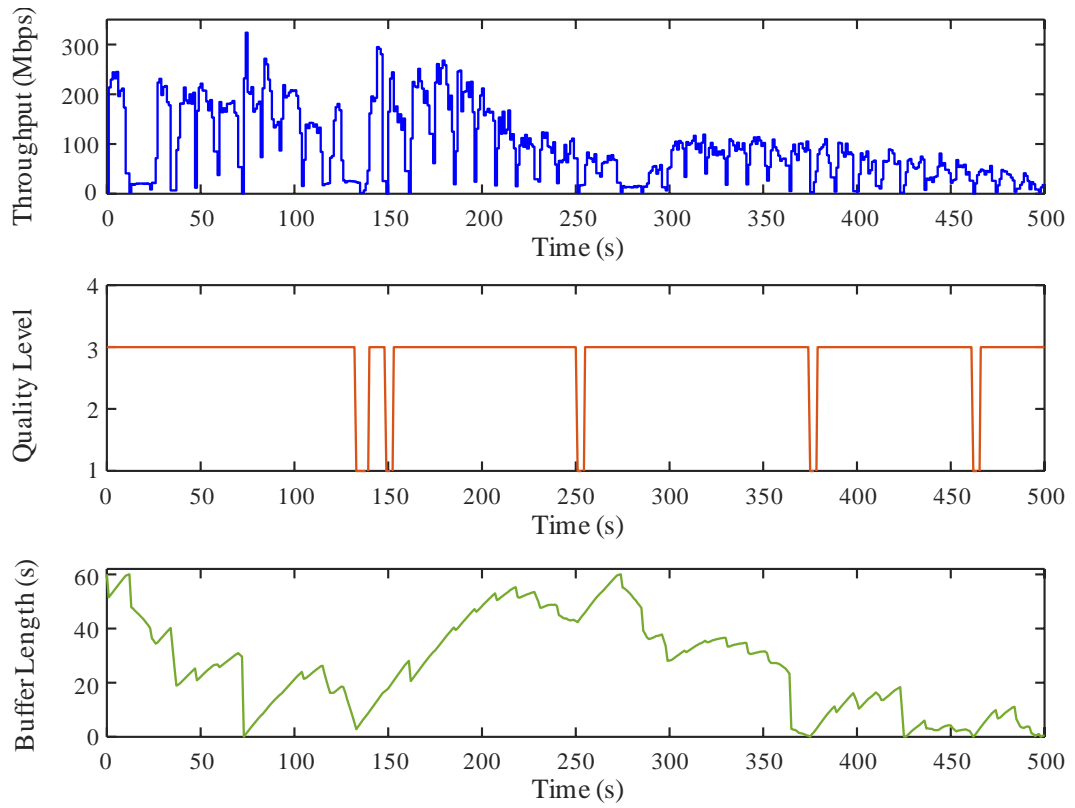


Figure 5.6: DQNReg simulation in a 5G network with stationary users.

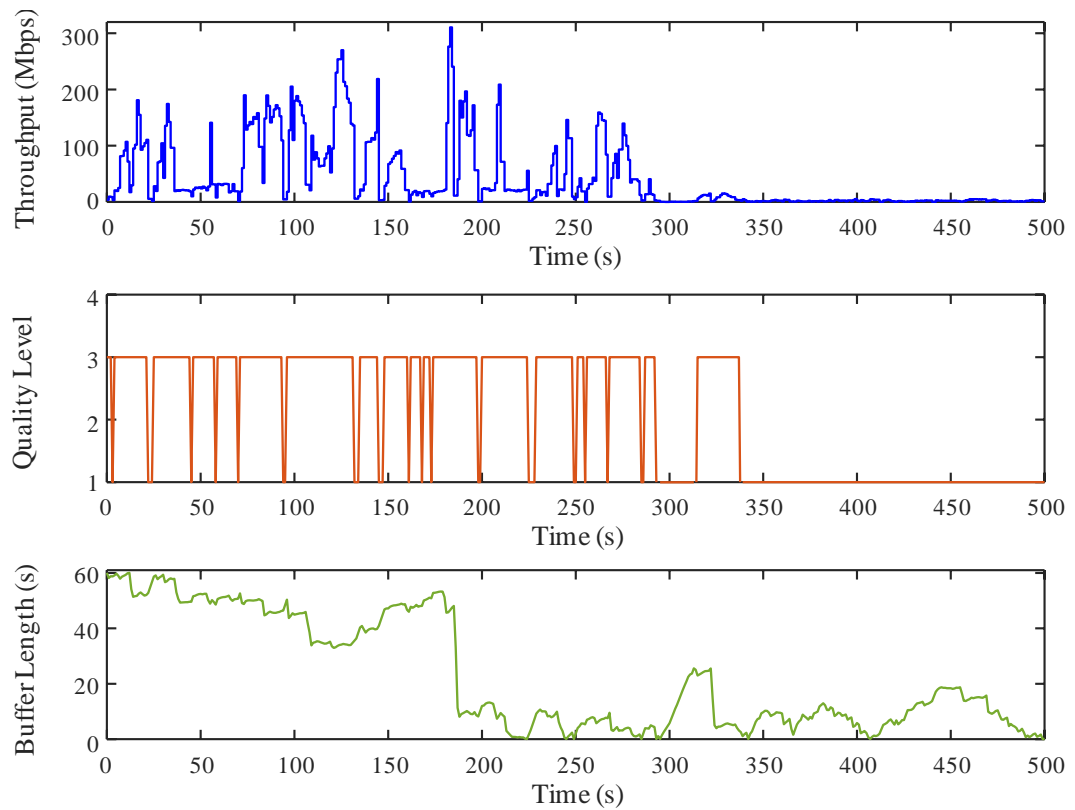


Figure 5.7: DQN simulation in a 5G network with mobile users.

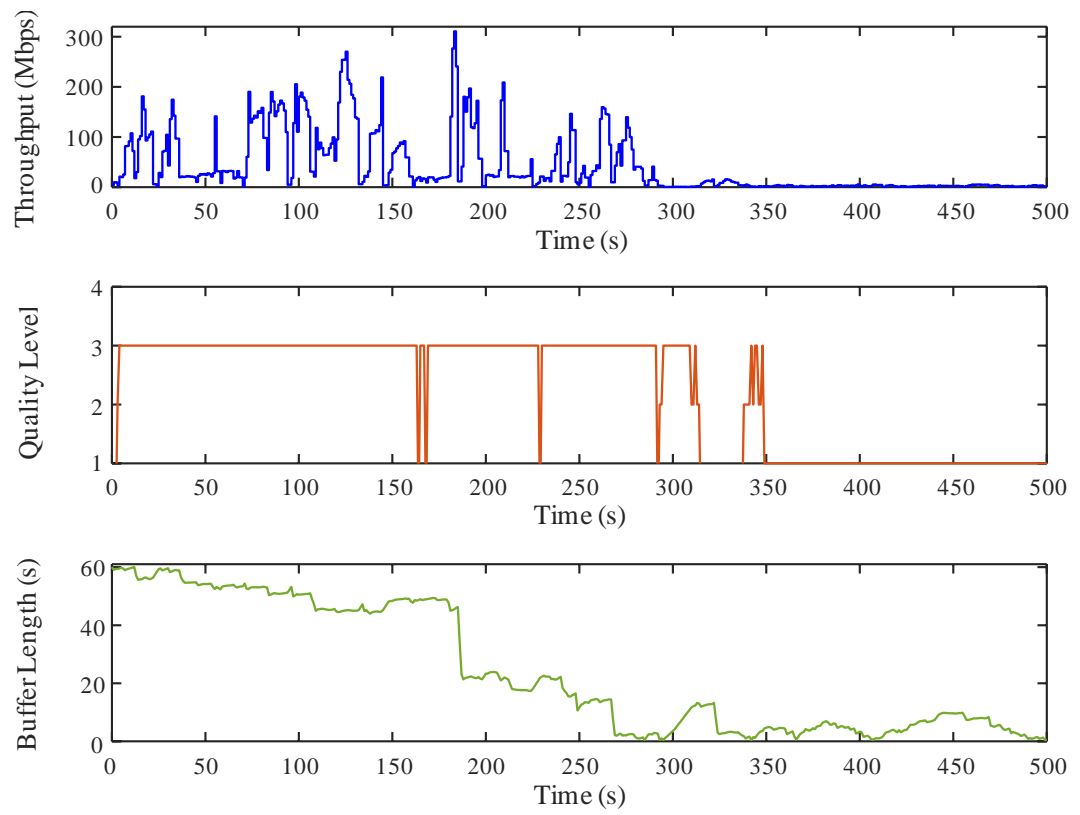


Figure 5.8: DQNReg simulation in a 5G network with mobile users.

Chapter 6: Results and Discussion

After performing extensive simulations in different network environments, the performance of the simulated approaches is analyzed and evaluated in this chapter. The evaluation metrics used are explained and then the proposed hybrid QoE-based and DQNReg adaptation approaches are evaluated and compared.

6.1. Evaluation Metrics

The performance of the adaptation methods is evaluated using the following metrics:

1. **Average QoE:** The QoE objective is a sum of weighted objectives that have varying orders of magnitudes and/or units. To make fair comparisons it is important to transform the objective functions in a way that they all have comparable orders of magnitude. There are various ways proposed for such a purpose, one of the classic methods is to normalize single objective functions by their respective absolute function values [41]. This ensures that objective functions are normalized to one to cancel the effect of their varying magnitudes; for instance the video bitrates are measured in thousands of kbps while the rebuffering duration is measured in seconds. High Average QoE reflect, high average video quality, low re-buffering duration and fewer quality switching times.
2. **Rebuffering times:** Measures the number of buffering instances (when buffer occupancy is zero).
3. **Rebuffering duration:** Measures the total rebuffering time (in seconds) during the video playback duration.
4. **Inter-starvation length:** The distance that separates successive starvation/rebuffering instants in seconds [42].
5. **Quality switching times:** Counts the number of times the algorithm switches across different quality levels. This reflects the number of times the user-perceived quality changes across video playback.

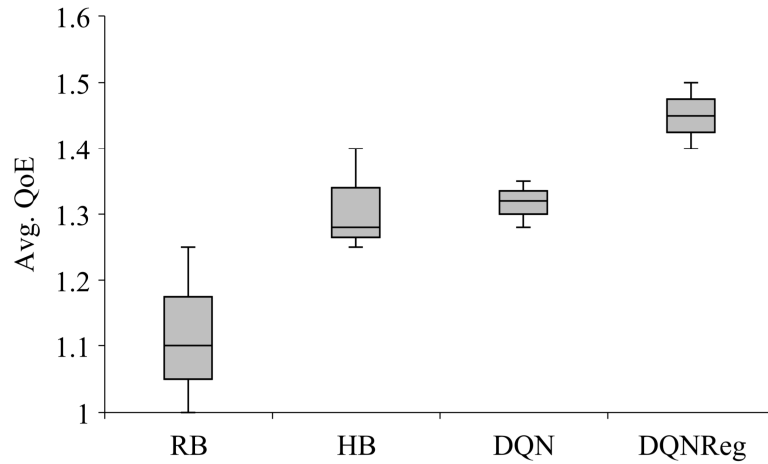


Figure 6.1: Average QoE for RB, HB, DQN, DQNReg methods.

6.2. Performance Comparison

6.2.1. Average QoE

Figure 6.1 illustrates the average QoE for all four simulated approaches across all simulated network environments. DQNReg achieved the highest average QoE of 1.45. The variance is small such that the average QoE value falls between 1.4 and 1.5. The average QoE for DQN is 1.32, it also has a small variance, as the QoE falls between 1.28 and 1.35. The proposed hybrid QoE average QoE is 1.28 which is comparable to DQN. Its falls between 1.25 and 1.4 which indicates it might perform better in certain conditions compared to other. Finally, the RB has the lowest average QoE of 1.1. Its values fall between 1 and 1.28, which means it can perform well in specific conditions like channels with good network throughput.

6.2.2. Rebuffering times

Figure 6.2 shows the number of rebuffering instances for all algorithms in all simulated network environments. Rate-based has the highest number of starvation instances of about 33 instance. This means that rebuffering instances will occur with rate-based regardless of the network condition. This is owed to the fact that the rate-based method ignores the playback buffer occupancy and considers the network throughput only. The hybrid QoE-based approach and DQN achieved a comparable number of around 7 with

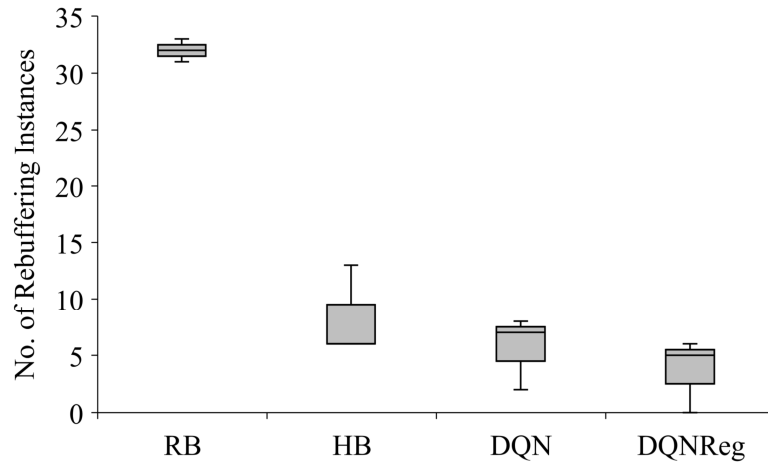


Figure 6.2: Rebuffering instances for RB, HB, DQN, DQNReg methods.

a small variance. Finally, DQNReg achieved the lowest average number of rebuffering times which is around 5. Its lowest value is zero which indicates that it does not experience rebuffering at certain times while other methods do. DQNReg variance is similar to that of DQN.

6.2.3. Rebuffering duration

Figure 6.3 shows the overall rebuffering duration (in seconds) experienced during the video playback for all algorithms in all simulated network environments. Again, the RB approach has the longest rebuffering duration of about 36 seconds and a small variance. The hybrid-QoE based method and DQN achieved similar average rebuffering duration of about 10 seconds. Finally, DQNReg achieved the lowest average rebuffering duration of about 7 seconds, while sometimes it does not experience rebuffering at all.

6.2.4. Inter-starvation length

Figure 6.4 illustrates the average inter-starvation length for all algorithms in all simulated network environments. Again, RB has the lowest performing method with respect to rebuffering. It has an inter-starvation length of about 4 seconds. RB has the lowest inter-starvation duration, meaning that rebuffering events will take place successively with short video playback time in between. The Hybrid QoE-based and DQNReg ap-

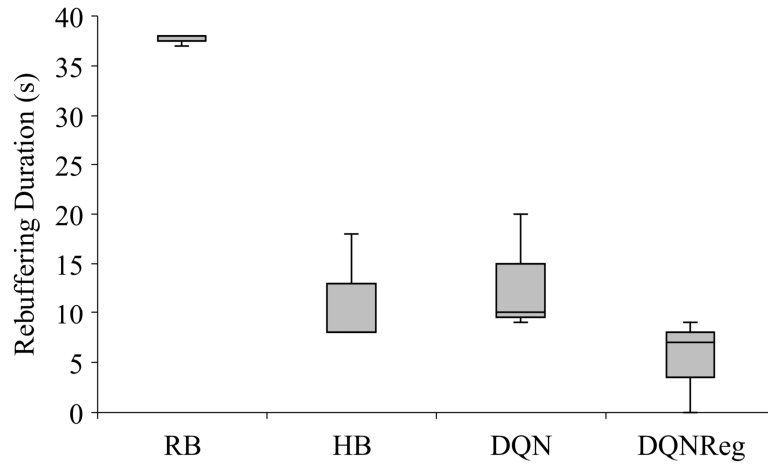


Figure 6.3: Rebuffering lengths for RB, HB, DQN, DQNReg methods.

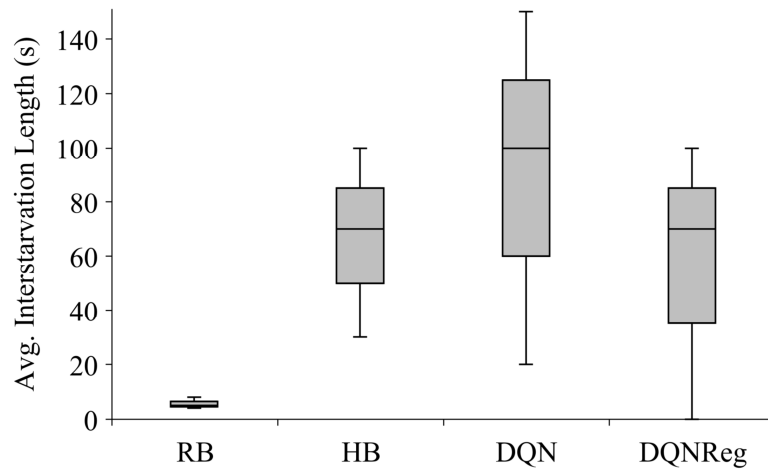


Figure 6.4: Inter-starvation lengths for RB, HB, DQN, DQNReg methods.

proaches have comparable average inter-starvation lengths of around 70 seconds, while DQN has the highest median value of about 100 seconds and is negatively skewed.

6.2.5. Quality switching times

Figure 6.5 shows the overall number of quality switch times experienced during the video playback for all algorithms in the simulated network environments. DQNReg keeps the quality stable with a low variance and a small number of switching times, with an average of 10 and a maximum of 12. Both rate-based and hybrid QoE-based have similar average for quality switching times of around 11 times. However, the hybrid QoE-based has a very high variance and is positively skewed. Similarly DQN

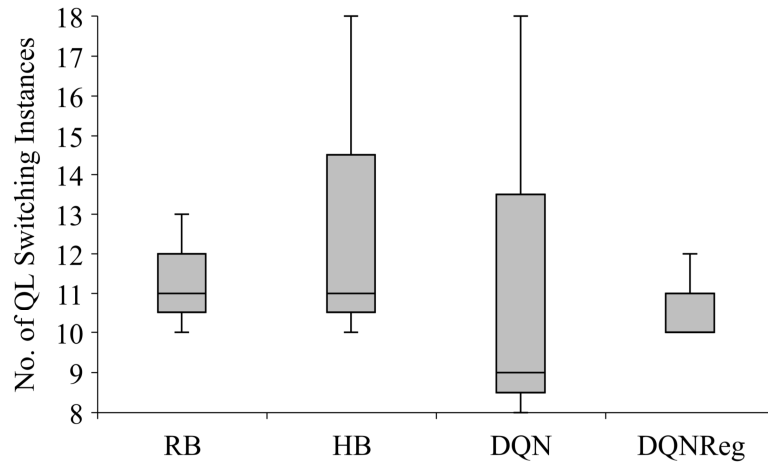


Figure 6.5: QL switching instances for RB, HB, DQN, DQNReg methods.

has a low average number of quality switching but a very high variance, which means that the number of quality switching may greatly vary across different environments.

6.3. Analysis and Discussion

Comparing the various simulated adaptation approaches, the advantage of rate based a low complexity algorithm is fewer quality switching times, however, it suffers greatly when it comes to handling the playback buffer issues such as rebuffering times, rebuffering duration and inter-starvation lengths. As for the hybrid-QoE based approach, it has lower quality switching times but some how experiences higher number of rebuffering times. Its performance is comparable to other methods with respect to rebuffering duration and inter-starvation lengths. On the other hand, DQN performs well with respect to number of rebuffering times and has a quite large inter-starvation length. It how ever has a high variance in the quality switching times, which may indicate its inability to generalize well under different environments.

By analyzing the various indicators, it is noted that when using the DQNReg approach, the number of rebuffering times, the rebuffering duration and the quality switching times are suppressed to the lowest, while the quality switching times and inter-starvation lengths are maintained at a level comparable to other methods. Under various mobility patterns of real-time network, the average QoE performance of DQNReg is still superior to other methods. This indicates that the trained DQNReg learning agent has strong

generalization ability and can flexibly adapt to various network conditions, so that the video service quality can match the network communication quality as well as possible.

Chapter 7: Conclusion and Future Work

7.1. Conclusion

With increased user expectations and demands for uninterrupted viewing and top video quality, studies concluded that users will leave video sessions if the quality is not adequate, harming the revenues of content providers [28]. Considering the intricate web-based video delivery ecosystem and its various bottlenecks, adaptive bitrate algorithms become essential to content providers to optimize video quality.

This thesis proposed two adaptation approaches: A hybrid QoE-based approach and DQNReg learning based approach for video adaptation. This hybrid approach deploys the traditional QoE evaluation metric with a twist. The adaptation problem is remodeled into a linear optimization problem, which is then split into a set of sub-optimization problems with their respective constraints. Various experiments with different network conditions are simulated. Simulation results showed that the proposed hybrid method outperforms classical RB approach as it uses both throughput estimation and buffer information to optimize a clearly defined QoE objective. The second approach proposed is the DQNReg, a reinforcement learning based technique that enhances the classical deep Q-learning method. A segment-wise QoE-based reward function is established so that the learning strategy can converge towards maximizing the QoE outcome. The proposed algorithms have been thoroughly evaluated using trace-based simulation for fixed and mobile networks. The DQNReg-based method outperforms other ones.

7.2. Future Work

Future research can integrate the initial start-up delay and the impact of latency onto the learning-based method, so that the learned policy can be better in terms of retrospective QoE. Moreover, the reinforcement learning algorithm can also be applied to the network resource allocation at the base station for multiple DASH clients, where the decision is made according to the network resource status and the received bitrate requests of the users.

References

- [1] D. Raca, D. Leahy, C. J. Sreenan, and J. J. Quinlan, “Beyond Throughput, the next Generation: A 5G Dataset with Channel and Context Metrics,” in *Proceedings of the 11th ACM Multimedia Systems Conference*, Jun. 2020, pp. 303–308.
- [2] Ericsson Mobility Report. Internet. [Online]. Available: <https://www.ericsson.com/en/reports-and-papers/mobility-report/reports/june-2021> [Accessed: Oct. 1, 2021].
- [3] X. Yin, A. Jindal, A. Sekar, and B. 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 (SIGCOMM’15)*, Aug. 2015, pp. 325–338.
- [4] L. Yu, T. Tillo, and J. Xiao, “QoE-Driven Dynamic Adaptive Video Streaming Strategy With Future Information,” *IEEE Transactions on Broadcasting*, vol. 63, no. 3, pp. 523–534, Sep. 2017.
- [5] F. Dobrian, A. Awan, D. Joseph, A. Ganjam, J. Zhan, V. Sekar, I. Stoica, and H. Zhang, “Understanding the Impact of Video Quality on User Engagement,” *Commun. ACM*, vol. 56, no. 3, pp. 91–99, 2013.
- [6] C. B. Ameur, “TCP Protocol Optimization for HTTP Adaptive Streaming,” Ph.D. dissertation, Université de Rennes, 2015.
- [7] Y. Sani, “Advanced Modelling of Adaptive Bitrate Selection,” Ph.D. dissertation, Lancaster University, 2017.
- [8] Y. Shuai, “Dynamic adaptive video streaming with minimal buffer sizes,” Ph.D. dissertation, Saarland University, 2018.
- [9] F. Garcia and E. Rachelson, *Markov Decision Processes*. John Wiley & Sons, 2013, ch. 1, pp. 1–38.
- [10] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Bradford Books, 2018.
- [11] S. Thrun, “Efficient exploration in reinforcement learning.” Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-CS-92-102, January 1992.
- [12] J. S. Bridle, “Probabilistic Interpretation of Feedforward Classification Network Outputs, with Relationships to Statistical Pattern Recognition,” in *Neurocomputing*. Springer, 1990, pp. 227–236.
- [13] M. Tokic and G. Palm, “Value-Difference Based Exploration: Adaptive Control between Epsilon-Greedy and Softmax,” in *Proceedings of the 34th Annual German Conference on Advances in Artificial Intelligence*, 2011, pp. 335–346.

- [14] C. J. C. H. Watkins and P. Dayan, “Q-learning,” *Machine Learning*, vol. 8, no. 3, pp. 279–292, 1992.
- [15] V. Mnih *et al.*, “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
- [16] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, M. A. Riedmiller, “Playing Atari with Deep Reinforcement Learning,” *CoRR*, vol. abs/1312.5602, pp. 1–9, 2013.
- [17] M. J. Khan, S. Harous, and A. Bentaleb, “Client-driven adaptive bitrate techniques for media streaming over HTTP: Initial findings,” in *2020 IEEE International Conference on Electro Information Technology (EIT)*, 2020, pp. 053–059.
- [18] A. Bentaleb, B. Taani, A. C. Begen, C. Timmerer, and R. Zimmermann, “A Survey on Bitrate Adaptation Schemes for Streaming Media Over HTTP,” *IEEE Communications Surveys Tutorials*, vol. 21, no. 1, pp. 562–585, 2019.
- [19] T.-Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson, “A Buffer-Based Approach to Rate Adaptation: Evidence from a Large Video Streaming Service,” in *Proceedings of the 2014 ACM Conference on SIGCOMM*, 2014, pp. 187–198.
- [20] Y. Zhou, Y. Duan, J. Sun, and Z. Guo, “Towards simple and smooth rate adaptation for VBR video in DASH,” *2014 IEEE Visual Communications and Image Processing Conference*, 2014, pp. 9–12.
- [21] L. De Cicco, S. Mascolo, and V. Palmisano, “Feedback Control for Adaptive Live Video Streaming,” in *Proceedings of the Second Annual ACM Conference on Multimedia Systems*, 2011, pp. 145–156.
- [22] B. Wang, X. Luo, P. Hu, and F. Ren, “Improving optimization-based rate adaptation in dash system,” in *2017 26th International Conference on Computer Communication and Networks (ICCCN)*, 2017, pp. 1–9.
- [23] T.-Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson, “A Buffer-Based Approach to Rate Adaptation: Evidence from a Large Video Streaming Service,” *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 4, Aug. 2014, pp. 187–198.
- [24] R. K. P. Mok, X. Luo, E. W. W. Chan, and R. K. C. Chang, “QDASH: A QoE-Aware DASH System,” in *Proceedings of the 3rd Multimedia Systems Conference*, 2012, pp. 11–22.
- [25] N. Bouten, S. Latré, J. Famaey, W. Van Leekwijck, and F. De Turck, “In-Network Quality Optimization for Adaptive Video Streaming Services,” *IEEE Transactions on Multimedia*, vol. 16, no. 8, pp. 2281–2293, 2014.
- [26] Y.-L. Chien, K. C.-J. Lin, and M.-S. Chen, “Machine learning based rate adaptation with elastic feature selection for HTTP-based streaming,” in *2015 IEEE International Conference on Multimedia and Expo (ICME)*, 2015, pp. 1–6.

- [27] F. Chiariotti, “Reinforcement learning algorithms for DASH video streaming,” Ph.D. dissertation, Univeristy of Padova, 2014.
- [28] H. Mao, R. Netravali, and M. Alizadeh, “Neural Adaptive Video Streaming with Pensieve,” in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication (SIGCOMM’17)*, Aug. 2017, pp. 1–14.
- [29] M. Claeys *et al.*, “Design of a Q-learning-based client quality selection algorithm for HTTP adaptive video streaming,” in *Adaptive and Learning Agents Workshop, part of AAMAS2013, Proceedings*, 2013, pp. 30–37.
- [30] H. Lin *et al.*, “KNN-Q Learning Algorithm of Bitrate Adaptation for Video Streaming over HTTP,” in *2020 Information Communication Technologies Conference (ICTC)*, 2020, pp. 302–306.
- [31] T. Huang *et al.*, “Comyco: Quality-Aware Adaptive Video Streaming via Imitation Learning,” in *Proceedings of the 27th ACM International Conference on Multimedia*, 2019, pp. 429–437.
- [32] F. Chiariotti, S. D’Aronco, L. Toni, and P. Frossard, “Online Learning Adaptation Strategy for DASH Clients,” in *Proceedings of the 7th International Conference on Multimedia Systems*, May 2016, pp. 1–12.
- [33] M. Gadaleta, F. Chiariotti, M. Rossi, and A. Zanella, “D-DASH: A Deep Q-Learning Framework for DASH Video Streaming,” *IEEE Transactions on Cognitive Communications and Networking*, vol. 3, no. 4, pp. 703–718, 2017.
- [34] J. Liu, X. Tao, and J. Lu, “QoE-Oriented Rate Adaptation for DASH With Enhanced Deep Q-Learning,” *IEEE Access*, vol. 7, pp. 8454–8469, 2019.
- [35] A. Balachandran *et al.*, “Developing a Predictive Model of Quality of Experience for Internet Video,” *SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, Aug. 2013, pp. 339–350.
- [36] B. Wang, X. Luo, P. Hu, and F. Ren, “Improving Optimization-Based Rate Adaptation in DASH System,” in *2017 26th International Conference on Computer Communication and Networks (ICCCN)*, 2017, pp. 1–9.
- [37] J. Jiang, V. Sekar, and H. Zhang, “Improving fairness, efficiency, and stability in HTTP-based adaptive video streaming with Festive,” *IEEE/ACM Transactions on Networking*, vol. 22, no. 01, pp. 326–340, Jan. 2014.
- [38] J. J. Quinlan and C. J. Sreenan, “Multi-Profile Ultra High Definition (UHD) AVC and HEVC 4K DASH Datasets,” in *Proceedings of the 9th ACM Multimedia Systems Conference*, 2018, pp. 375–380.
- [39] WLAN Throughput Project. Internet. [Online]. Available: <https://data.world/engrasifkhan/wlanthroughput/workspacef> [Accessed: Mar. 1, 2020].
- [40] J. Co-Reyes *et al.*, “Evolving Reinforcement Learning Algorithms,” in *International Conference on Learning Representations*, May. 2021, pp. 1–15.

- [41] J. Arora, *Introduction to Optimum Design*. Academic Press, 2017.
- [42] H. Mukhtar, M. Hassan, and T. Landolsi, “An occupancy-based and channel-aware multi-level adaptive scheme for video communications over wireless channels,” *EURASIP Journal on Wireless Communications and Networking*, vol. 2011, no. 1, pp. 199–205, Dec. 2011.

Vita

Nada Abdelhafez graduated from the American University of Sharjah. She earned a B.Sc. degree in Electrical Engineering. After Graduation she worked for Etisalat as a telecommunications engineer. She worked on various projects like 5G spectrum allocation in the UAE, interference management and mobile network management automation and optimization.

In 2019, she joined the Electrical Engineering Master's program at the American University of Sharjah where she was awarded a two-year Graduate Assistantship from the Department of Electrical Engineering. During her graduate studies, she co-authored a paper which was presented in an international conference and published in a journal. Her research interests are in wireless communications, video streaming, mobile networks, and deep learning.