Resource Allocation Strategies for Multimedia Services in Cellular Networks

Thesis submitted to the
Indian Institute of Technology Guwahati
for the award of the degree

of

Doctor of Philosophy
in
Computer Science and Engineering

Submitted by
Satish Kumar

Under the guidance of
Dr. Arnab Sarkar and Dr. Arijit Sur

Department of Computer Science and Engineering
Indian Institute of Technology Guwahati
October, 2017

Abstract

The ever increasing demand for high bandwidth, low latency multimedia applications on mobile devices is set to pose an enormous challenge on the bandwidth allocation and multiplexing mechanisms in *Long Term Evolution (LTE)* and future wireless networks. In order to face this challenge, these cellular network infrastructures must be empowered with sophisticated but low overhead online radio resource allocation mechanisms to efficiently serve a variety of heterogeneous user equipments (mobile phones, laptops, tablets etc.) such that the *Quality of Service (QoS)/ Quality of Experience (QoE)* demands of all flows/end-users are met. In addition to satisfying *QoS/ QoE*, the resource scheduling mechanisms may also need to simultaneously cater to other practical constraints/objectives like limited power budget, maximizing spectral efficiency and graceful degradation in times of overload, in the face of ever changing network dynamics, user mobility etc. *In this dissertation, we present a few novel scheduling methodologies for system level as well as client centric QoS/QoE management corresponding to multimedia streaming over cellular networks in general, and LTE based systems in particular.*

The entire thesis work is composed of six distinct contributions which are categorised into four phases. In the first phase, scheduling strategies for generic real-time variable bit rate traffic was considered. The second phase extended the scheduling mechanisms designed in the first to specifically support non-adaptive video streaming. Mathematically structuring the intended design as an optimization problem with constraints, we have proposed optimal, stochastic and heuristic solutions for the same. The problem and algorithms designed in the first and

second phases were extended to handle adaptive video flows, in the third phase. We not only presented a *Dynamic Programming (DP)* solution but also streamlined the *DP* solution based on the characteristic of the system at hand in order to provide optimal solutions with far lower overheads. In addition, a scalable approximation algorithm which can judiciously trade-off scheduling overheads with solution accuracy, has been provided. While the first three phases dealt with the design of primarily in-network scheduling approaches, in the final phase, we have endeavored towards the development of client-side *SVC-DASH* based video streaming adaptation mechanisms that attempt to maximize the perceived *QoE* of an end-user. Experimental results have demonstrated the versatility and efficacy of the proposed approaches.

Declaration

I certify that:

a. The work contained in this thesis is original and has been done by me under the guidance of my supervisors.

b. The work has not been submitted to any other Institute for any degree or diploma.

c. I have followed the guidelines provided by the Institute in preparing the thesis.

d. I have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.

e. Whenever I have used materials (data, theoretical analysis, figures, and text) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the references. Further, I have taken permission from the copyright owners of the sources, whenever necessary.

Satish Kumar

Copyright

Certificate

This is to certify that this thesis entitled, **"Resource Allocation Strategies for Multimedia Services in Cellular Networks"**, being submitted by **Satish Kumar**, to the Department of Computer Science and Engineering, Indian Institute of Technology Guwahati, for partial fulfillment of the award of the degree of Doctor of Philosophy, is a bonafide work carried out by him under our supervision and guidance. The thesis, in our opinion, is worthy of consideration for award of the degree of Doctor of Philosophy in accordance with the regulation of the institute. To the best of our knowledge, it has not been submitted elsewhere for the award of the degree.

.....

Dr. Arnab Sarkar
Assistant Professor
Department of Computer Science and Engineering
IIT Guwahati

.....

Dr. Arijit Sur
Associate Professor
Department of Computer Science and Engineering
IIT Guwahati

Dedicated to

Ma, Papa, Bhaia, Bhabhi, Didi, Jija ji, Sajal

Whose blessing, love and inspiration paved my path of success

Acknowledgments

A lot of people have contributed to the production of this dissertation. I owe my gratitude to all those people who have made this possible.

I wish to express my deepest gratitude to my supervisors, Dr. Arnab Sarkar and Dr. Arijit Sur for their valuable guidance, inspiration, and advice. I feel very privileged to have had the opportunity to learn from, and work with them. Their constant guidance and support not only paved the way for my development as a research scientist but also changed my personality, ability, and nature in many ways. I have been fortunate to have such advisors who gave me the freedom to explore on my own and at the same time the guidance to recover when my steps faltered. Besides my advisors, I would like to thank the rest of my thesis committee members: Prof. S. V. Rao, Dr. T. Venkatesh, and Dr. Partha Sarathi Mandal, for their insightful comments and encouragement. Their comments and suggestions helped me to widen my research from various perspectives.

I would also like to express my heartful gratitude to the director, the deans and other managements of IIT Guwahati whose collective efforts has made this institute a place for world-class studies and eduaction. I am thankful to all faculty and staff of Dept. of Computer Science and Engineering for extending their co-operation in terms of technical and official support for the successful completion of my research work.

I am thankful to my friends Sibaji, Sangeet, Rajesh and Rana for supporting and motivating to overcome any problems either in work and otherwise. The countless discussions, sharing ideas has improved our research. I am also grateful to all my seniors, friends and juniors especially Mamata di, Pravati di, Shilpa di, Shirshendu bhaia,

Nil bhaia, Mayank bhaia, Senko bhaia, Anirban, Piyoosh, Deepak, Sathish, Brijesh, Mritunjay, Rahul, Basant and many others for their unconditional help and support. You made my life at IIT Guwahati a memorable experience.

Most importantly, none of this would have been possible without the love and patience of my family. I want to thank parents, Bhaia, Jija ji, Didi, Bhabhi, Sajal, Muskan, Khusi, Tanay and Tanush for being a constant source of love, concern, support, and strength all these years.

Contents

List of Figures

LIST OF FIGURES

List of Algorithms

List of Tables

List of Acronyms

AVRC *Adaptive Video Resource Controller*

CQI *Channel Quality Indicator*

DASH *Dynamic Adaptive Streaming over HTTP*

DES *Discrete Event System*

DP *Dynamic Programming*

DPQA *DP Based Quality-level Allocator*

FI *Fairness Index*

FLRA *Frame Level Resource Allocator*

GA *Genetic Algorithm*

ILP *Integer Linear Programming*

LTE *Long Term Evolution*

MAC *Media Access Control*

MDP *Markov Decision Process*

MPEG *Moving Picture Experts Group*

NRT *non-real-time*

VoIP Voice over IP

PLR *Packet Loss Rate*

QoE *Quality of Experience*

QoS *Quality of Service*

RB *Resource Block*

RT *real-time*

SCT *Supervisory Control Theory*

SDQA *Streamlined DP-based Quality-level Allocator*

SDQA-AA *SDQA Approximation Algorithm*

SE *Spectral Efficiency*

SINR *Signal-to-interference-plus-noise ratio*

SSC *Switching Stability Controller*

SVC *Scalable Video Coding*

TRC *Throttle Rate Controller*

TTI *Transmission Time Interval*

UE *User Equipment*

VBR *Variable Bit Rate*

List of Symbols

m_{ir}	Metric value for the i^{th} flow corresponding to the r^{th} RB
SE_{ir}	Spectral Efficiency for the i^{th} flow corresponding to the r^{th} RB
$HOLD$	Head of Line Delay
d_f^i	Amount of data required to transmit for the i^{th} flow in the f^{th} frame
BS_i^{th}	Buffer threshold for the i^{th} flow
BS_i^{cur}	Current playout buffer size for the i^{th} flow
R_{il}	Reward value for the i^{th} flow at the l^{th} robustness level
r_{il}	i^{th} flow's RB demand at l^{th} quality level
q_{il}	i^{th} flow's video quality at l^{th} quality level
x_i	Min quality level allowed for the i^{th} flow
y_i	Max quality level allowed for the i^{th} flow
ρS_i	List of partial solutions using atmost i flows
ρS_{il}	List of partial solutions using upto i flows - with the i^{th} flow fixed at l^{th} quality level
$\rho SN_{ip} \rightarrow \alpha$	The bound on the number of RBs for the p^{th} node in ρS_i
$\rho SN_{ip} \rightarrow q$	The quality value corresponding to the p^{th} node in ρS_i

$\rho SN_{ip} \rightarrow QL$ List of selected quality levels for the flows $f_1, f_2, \ldots, f_i$ corresponding to the p^{th} node of solution ρS_i

sg_i i^{th} video segment

$q(t)$ Playout buffer status at time t

$r(t)$ Received rate at time t

$\hat{r}(t)$ Server sending rate at time t

$b(j)$ Available bandwidth during the download of the j^{th} segment

$\hat{b}(j)$ Estimated bandwidth for the download of the j^{th} segment

q_s^+ Upper safe threshold limit

q_s^- Lower safe threshold limit ($q_s^- < q_s^+$)

q_{min} Minimum safety threshold for buffer size to avoid buffer outages

$\mathcal{S}$ Finite set of system states

$\mathcal{A}$ Finite set of actions

$\mathcal{P}_a(s, s')$ Transition probability that action a in state s will lead to state s'

$\mathcal{R}_a(s, s')$ Reward obtained after transition form s to s' by taking action a

γ Discounting factor for reward collected from future actions and states

Chapter 1

Introduction

Recent advances in wireless broadband technology have spurred an ever increasing demand for diverse data rate traffic flows with varied *QoS* requirements ranging from *real-time (RT)*, *Voice over IP (VoIP)*, streaming video/audio, online gaming etc. to soft real-time flows like telnet, web-browsing etc. and even *non-real-time (NRT)* best-effort data downloads. The proliferation of high-end mobile devices, such as smartphones, tablets and laptops have pushed the momentum further in recent years. In 2013-14, the number of mobile devices have exceeded the total number of people on earth. It is estimated that mobile data traffic will grow at a compound annual growth rate of 47%, reaching 49.0 exabytes per month by 2021 [2]. A separate study estimates that there will be a huge growth in IPv6-capable smartphones and tablets and their combined share is expected to become 73% of the total number of mobile devices by 2021 [2]. As a result, Internet over wireless is witnessing an exponential growth in the production and consumption of multimedia content of various domains including entertainment industries, news media, education, and user generated data. It is estimated that more than three-fourths of the worlds mobile data traffic will be video by 2021. Mobile video is expected to increase by 9-folds between 2016 and 2021, accounting for 78% of the total mobile data traffic by the end of the forecast period.

To meet the challenges imposed by such demands, the *LTE* [3] technology has

been introduced in 2008 and is now the de facto standard for the 4^{th} generation cellular system. The technology was designed and evolved with the determination to deliver a number of advantages over its predecessors, such as, low latency, high peak data rates, greater efficiency in the usage of the wireless spectrum and enhanced support for end-to-end *QoS*. In addition, *LTE* systems provide an elaborate packet scheduling infrastructure which if efficiently harnessed has the ability to support a wide variety of high data rate multimedia and Internet services even in high mobility scenarios. Founded on the *Orthogonal Frequency Division Multiplexing (OFDM)* mechanism, *LTE* allows its total radio resource bandwidth (BW) (upto 20 MHz according to *LTE* release 8) to be simultaneously frequency multiplexed into a fixed number of sub-channels (up to 100 sub-channels of 180 KHz each), each of which may be further time multiplexed at very fine granularities of a *Transmission Time Interval (TTI)* (1 $TTI = 1$ ms). A *TTI* is made up of two time slots of length 0.5 ms. A time/frequency radio resource spanning over one time slot in time domain and over one sub-channel in frequency domain is called a *Resource Block (RB)*[1] and corresponds to the smallest radio resource unit that may be assigned to an user equipment (UE) for data transmission. Along with this, *LTE* is equipped with features like *Channel Quality Indicator (CQI)* reporting, link adaptation through *Adaptive Modulation and Coding (AMC)* and *Hybrid Automatic Retransmission Request (HARQ)* to support better *QoS*, low latencies and improved spectral efficiency.

A pictorial bird's eye view of the cellular network system with respect to downlink radio resource allocation is presented in Figure 1.1 below. The overall system is composed of three principal components, the "*User Equipments* (*UEs*)", the "*Radio Resource Allocation (RRA)* framework" and the "traffic flows". In the recent past, there has been a proliferation of a variety of mobile devices (UEs) such as smartphones, tablet PCs, laptops etc. equipped with wireless access capabilities. They are heterogeneous in terms of *CPU* speed, *RAM* and storage, display resolution, battery capacity, and connectivity. Such heterogeneous user

[1]Basic unit of resource that may be assigned to an UE in LTE

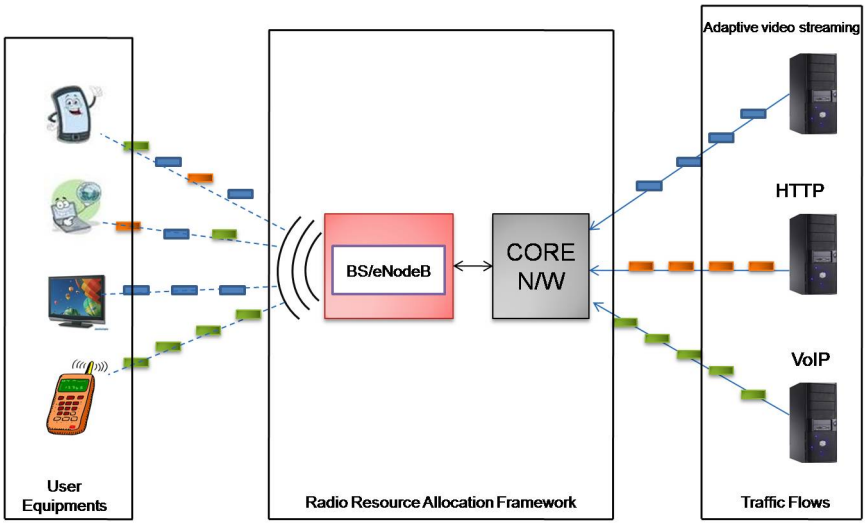


Figure 1.1: *Domain of Research*

equipments are depicted in the left of the figure. On the other hand, the right of the figure shows diverse traffic flows which the scheduling mechanism contained in the middle component must handle. Today, such traffic includes various types of stringent delay-sensitive flows such as continuous multimedia (say, *VoIP*, *RT* and streaming audio/video, online gaming etc.), soft *RT* flows like telnet and web-browsing, to even best effort data downloads. The radio resource allocation framework in the middle is further composed of "The Core Network", "The Base Station/eNodeB" and "The Wireless Propagation Medium". The core network receives traffic flows from the external world (w.r.t downlink transmission) along with their *QoS*/*QoE* specifications and forwards these flows to the base station/eNodeB. The eNodeB does the actual work of multiplexing and allocating the wireless spectrum (scheduling) in both time and frequency among the received flows. The wireless propagation medium is subject to high variability in both time and frequency domains due to several causes such as user mobility (Doppler effect) and network dynamics (fading effects, multi-path propagation, inter and intra cell interference and so on).

The job of the scheduler is thus to appropriately allocate radio resources such

that the *QoS*/*QoE* demands of all flows/end users are met while simultaneously satisfying other practical constraints/objectives like limited power budget, maximizing spectral efficiency in the face of ever changing user mobility and network dynamics, graceful degradation in times of transient overloads etc. Dynamic resource allocation for multiuser wireless networks has attracted a lot of interest [4–8] in the recent past.

Typical downlink scheduling mechanisms like *Maximum Throughput (MT)*, *Proportional Fair (PF)* [9] etc. are quality of service unaware. Hence, they may not directly be applicable for multimedia applications. However, most of the recent *QoS* aware schemes [10] employ them to better cell spectral efficiencies and fairness between flows. Many schemes that attempt to handle delay sensitive applications have been proposed in [11–13]. Among them, two notable algorithms namely, the *LOG*-rule [11] and *EXP*-rule [12] consider all active flows (including both RT and non-RT flows) together at every *TTI* and select flows based on their individual metric values. Piro et. al. in [14] proposed a frame-based two level packet scheduling algorithm (called *FLS*) where the first level computes the amount of data to be transmitted by each RT flow (such that their delay constraints may be satisfied) in the next *LTE* frame (a fixed time interval of duration 10 ms) using discrete time linear control law. For non-RT flows, the metric values for both schedulers are obtained based on a proportional fair policy [9].

As video based applications are expected to generate a majority of the wireless traffic in the recent future, devising radio resource allocation methodologies which are specifically tuned to provide effective *QoE* to all end users over a limited bandwidth, has become a research topic of immense importance. Research towards *QoE* aware resource adaptation strategies has primarily progressed in two distinct categories. The first class of solutions [15,16] embed video bit-rate/playback quality adaptation within *client/User Equipment (UE) applications* taking care of network and device dynamics. The second class provides *in-network solutions* [17] and are usually oblivious towards client application requirements. The typical objective of these solution approaches is to maximize system-level *QoE*

corresponding to a set of clients under varying network conditions and video bit-rate demands. It is evident that a purely client-driven adaptation methodology as proposed by the first stream is ignorant of instantaneous bandwidth availability as well as demands of the overall system (consisting of all clients taken together) and hence may be prone to unacceptably high bit-rate/quality oscillations and sub-optimal adaptation decisions due to the lack of an integrated system-level view [18]. An important shortcoming of completely in-network solutions is that it cannot keep track of the buffer status of individual clients which may lead to playback interruptions due to buffer outages. The above discussion indicates that both fully in-network as well as purely client-side approaches cannot comprehensively handle all the parameters that are necessary to deliver high system-level *QoE* to all clients. However, literature [19–21] reveals that there is a dearth of resource allocation strategies which can effectively integrate in-network and client-side techniques to provide satisfactory *QoE* to all clients in the system.

This research intends to investigate into the theoretical and practical aspects of scheduling mechanisms for downlink radio resource allocation and develop new efficient *QoS*/*QoE* aware in-network/client-side downlink scheduling strategies suited to today's wireless network demands as mentioned above.

1.1 Challenges

A scheduling mechanism which efficiently caters to diverse applications and serves the variety of user equipments in today's wireless systems, must meet several challenges. We now enumerate a few such important challenges and discuss them.

1. **Satisfying end-to-end delay bounds:**
   Current user equipments support various applications like VoIP, streaming video, web-browsing, downloads etc. with varying degrees of timeliness constraints or *QoS* demands. For example, for real-time video streams, the scheduler may need to transmit at-least 25 frames per second for high quality stutter free video playback. On the other hand, although such a

strict rate requirement is not essential for soft real-time services like web-browsing, there must be a tolerable upper bound to the maximum allowable delay.

2. **Low scheduling overhead:**
   Resources have to be allocated at every TTI over numerous frequency defined channels and each such flow-to-channel mapping must be done optimizing numerous objectives and satisfying several constraints. For example, in *Long Term Evolution (4G) systems*, scheduling decisions must be taken at time granularities of the order of 1 ms ($TTI = 1$ ms) and at each such scheduling event, up to 100 different channels may need to be allocated to the traffic flows. This necessitates efficient but low overhead schedulers to maintain processing requirements at the base station/eNodeB within a reasonable limits.

3. **Delivering satisfactory quality of experience to the end-user:**
   Meeting the ultimate objective of delivering all clients/User Equipments (UEs) with high quality QoE pivots around several key factors including: (i) Transmitting all video streams with satisfactory quality over a limited and temporally varying wireless bandwidth, (ii) Minimizing buffer outage at the UE in order to guarantee seamless video viewing experience and (iii) Stabilizing bit-rate switches to avoid user annoyance due to flickering.

1.2 Motivation and Objectives

In spite of progressively increasing data transmission capacities over wireless, effective allocation of radio resources to diverse latency sensitive applications is set to remain an daunting problem in LTE and other futuristic mobile networks. This is because, as discussed above, delivering satisfactory QoS/QoE to a set of multimedia application over a network with unpredictable and time varying capacities is a very challenging problem and demands extensive research. Although, the

designed scheduling algorithms need to have low and controllable overheads, it is essential to carefully and simultaneously consider all the above factors to provide a seamless viewing experience to end-users while avoiding transient overloads.

The principal aim of this dissertation has been to investigate the theoretical and practical aspects of in-network/client-side radio resource allocation strategies keeping in view the challenges/hurdles discussed in the previous section. In particular, the objectives of this work may be summarized as follows:

1. Design and implementation of a novel low overhead downlink scheduling frameworks for heterogeneous *RT Variable Bit Rate (VBR)* traffic over *LTE* networks.

2. Extending the generic *LTE* downlink scheduling framework mentioned in objective 1 to tailor it towards effective streaming video management with *QoE* control mechanisms such as buffer awareness and bit-rate adaptability during transient overloads.

3. Design and implementation of a *Dynamic Adaptive Streaming over HTTP (DASH)* based client-side aware in-network adaptive streaming architecture whose objective is to maximize aggregate *QoE* over all downlink flows by simultaneously balancing important *QoE* verticals.

4. Development of client-side bit-rate adaptation strategies for adaptive video flows in general, and *Scalable Video Coded (SVC) flows* in particular.

1.3 Contributions

As a part of the research work, six adaptive resource allocation strategies have been designed.

1. **A Three Level LTE Downlink Scheduling Framework for RT VBR Traffic**

   The first work presents a flexible downlink resource allocation framework

for *LTE* systems. The framework is aimed at enabling mobile operators to effectively achieve good *QoS* and cell spectral efficiencies while incurring low overall scheduling overheads.

2. **Hybrid Offline-Online Approach to Downlink Resource Allocation Over LTE**

   As a spin-off from the previous work, we have designed a hybrid offline-online version of the three level scheduler whose objective is to lower on-line scheduling complexity.

3. **Buffer Aware Resource Allocation for Video Streaming Over LTE**

   Next, the basic *TLS* framework is extended to support video streaming services. The objective of this work is to minimize stutters in the video output by imbibing client-side buffer awareness in the downlink scheduling framework.

4. **A Resource Allocation Framework for Adaptive Video Streaming Over LTE**

   Delivering high overall playback quality to all end-users is not possible only through a client oblivious dynamic bit-rate adaptation mechanism. This work thus proposes a client status aware in-network adaptive video streaming architecture that attempts to improve delivered *QoE* through a judicious balance between various *QoE* parameters including: (i) Stutter-free video playout, (ii) Startup delay, (iii) Video quality and (iv) Stability against bit rate switches.

   While all the previous contributions were primarily in-network downlink schedulers which reside at *eNodeB* of an *LTE* network, the last two resource allocation strategies are client-side video streaming management mechanisms which are implemented within a client's video playback application.

5. **Client-side QoE Control for SVC-DASH Video Streaming: A DES Supported Design Approach**

DASH is becoming the de facto technology for live and on-demand video streaming services. In this work, we present the step-by-step design of a client-side *Video Quality Adaptation Unit (VQAU)* whose behavior is formally represented by a *Discrete Event System (DES)*. The objective of this work is to maximize the *QoE* of an end user by simultaneously balancing all the important *QoE* verticals mentioned above.

6. **An Efficient QoE Aware MDP Based Client-side Agent for Video Adaptation in Cellular Networks**
   In this work, we have formulated the problem of rate adaptation of *SVC-DASH* videos as a *Markov Decision Process (MDP)* because of its salience in taking optimal decisions under uncertainty. The proposed adaptation agent learns and determines one among a set of available actions at each state based on iterative interactions with its environment. The possible actions and associated reward values are appropriately defined for each state of *MDP* in order to achieve smooth video viewing experience.

We now provide a more detailed overview of each of the above contributions.

1.3.1 A Three Level LTE Downlink Scheduling Framework for RT VBR Traffic

Three Level Scheduler (TLS) is a flexible resource allocation framework aimed at enabling mobile operators to effectively achieve good *QoS* and high cell spectral efficiency while incurring low overall scheduling overheads. The *TLS* framework has been designed as a three layered architecture (which consists of the super-frame, the frame layer and the *TTI* layer). These layers interact together in order to dynamically allocate *RBs* to the active flows, taking into account the instantaneous channel states, system load and maximum tolerable delays of *RT* flows.

At the outermost layer, *TLS* divides time into a sequence of fixed sized intervals called super-frames. At each super-frame boundary, the *Traffic Prediction*

Module calculates the amount of data that the *RT* flows should transmit in the following super-frame in order to satisfy their delay constraints. The *System Overload Handler* is also embedded at this layer and imposes an upper cap on the transmission data rates (during system overload) for each flow such that the total data rate demand may be reduced to atmost the total available cell capacity (computed in the *Cell Capacity Prediction Module*).

A super-frame is further divided into a specific number of frames whose duration is dynamically adjusted at each super-frame boundary using a procedure called *Dynamic Frame Size Adjustment* such that a required scheduling accuracy may be obtained while keeping *RB* selection overheads under control. The *Frame Level Resource Allocator (FLRA)* at each frame boundary allocates the required number of RBs to a selected set of flow so that their data rate demand may be satisfied after execution within the next frame. *FLRA* selects the available traffic flows (*RT* flows having the highest priority, followed by bursty *RT* flows and *NRT* flows) in three distinct rounds. The frame level scheduler is called at each frame boundary within a super-frame in order to satisfy the data rate demands of flows within each frame. All RBs within a given frame duration are statically allocated by the *FLRA* module which selects the available traffic flows (*RT* flows having the highest priority, followed by bursty *RT* flows and *NRT* flows) in three distinct rounds. Finally, at the third and innermost level, the RBs are physically allocated to the flows for transmission at each *TTI* through an $O(1)$ lookup operation on the *RB Allocation Matrix* obtained from the *FLRA* module. During such *RB* allocation at any given *TTI*, an appropriate *Modulation and Coding Scheme (MCS)* is selected based on the instantaneous *CQI* feedback received for a flow-RB pair at that *TTI*.

Simulation results clearly reveal that TLS outperforms a few important state-of-the-art schedulers such as *New Two Level Scheduler* [22], *LOG-Rule* [11], *EXP-Rule* [12], *EXP-VT-SH* [13], and *FLS* [14] in most scenarios, and points to the general efficacy of the proposed framework.

1.3.2 Hybrid Offline-Online Approach to Downlink Resource Allocation Over LTE

As a spin-off from the previous work, we have designed a hybrid offline-online version of the three level scheduler whose objective is to lower on-line scheduling complexity. The offline strategy tentatively selects an appropriate subset of flows so that system loads do not cross an overload threshold, the online strategy allocates *Resource Blocks (RBs)* to selected flows so that *QoS* may be maximized. As the complexity of the offline component is independent of the scheduling complexity, we have adopted an elaborate, formal and correct-by-construction design mechanism for the offline component based on *Supervisory Control Theory* of *Discrete Event Systems* (*Supervisory Control Theory (SCT)* of *DES*).

As mentioned above, the objective of the offline phase is to select the appropriate number of flows for *RB* allocation at each *TTI* such that the system load within the *TTI* is assured to remain within a given safe threshold limit. Such a selection mechanism must consider the expected instantaneous load and capacity as well as the urgency of the flows at the *TTI* under consideration. A good solution to the problem should exhaustively enumerate and analyze the effect of the above parameters under all possible scenarios so that the online *RB* allocator can generate schedules which provide high *QoS* by judiciously considering all parameters. In this work, we employ *SCT* of *DES* to synthesize a supervisor which can control the number of flows to be considered for *RB* allocation at a given *TTI*.

It may be noted that the offline solution always provides a conservative bound on the number of flows to be selected such that the system never transits into overloads. This solution can be bettered through the online consideration of the exact actual values of parameters such as *RBs* demands, instantaneous sub-band channel conditions etc. Therefore, in order to capture the inherent variability in the system, an online resource allocation strategy has been proposed. This online strategy runs on top of the offline policy and effectively tunes the offline decisions (if needed) at each *TTI* such that delivered *QoS* and resource utilization

is maximized.

1.3.3 Buffer-Aware Resource Allocation for Video Streaming over LTE

In this work, we have proposed a downlink resource allocation framework called *Buffer-Aware Three Level Scheduler* (*BA-TLS*) which endeavors to deliver smooth viewing experience to each active end user even during transient network overloads. Founded on the basic architecture of *Three Level Scheduler* (*TLS*), *BA-TLS* inherits all its salient facets including low *RB* allocation overheads, minimum guaranteed delay bounds for the flows, high spectral efficiency etc. However, *TLS* does not consider client side buffer status in its *RB* allocation strategy. Due to this buffer status ignorance, basic *TLS* is susceptible to frequent buffering events which may induce stutters in the transmitted videos which ultimately pulls down the *quality of viewing experience*. One of the principal objectives of *BA-TLS* is to mitigate this problem by minimizing re-buffering events caused by client side buffer outages. It may be noted that seamless playout experience in the face of temporally varying wireless bandwidths, may only be ensured by continuously maintaining playout buffer size above a specific threshold [23]. The robustness of a flow against buffer outages is directly proportional to the length of the video playout duration that can be transmitted in the ensuing super-frame (higher this length, higher will be the number of video frames contained in the buffer's repository). However at any given time, the exact video playout length required to quantitatively achieve a specific degree of robustness depends on the instantaneous channel quality being experienced. This is because, flows encountering poor channel qualities typically tend to suffer higher packet loss rates which in turn effects increased packet retransmissions. Thus, in such a situation, buffer outages may only possibly be avoided by maintaining a relatively larger number of video frames in the playout buffer. This problem has been posed as an optimization problem with constraints based formulation such that aggregate robustness against buffer outages for a given set of flows over a limited wireless

bandwidth, is maximized. The optimal *DP* solution for the optimization problem has been shown to incur substantially high computational overheads which makes it impractical for it to be employed as an on-line resource allocation policy in dynamically evolving networks. In order to control on-line complexity, we have first proposed an efficient genetic algorithm [24] based stochastic solution strategy. Next, we have designed a deterministic heuristic strategy known as *Proportionally Balanced Robustness-level Allocator (PBRA)*, which is able to achieve robustness levels which are comparable to the stochastic solution, while incurring drastically lower computational overheads. Experimental results show that *PBRA* is about *300* to *600* times faster on average compared to the genetic algorithm approach.

1.3.4 A Resource Allocation Framework for Adaptive Video Streaming Over LTE

In this work, we present a novel client-status aware in-network radio resource scheduling strategy for *LTE* networks that is able to provide satisfactory *QoE* to all *UEs* in the face of dynamic variations in channel qualities, client-side buffer status and data-rate demands of video flows. The proposed scheduling framework called *Adaptive Video Streaming Architecture (AVSA)* operates at the temporal granularity of a duration called adaptation interval (which typically varies from a few secs to tens of secs). *AVSA* primarily operates using three controllers: (i) *Throttle Rate Controller (TRC)*, (ii) *Switching Stability Controller (SSC)* and (iii) *Adaptive Video Resource Controller (AVRC)*. *TRC* attempts to maintain stable buffer sizes for each client by appropriately throttling transmission rates of its video flow from the server. *SSC* on the other hand, endeavours to control and bound the encoding quality/bit-rate switches between consecutive video segments corresponding to each flow. Given the throttled data-rate (obtained from TRC) and allowed switching (obtained from *SSC*) for each flow at any adaptation interval boundary, *AVRC* selects appropriate bit-rates/quality levels at which the flows should be transmitted in the ensuing adaptation interval such that the total

bandwidth demand of all flows approximately equals the expected cell capacity.

However, selecting appropriate data-rates and efficiently multiplexing the available bandwidth among clients in both frequency and time in order to maximize *QoE* is a complex online scheduling problem. We first pose this problem as an *Integer Linear Programming (ILP)* formulation and solve through a conventional *DP* strategy. However, conventional *DP* proves to be prohibitively expensive in terms of online computational overheads. Our experimental results show that given an *LTE* bandwidth of 20 MHz in a system with 50 active users, conventional *DP* takes 9.32 ms ($\sim$ 19 times the size of resource block) on average to generate a solution even for a moderate adaptation interval size of 1 sec on a 2.5 GHz computing core. Therefore, in order to design AVRC as an effective online mechanism which can be implemented with moderate computational resources and applied at each adaptation interval boundary, we have modified conventional *DP* and devised a new strategy known as *Streamlined DP-based Quality-level Allocator (SDQA)*. *SDQA* intelligently leverages the discrete nature in the data-rate scalability of video flows to retain a far lower number of non-dominating partial *DP*-solutions and allows the ultimate optimal solution to be generated much quicker. Further, we propose a tunable approximation scheme called *SDQA Approximation Algorithm (SDQA-AA)* that may be employed to accelerate the speed of generating solutions (or limit necessary computational resources) by various optional degrees with distinct bounds on the degradation in solution quality.

1.3.5 Client-side QoE Control for SVC-DASH Video Streaming: A DES Supported Design Approach

In this work, we present the step-by-step design of a client-side *Video Quality Adaptation Unit (VQAU)* whose behavior is formally represented by a *DES*. A *DES* supported design approach has allowed us to accurately model the discrete event dynamics corresponding to the *DASH* based video streaming control framework considered in this work. In particular, *DES* has helped in precisely modelling relevant system components and their interactions under the influence of a set of

events that are active at a given time. The principal objective of *DES* supported video quality adaptation unit is to maximize the quality of video viewing experience of an end user by dynamically taking one of the following two actions at any given state: (i) download a new segment at a selected enhancement level or, (ii) upgrade (smooth-out) the enhancement level of an already downloaded segment in the playout buffer by a stipulated value.

In order to avoid playback interruptions even under fluctuating bandwidth conditions, *VQAU* performs upgradation/smoothing over downloaded video segments only when the playout buffer size attains an upper safety threshold (q_s^+ in seconds). Once q_s^+ is reached, *VQAU* continues smoothing for a fixed time interval T without downloading any new segment until the buffer size reduces to a lower threshold q_s^- (i.e., $T = q_s^+ - q_s^-$). This smoothing action has a two-fold objective: (i) improve overall playback encoding qualities through enhancement level upgradation and (ii) smooth-out bit-rate switching (difference in enhancement levels) between consecutive segments to reduce flickering. In this work, we have posed the problem of smoothing a set of eligible segments in the buffer (given, time constraint T) as an optimization problem. This problem has been solved through a novel low-overhead variant of the conventional *DP* approach which we have named as the *Streamlined Enhancement-level Smoother (SES)* strategy.

In short, the salient facets of the designed *VQAU* adaptation framework may be listed as follows:

- The framework effectively maximizes *QoE* by selecting satisfactory encoding qualities for video segments while simultaneously avoiding buffer outages and restricting bit-rate switches as far as possible.

- The designed *SES* algorithm provides a low-overhead (compared to conventional *DP*) optimal smoothing strategy for a given set of segments in the buffer and a smoothing interval T.

- We have shown through simulation based experiments that the proposed

framework is able to outperform a significant state-of-the-art dynamic adaptive streaming technique presented in [25].

- *VQAU* being modeled as a *DES* is modular and scalable. This makes it flexible towards *easy modification/reconfiguration* in case of changes in design policy.

1.3.6 An Efficient QoE Aware MDP Based Client-side Agent for Video Adaptation in Cellular Networks

Several rate adaptation strategies based on recently observed link bandwidths and instantaneous buffer sizes [25, 26] have been proposed in the recent past. These solutions perform reasonably well, but are not able to provide an optimal trade-off between the various *QoE* verticals like video playback quality, video bit-rate/enhancement level switching and playout buffer size, especially in unpredictable cellular networks. The primary objective of this work is to enable incorporation of the influence of future segments in the rate adaptation strategy so that the system can achieve an optimal trade-off between the different *QoE* metrices in cellular environments. In this thesis, we have formulated the rate adaptation of *SVC-DASH* as a *MDP* due to its ability to take optimal decisions under uncertainty [27]. The adaptation agent learns and determines one of the available actions based on iterative interactions with its environment. In order to measure the effectiveness of an action, the environment provides a numeric reward value to the agent for each action. Based on the received reward, the agent gradually learns the optimal action to be taken at a specific system state. Various actions and reward values are defined for each state of *MDP* in order to achieve smooth video viewing experience. Experimental results using *LTE-Sim*, a standard simulator for *LTE* networks, are promising.

1.4 Organization of the Thesis

The thesis is organized into seven chapters. A summary of the contents in each chapter is as follows:

- **Chapter 2** begins by presenting a background on the evolution and architecture of cellular systems in general and *LTE* based systems in particular. This is followed by a discussion on the fundamental and traditional resource allocation policies that have been developed for these systems. Then, the chapter presents a survey of various non-adaptive and adaptive in-network downlink mechanisms with a special emphasis on video flow scheduling approaches. Finally, the chapter provides a state-of-the-art review on client-side video adaptation techniques.

- **Chapter 3** discuss two low overhead *LTE* downlink scheduling frameworks for RT VBR traffic. The frameworks are aimed at enabling mobile operators to effectively achieve good *QoS* and cell spectral efficiencies while incurring low overall scheduling overheads.

- **Chapter 4** presents an extension of the *Three Level Scheduling* framework (called *BA-TLS*) to support video streaming services. The objective of this work is to minimize stutters in the video output by imbibing client-side buffer awareness in the downlink scheduling framework.

- **Chapter 5** deals with the design of in-network adaptive video streaming architecture. The objective of the designed architectures is to deliver satisfactory video viewing experience for all end-users even during transient network overloads.

- **Chapter 6** proposes efficient *QoE* aware client-side streaming management mechanisms which are implemented within a client's video playback application. The main objective of the works presented in this chapter is to utilize the power of *SVC-DASH* to its fullest so that overall encoding quality of

the video output is maximized while maintaining playout buffer sizes above a safe threshold and restricting the degree of switching as far as possible.

- **Chapter 7** concludes this thesis. We discuss the work in progress, possible extensions and future work that can be done in this area.

Chapter 2

Resource Allocation Strategies in Cellular Networks: Background, Flavours, Trends

This dissertation is oriented towards the design of resource allocation strategies for multimedia services over cellular networks in general and *LTE* systems in particular. The previous chapter provided a view of the challenges imposed by the diversity in the types of network traffic, heterogeneity in user equipment capabilities, uncertainties due to temporal variations in the availability of scarce radio resources etc., towards *QoS*/*QoE* aware scheduling of multimedia flows.

In this chapter, we first provide an overview on the evolution of cellular networks. Then, the chapter presents the architecture of *LTE* with a particular emphasis towards radio resource management. Subsequently, we present a survey of various resource allocation strategies in *LTE*. The chapter concludes by presenting a review of various in-network/client-side scheduling strategies in *LTE* Networks.

2.1 Evolution of Cellular Networks

All over the world, cellular communication services have witnessed a phenomenal growth over the last few decades. The evolution of mobile cellular technology has been categorized into various generations. During the past few decades, cellular

networks have witnessed 4 or 5 generations of technology evolution, namely, from 1G to 4G.

The era of cellular network communications may be considered to have begun way back in the early 1980s through the deployment of the *first generation (1G)* system invented by AT&T's Bell labs about a decade earlier. 1G technology which used analogue transmission for speech services came in three principal verities, namely, *Nordic Mobile Telephones (NMT)*, *Total Access Communication Systems (TACS)* and *Advanced Mobile Phone Service (AMPS)*. *NMT* was deployed in the northern European region. *TACS* was mainly deployed in England, Ireland and Japan. In the United States, the first commercial deployment of cellular telephony was based on *AMPS* and done in late 1983 by Ameritech. It may be noted that the different systems could not communicate among each other at that time. Only 20 million people worldwide, which was less than 1% of the global population, ever used the first generation system.

Second generation (2G) mobile phone systems emerged in the 1990's. There were four different 2G systems worldwide, namely, *Global System for Mobile communication (GSM)*, *Code Division Multiple Access (CDMA)*, *US-CDMA*, *Personal Digital Cellular (PDC)*. These *2G* systems differed from the previous generation in their use of digital transmission instead of analogue transmission. 2G systems offered higher spectrum efficiency, better data services, and more advanced roaming facility. In addition to providing improved voice quality, capacity and security, 2G cellular systems also enabled a few new applications, the most popular among which was the *Short Message Service (SMS)*, first deployed in Europe in 1991. The *SMS* application proved to be incredibly successful from a commercial standpoint, so much so that in some networks *SMSs* constituted a major part of the total traffic. In order to enable data transmission on the air-interface, *GSM Packet Radio Systems (GPRS)* (also known as 2.5G) was introduced and this proved to be an evolutionary step for *GSM* towards high data rates. *GPRS* and *GSM* shared the same frequency bands, time slots, and signaling links. *GPRS* supported flexible data transmission rates as well as continuous

connection to the network. Typical implementations of *GPRS* provided user data rates of 20-40kbps. *GPRS* may be considered to be the most significant step towards *third generation (3G)*. The *GSM* standard got a further boost in its data handling capabilities with the introduction of *Enhanced Data Rate for GSM Evolution (EDGE)*, in the early part of 1997. It allowed the clear and fast transmission of data and information up to 384kbps speed.

In the beginning of the new millennium, people began to use mobile phones in their daily lives and therefore, service providers experienced exponential growth in the use of 2G phones. In order to meet such demands, the researchers and industry started to work on the next generation of technology, known as 3G. The design objectives of 3G systems were to deliver much higher data rates, provide significant increment in voice capacity, and support advanced services and applications, including multimedia. The *International Telecommunication Union (ITU)* formalized the objectives for 3G mobile networks with the IMT-2000 standard. *3rd Generation Partnership Project (3GPP)*, a mobile communications industry collaboration, has continued that work by defining a mobile system that fulfills the IMT-2000 standard. In Europe it was called *UMTS (Universal Terrestrial Mobile System)*, which is *European Telecommunications Standards Institute (ETSI)* driven. 3G networks enabled service providers to offer services which include voice telephony, video calls, and broadband wireless data, all in a mobile environment. In the mid 2000's, first implementations of an evolution of the 3G technology called *High Speed Downlink Packet Access (HSDPA)* was begun. It is an enhanced 3G mobile telephony communications protocol in the *High-Speed Packet Access (HSPA)* family, also named as 3.5G, 3G+ or turbo 3G, which allows networks based on *UMTS* to have higher data transfer speeds and capacity. Initially, *HSDPA* deployments supported downlink speeds of 1.8, 3.6, 7.2 and 14.0 Mbps. Further speed enhancements were available with HSPA+, which provided speeds of up to 42 Mbps (and up to 84 Mbps with Release 9 of the 3GPP standard).

The ever increasing demands of *QoS* sensitive multimedia applications and

emergence of new technology in the mobile communication system triggered the researcher community and industry to come up with data-optimized forth generation (4G) technologies. One of the major technological enhancement in 4G is the shift from circuit-switched networks (3G) to an all-IP network. An all IP-based 4G wireless network has intrinsic advantages over its predecessors. The design goal of 4G technology was to deliver a new level of experience to the users in which they have freedom to select any services with desired *QoS* levels at affordable prices, anywhere, anytime. *Wimax* and *LTE* were the first two commercially available 4G technologies deployed in Scandinavia by TeliaSonera. Peak download and upload data transfer speeds of 4G *LTE* can reach up to 100Mbps and 50Mbps, respectively. On the other hand, *WiMAX* offers peak data rates of 128 Mbps in the downlink and 56 Mbps in the uplink.

2.2 LTE Architecture

LTE [28] represents a major step in mobile radio communications due to its shift from circuit-switched networks to an all-IP network. A typical *LTE* network architecture is shown in Figure 2.1. It consists of two parts, namely the *Core Network* (*CN*) and the *Radio Access Network* (*RAN*). Core Network (which includes the serving gateway, Mobility management and the *Packet Data Network* (*PDN*) gateway) provides functionalities like IP connectivity, authentication, authorization, accounting, handling and routing of traffics to and from several base stations etc. *RAN* is mainly composed of eNodeB where *Radio Resource Management* (*RRM*), interference migration, and handover initiations are performed.

In *LTE*, exchange of data and control information between eNodeB and *User Equipment (UE)* is typically occur through the physical channel [29]. The *LTE* physical channel is actually composed of two radio links, namely uplink (radio channel from *UE* to eNodeB) and downlink (radio channel from eNodeB to *UE*). *LTE* is expected to cater to the high data rates (peak data rates of 100 Mbps in the downlink and 50Mbps in the uplink with 20 MHz of bandwidth in the

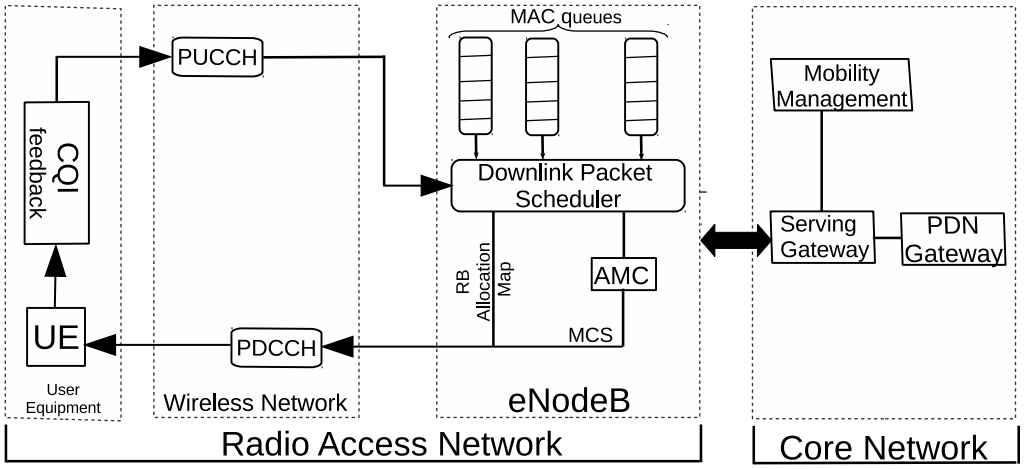


Figure 2.1: *Overview of an LTE system*

3GPP Release 8 physical layer [30]) and low latency demands of multimedia applications. To accomplish these objectives, *LTE* systems provide separate packet scheduling infrastructure for both the links, namely, downlink packet scheduler and uplink packet scheduler, in the *Media Access Control (MAC)* layer of eNodeB [31]. In order to achieve lower *Peak-to-Average Power Ratio* (*PAPR*) for the uplink channel, the uplink packet scheduler is constraint to select RBs corresponding to a single *UE* only from consecutive sub-channels in a continuous manner [32]. This limitation significantly reduces the flexibility in the resource allocation, particularly when compared to downlink packet scheduling. On the other hand, flexibility of the downlink packet scheduler motivates us to implement a downlink resource allocation framework which has the ability to efficiently allocate RBs among *RT VBR* traffics, considering their channel conditions and *QoS* requirements.

In *LTE*, radio resources are allocated in both frequency and time domains. In the frequency domain, the total bandwidth is frequency multiplexed into sub-channels of 180 kHz each. *LTE* supports various values for total bandwidth, including 1.4, 3, 5, 10, 15 or 20 MHz, and this bandwidth is multiplexed into 6, 15, 25, 50, 75 or 100 sub-channels, respectively. Each sub-channel is further time

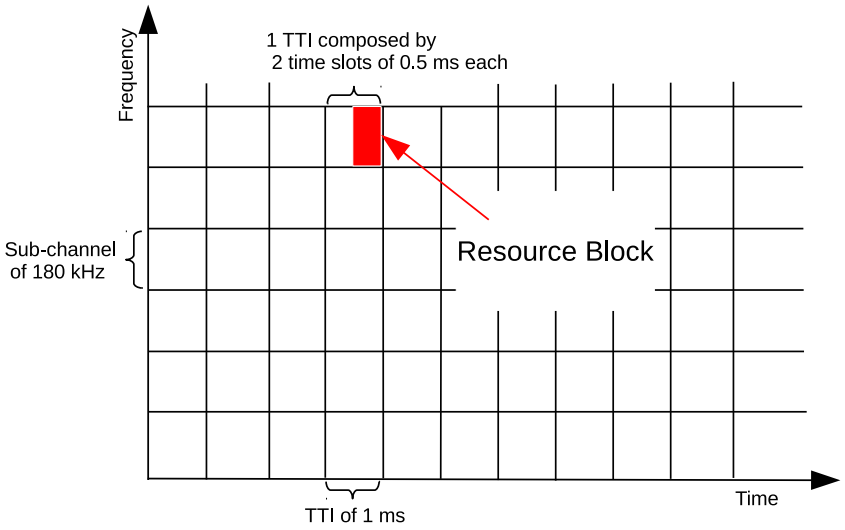


Figure 2.2: *Combined time-frequency multiplexed resource abstraction in LTE*

multiplexed into 1 ms long intervals called *TTI*. A *TTI* is composed of two time slots of length 0.5 ms each. A time/frequency radio resource spanning over one time slot in the time domain and over one sub-channel in the frequency domain is called a *Resource Block* (*RB*) [33] as shown in Figure 2.2 and corresponds to the smallest radio resource unit that may be assigned to an *UE* for data transmission. Resource allocation for each *UE* is usually based on the comparison of per-*RB* metrics. The r^{th} *RB* block is allocated to the i^{th} user if metric m_{ir} is maximum i.e. if it satisfy the equation:

$$m_{ir} = \max_{j} \{m_{jr}\} \tag{2.1}$$

The value of metric for a user depends on the resource allocation objectives.

At the physical layer, it is assumed that the total available transmission power (equal to 43 dBm) at eNodeB is uniformly spread over all the available sub-channels. Each *UE* estimates the *Signal-to-interference-plus-noise ratio (SINR)* of the received reference signals for all downlink sub-channels [34]. The estimated *SINR* values are then mapped to a corresponding set of *CQI* feedbacks (integers in the range 1 to 15) and forwarded to eNodeB periodically, using the *Physical Uplink Control Channel* (*PUCCH*) as shown in Figure 2.1. The *Physical Uplink Shared*

Channel (*PUSCH*) carries data and signalling messages from the *Uplink Shared Channel* (*UL-SCH*; this channel belongs to the set of transport channels) and can sometimes also carry uplink control information. The *Physical Random Access Channel* (*PRACH*) carries random access transmission from the *Random Access Channel* (*RACH*; this channel belongs to the set of transport channels) [35]. A *CQI* represents the quantized version of a corresponding *SINR* such that a certain maximum *Block Error Rate* (*BLER*) may be guaranteed for downlink data transmission (the default value is 10%) [36].

As shown in Figure 2.1, the downlink packet scheduler which is available at eNodeB is responsible for allocating *RBs* to an active flow in each *TTI*. For each scheduled flow, the *Adaptive Modulation and Coding* (*AMC*) module selects a proper *Modulation and Coding Scheme (MCS)* based on the *CQI* feedback. The information about the allocated *RBs* and the selected *MCS* are transmitted to the UEs on the *Physical Downlink Control Channel (PDCCH)* [10]. However, the *Physical Downlink Shared Channel (PDSCH)* is the main downlink data bearing channel which is dynamically multiplexed in frequency and time among the user equipments [33]. The *Transport Block Size (TBS)*, or in other words, the amount of data that a flow can transmit at the *MAC* layer during a *TTI* using a sub-channel, is obtained from the selected *MCS*, taking into account the physical configuration proposed in [37]. For each active flow, eNodeB maintains a buffer (queue) as the packet container for the flow. Each packet of a flow is time stamped as it arrives into the queue and subsequently transmitted through the wireless channel using the first-in first-out (FIFO) principle. At each *TTI*, delays (waiting time of a packet after arrival) for all packets at eNodeB are updated. An *RT* packet (for the i^{th} flow say) is dropped from the *MAC* queue if the packet scheduler fails to transmit it within a specified stipulated delay (denoted by $maxdelay_i$) and this effects a packet loss.

We now present an overview of various radio resource management strategies that have been used in cellular environments.

2.3 Resource Allocation and Management

Recent advances in wireless broadband technology have spurred an ever increasing demand for diverse data rate traffic flows with varied *QoS* requirements ranging from *RT* and streaming video/audio, online gaming, telepresence etc. to soft real-time flows like telnet, web-browsing etc. and even non-real-time (*NRT*) best-effort data downloads. Meeting such diverse *QoS* demands of a given set of flows to be transmitted pose considerable challenges on the resource allocation frameworks in current and futuristic cellular networks.

Effective radio resource management is an important tool by which the service provider tunes the amount of bandwidth resources received by different flows such that their *QoS* demands can be satisfied. Significant performance gains may possibly be achieved by efficiently multiplexing the total available wireless bandwidth in both frequency and time among the different user equipments. The scheduling schemes can be classified into multiple groups of strategies based on their input parameters, objectives, and service targets. Here, we have considered and classified the important resource allocation policies available in literature into the following distinct categories: (i) In-network channel-unaware/*QoS*-unaware, (ii) In-network channel-aware/QoS-unaware and (iii) In-network channel-aware/*QoS*-aware strategies. A significant component of this dissertation deals with scheduling mechanisms for adaptive video flows in particular. Hence, we have separately presented a state-of-the-art survey of (i) In-network channel-aware and *QoE* aware resource allocation schemes for adaptive video flows, and (ii) Client-side network and device aware video flow adaptation methodologies.

2.3.1 In-network Channel-unaware/QoS-unaware Strategies

These strategies were traditionally implemented in wired networks assuming time-invariant and error-free transmission media. Although there direct implementation in the cellular network is not realistic, they are typically employed jointly

with the channel-aware approaches to improve the system performance. Following are a few important and well acknowledged channel-unaware/*QoS*-unaware strategies:

First-in, first-out (FIFO)

FIFO is the simplest resource allocation strategy which schedules flows according to the order of resource requests. The metric value for the i^{th} flow over the r^{th} *RB* is expressed as:

$$m_{ir} = t - T_i \tag{2.2}$$

where, t is the current time and T_i is the time instant when request was issued by the i^{th} flow. Although this technique is simple, it is inefficient and unfair to the user.

Round Robin (RR)

The objective of this strategy is to allocate fair share of time resource to all users. Metric calculation for Round Robin is similar to *FIFO* with the following difference: T_i now refers to the last time instant at which the i^{th} user was served. Thus here, the concept of fairness is related to providing equal transmission opportunities to a given set of competing flows over time. The strategy neither guarantees fairness in the throughputs achieved by these flows, nor does it ensure fairness in received bandwidths based on the flows' resource demands.

Blind Equal Throughput (BET)

Throughput fairness among users can be achieved by *Blind Equal Throughput (BET)* strategy. In order to maintain throughput fairness, *BET* keeps account of the past average throughput achieved by each user and uses it in the metric calculation. Metric value for the i^{th} flow over the r^{th} *RB* is calculated as:

$$m_{ir}^{BET} = \frac{1}{\overline{R_i}(t-1)} \tag{2.3}$$

where, $\overline{R_i}(t-1)$ is running average throughput for the i^{th} flow and updated in every TTI as:

$$\overline{R_i}(t) = \begin{cases} (1-\frac{1}{t_c})\overline{R_i}(t-1) + \frac{1}{t_c}d_i(t), & \text{if } i^{th} \text{ user is served} \\ (1-\frac{1}{t_c})\overline{R_i}(t-1), & \text{if } i^{th} \text{ user is not served} \end{cases} \tag{2.4}$$

here, $d_i(t)$ denotes the achievable throughput for the i^{th} flow at time t. It may be observed form the above equations that BET allocates resources to that flow which has the lowest running average throughput.

2.3.2 In-network Channel-aware/QoS-unaware Strategies

As discussed above, UEs periodically send CQI values to eNodeB using ad-hoc control messages. Based on the received CQI, the scheduler at eNodeB can predict the channel quality perceived by each UE and therefore, can estimate the maximum achievable throughput. This section discusses a few important strategies which utilize CQI feedbacks during the resource allocation process.

Maximum Throughput (MT)

The objective of this strategy is to maximize the system throughput in terms of spectral efficiency. In the endeavor to maximize overall system throughput, the strategy allocates RBs to those flows in the current TTI which are enjoying the best $CQIs$. In this strategy, metric value for the i^{th} flow over the r^{th} RB is calculated as:

$$m_{ir}^{MT} = d_{ir}(t) \tag{2.5}$$

here, $d_{ir}(t)$ denotes the achievable throughput for the i^{th} flow over the r^{th} RB at time t.

The main disadvantage of this strategy is that it is unfair to users which experienced bad channel conditions. A practical scheduler should be intermediate between MT, that maximizes the cell throughput, and BET, that guarantees fair throughput distribution among users, in order to exploit fast variations in channel conditions as much as possible while still satisfying some degree of fairness.

Proportional Fair (PF)

As pointed to its necessity above, the proportional fair scheduler attempts to achieve a trade-off between extreme throughput maximization (MT) and absolute throughput fairness (BET). Metric for the PF strategy is define as [38–40]:

$$m_{ir}^{PF} = m_{ir}^{MT} \times m_{ir}^{BET} = \frac{d_{ir}(t)}{\overline{R_i}(t)} \tag{2.6}$$

Here, $d_{ir}(t)$ and $\overline{R_i}(t)$ are define in equation 2.3 and 2.5.

Although, the QoS unaware strategies may not directly be applicable for today's multimedia and Internet applications, however, most of the recent QoS aware schemes [10] employ them to achieve better cell spectral efficiencies, fairness between flows, reduce computational complexity etc.

2.3.3 In-network Channel-aware/QoS-aware Strategies

Modern wireless networks concurrently serve diverse applications with distinct timeliness criticalities or QoS requirements. Typically, QoS differentiation can be handled by associating a set of QoS parameters to each flow. With the knowledge of such parameter values, the scheduler can allocate RBs to flows such that a minimum performance guarantee can be provided, either in terms of data rate or delivery delay.

A plethora of channel cum QoS aware algorithms has been discussed in the literature. Authors in [22] proposed a two level resource allocation strategy where the outer level divides time into fixed duration windows of size β, over an inner TTI level. Resource allocation is done at successive window junctions by allocating the best available flows (obtained by ordering their decision index values) on a round-robin basis over the sub-channels progressing TTI by TTI until all RBs of all $TTIs$ in β are allocated. There has been several recent contributions [41–43] which prioritize flows primarily based on head-of-line packet delays. In [41], a two step per TTI flow scheduling and mapping scheme has been proposed. At the first step, RBs are allocated to those flows whose packet waiting times are nearing

their transmission deadlines. Throughput enhancement for the residual RBs is done at the second step. Yang et. al. in [42] formulate the resource allocation problem as a mixed integer nonlinear problem to minimize the average waiting time across all active flows and claim that the problem to be non-scalable due to huge computational overhead. Further, they have designed a four step heuristic approach as a cost effective sub-optimal solution to the problem. In [44], Wang et. al. presents a cross time interference aware evolutionary scheduling algorithm that fairly allocates resources based on an adaptive fairness threshold.

Many schemes that attempt to handle delay sensitive applications have been proposed in [11–13]. Among them, two notable algorithms namely, the *LOG-Rule* [11] and *EXP-Rule* [12] consider all active flows (including both *RT* and *NRT* flows) together at every *TTI* and select flows based on their individual metric values. For *NRT* flows, the metric values for both schedulers are obtained based on a proportional fair policy [9]. For *RT* flows, the metric value for the i^{th} flow on the r^{th} sub-channel is obtained as a trade-off between head-of-line packet delays and spectral efficiency as shown in equations 2.7a and 2.7b for *LOG-Rule* and *EXP-Rule* respectively:

$$m_{ir}^{LOG-Rule} = b_i log\Big(c + \alpha_i \times HOLD_i\Big) SE_{ir} \tag{2.7a}$$

$$m_{ir}^{EXP-Rule} = b_i exp\left(\frac{\alpha_i \times HOLD_i}{c + \sqrt{(1/N_{rt})\sum_p HOLD_p}}\right) S_{ir} \tag{2.7b}$$

where, b_i, c , α_i are tunable parameters, SE_{ir} is the spectral efficiency for the i^{th} flow on the r^{th} sub-channel, $HOLD_i$ is the head-of-line packet delay for the i^{th} flow and N_{rt} denotes the total number of active *RT* flows. Analysis in [10] and also the experimental analysis done in this dissertation, reveal that with its exponential nature, the *RT* metric values for *EXP-Rule* is in general more sensitive to packet delay urgencies as compared to *LOG-Rule* and therefore is more robust especially in underloaded situations. *EXP-Rule* also attempts to maintain fairness among *RT* flows by normalizing the delay of a flow over the square root of the mean delay of all *RT* flows [45].

However, both *EXP-Rule* and *LOG-Rule* suffer from three principal drawbacks. Firstly, by following a metric based selection policy that considers all active flows (combining both *RT* and *NRT* flows) together at every *TTI*, these algorithms incur high scheduling complexities. Secondly, the same reason as above (consideration of both *RT* and *NRT* flows together) creates possibilities where *NRT* flows may be prioritized over urgent *RT* flows resulting in packet loss due to missed deadlines. Thirdly, in the absence of overload estimation and dynamic *QoS* adaptation mechanisms, these algorithms often perform poorly in transient overload situations.

There has been a few works which focus on two layer resource allocation architectures [14,46,47] as heuristics towards lowering the complexity of the combined optimization problem of maximizing both *QoS* and spectral efficiency by splitting the problem and tackling it at distinct layers. In a recent work proposed in [13] (known as *EXP-VT-SH*), authors conceived an interesting two phase procedure based on cooperative game-theory that performs resource sharing combining the *EXP-Rule* with a virtual token mechanism.

Piro et. al. in [14] propose a frame-based two level packet scheduling algorithm (called *FLS*) where the first level computes the amount of data to be transmitted by each *RT* flow (such that their delay constraints may be satisfied) in the next *LTE* frame (a fixed time interval of duration 10 ms) using discrete time linear control law. The inner level then physically allocates RBs to flows at each *TTI* using a PF scheduler. To prioritize *RT VBR* flows over *NRT* flows, in the *TTI* corresponding to an *LTE* frame, RBs are first allocated to all the active *RT VBR* flows until their total amount of data calculated for the entire frame has been transmitted. The remaining RBs are then allocated to the *NRT* flows. It may be observed that by computing the total data transmission requirement of each *RT* flow for an entire frame, employing a PF scheduler for *RT* flows at each *TTI* and by distinctly prioritizing *RT* over *NRT* flows, *FLS* attempts to guarantee strict packet delivery delay bounds for the *RT* flows. However, in this endeavor, it tends to neglect spectral efficiency and therefore often suffers low

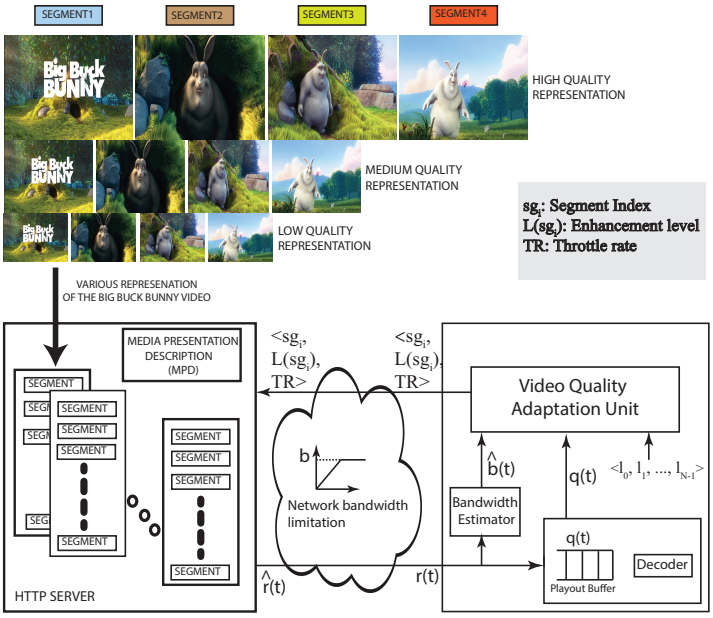


Figure 2.3: *Overview of SVC-DASH*

resource utilization. A more severe consequence of this drawback is that *FLS'* performance tends to degrade quickly when the system load increases beyond a certain utilization bound.

As discussed above, a significant component of this dissertation deals with the in-network/client resource allocation strategies for the adaptive video streaming. Before presenting the discussion of the state-of-the-art schemes for adaptive video streaming, we provide an overview of the video streaming architecture used in this dissertation.

Adaptive Video Streaming Architecture:

Video streaming over cellular networks has been growing at an exponential rate due to the ever increasing demand for video content and proliferation of heterogeneous mobile devices with increased processing capabilities. It is estimated that mobile video will generate over 73% of the total mobile data traffic by 2021 [2] and hence, video streaming over wireless is expected to be one of the main revenue generators for the current and future mobile broadband networks. However,

meeting the ultimate objective of delivering all clients/user equipments with high quality video viewing experience in the temporary varying cellular networks is a challenging task. In this effort, *Adaptive Video Streaming (AVS)* has emerged as a key technology that enables enhancement of the overall transmission quality of a service provider by allowing online video bit-rate adaptation over time. Adaptive video streaming is a performance management technique for streaming multimedia over networks which adjusts the bit-rates of videos based on perceived users' bandwidths, device capacities, screen resolutions etc. Modern adaptive video streaming technologies are almost exclusively based on *HTTP* and designed to work efficiently over large distributed *HTTP* based frameworks like Internet.

Dynamic Adaptive Streaming Over HTTP (DASH):
DASH [48] has the potential to play a major role in the transmission of video traffic in current and futuristic cellular networks. The *DASH* framework dynamically adjusts the transmission bit rate of a flow over time, based on mobile device capability and estimated link conditions. In order to achieve such dynamic bit rate adaptation, *DASH* fragments the entire video into multiple short duration chunks/segments of one to a few seconds (as shown in Figure 2.3), where each chunk is encoded at various distinct bit rates. A few technologies including *Moving Picture Experts Group (MPEG)*, *Scalable Video Coding (SVC)* etc. may be used to encode a chunk at multiple bit rates.

Each fixed duration chuck in *DASH* is stored as a regular file in the web server. The chunks may be downloaded by a user equipment periodically using standard *HTTP GET* requests. When the system experiences an overload/underload, the adaptation unit appropriately lowers/increments the number of temporal, spatial and quality layers for the next chunk of a flow in order to maintain the system within targeted load bounds. To enable such a framework, a file (known as *Media Presentation Description (MPD)* for the *DASH* standard) describing the list of chunks at all the video bit-rate versions along with the corresponding *HTTP* links is downloaded by the client prior to streaming. While most of the framework is

standardized as part of the *DASH* standard, the adaptation algorithm to select the most appropriate bit-rate for future chunks is left to the specific implementation.

2.3.4 In-network Channel-aware and *QoE*-aware Resource Allocation Schemes for Adaptive Video Flows

Numerous studies [17, 49–54] have demonstrated the benefits of in-network bit-rate adaptation strategies for adaptive videos in *LTE*. The authors in [50] present an adaptive *Guaranteed Bit Rate (GBR)* selection methodology for *SVC* video streaming over *LTE*, based on wideband *CQI* feedbacks. In this scheme, *SVC* based adaptations are performed over groups of *CQIs* and for each group a fixed temporal and quality layer is chosen. In [51], authors propose online streaming bit-rate control from content servers for adaptive video over *High Speed Packet Access (HSPA)* and *LTE*-style networks. The objective is to maximize attainable visual quality while maintaining the probability of RLC buffer overflows below a desired threshold at the transmitter. Authors in [52] have solved the problem of maximizing average video quality in *LTE* networks by reducing the resource allocation problem into an *ILP* formulation with constraints. A scheduling framework called *AVIS* for adaptive video delivery over cellular networks has been proposed in [17]. *AVIS* attempts to maximize system throughput over all flows. They modeled the resource allocation strategy as a *Multiple Choice Knapsack Problem (MCNP)* whose optimal solution is NP-hard. Further, the authors proposed a heuristic solution which is based on a greedy approach aimed at minimizing the loss in utilization of the wireless resources. This framework also provides good stability to the system by controlling the frequency of bit-rate switching perceived by the end users during a video session. However, they have not considered client-side buffer status (in their optimization framework) thus, making it susceptible to frequent buffer outages. In [53], authors propose an adaptive video streaming solution for *LTE* network which attempts to achieve fairness in the video qualities delivered to all end users. A cross-layer *QoE* aware optimization framework called *Re-buffering Aware Gradient Algorithm* (*RAGA*) for *LTE*

networks has been presented in [54]. *RAGA* attempts to restrict re-buffering probability of flows through periodic feedbacks of the playout buffer status of a set of adaptive streaming clients. A drawback of this approach is that it does not consider bit-rate switching as a component of the overall perceived *QoE*. El Essaili et al. [49] presented a content-aware multiuser *HTTP Adaptive Streaming (HAS)* video delivery framework for *LTE*. In this work, a media-aware network element (named as proxy) selects the bit-rates for all clients/video flows such that radio resource utilization in maximized. This adaptation strategy has been further extended in order to include buffer awareness in the streaming solution.

All of the aforementioned scheduling strategies are implemented in the network, primarily at *eNodeB*. A diametrically opposite approach to resource allocation are client-side schemes where each client attempts to adapt the streaming strategy and/or bit-rates corresponding to the video flow it is receiving, with the typical objective of maximizing *QoE*. In-network resource allocation schemes tend to be superior towards effective utilizing the available bandwidth while multiplexing radio resources among clients. However, in order to make good scheduling decisions, these resource allocators require periodic feedback of client status like channel conditions, buffer size etc. Obtaining periodic reports from all clients may not always be realistic. In addition to being poor in terms of significantly increasing communication costs, the strategy necessitates maintenance of a high degree of coordination among clients in order to make good adaptation decisions, and this incurs high computational overheads. Therefore, this dissertation has also delved towards the design of efficient client-side schemes which adapt video bit-rates based on expected channel conditions and player status.

2.3.5 Client-side Network-aware and Device-aware Video Flow Adaptation Methodologies

Effective rate adaptation strategy is an important tool for delivering satisfactory quality of viewing experience to the end user even during fluctuating network conditions. In order to provide satisfactory *QoE*, a plethora of adaptation strate-

gies [55, 56] has been proposed in literature. Riiser et. al. [21] investigated the performance of a few important client-side *HTTP* based video streaming services (viz. *Adobe HTTP Dynamic Streaming*, *Apple HTTP Live Streaming*, *Microsoft IIS Smooth Streaming*) in a 3G mobile network scenario. Their comparison revealed that the *Adobe HTTP Dynamic Streaming* service was quite responsive to bandwidth fluctuations and thus achieved good bandwidth utilization. However, it experienced frequent video stuttering due to short buffering at the client. *Microsoft IIS Smooth Streaming* achieved better video quality with respect to *Apple HTTP Live Streaming*, but at the same time encountered a higher degree of re-buffering. Authors in [23] evaluated dynamic adaptive streaming using bandwidth traces obtained in a vehicular scenario and inferred that re-buffering events may be minimized by maintaining playout buffer sizes above a certain threshold. In their work, the *DASH* framework performed well, avoided re-buffering events, but encountered frequent bit-rate switches which ultimately pulled down the overall quality of experience. In [25], authors have proposed a client-side buffer management algorithm which attempts to reduce playback interruptions by appropriately selecting the video quality of the next segment based on estimated bandwidth. In this work, the adaptation strategy attempts to keep the playout buffer filled with high quality segments by randomizing the segment request time from the client. There are few works which have been proposed within the *SVC*-based adaptation streaming framework [57, 58]. In [57], authors proposed an adaptation strategy that decides either to upgrade an already downloaded *SVC*-coded segment to an appropriate level or to download the next segment at a particular enhancement level. Authors in [58] proposed a hybrid approach for adaptive video streaming. In this scheme, the *SVC* video chunks/segments have been downloaded using two existing video streaming technologies, namely, progressive download (for the base layer) and adaptive streaming (for the enhancement layers).

Control-theoretic approaches, including the use of proportional-integral controllers, have also been employed to improve streaming performance [59–61]. These approaches typically attempt to control bit-rates, throttling rates etc. for

video segment downloads by measuring and estimating critical time varying system parameters including instantaneous bandwidth, playout buffer size etc. Recently, few works have adopted a state machine based control approach for *DASH* based streaming [62,63]. However, both these works suffer from the critical drawback of ignoring instantaneous network conditions in their decision strategy and working by observing only the client side playout buffer status. A similar strategy has also been proposed by Huang et al. [64].

In the endeavor to enable the influence of the quality of future segments downloads in the rate adaptation strategy, *MDP* based adaptation strategies have been proposed to model the dynamics of a video streaming system under time-varying network conditions [65–69]. Authors in [65] designed the rate adaptation of video streaming using *MDP* where uncertainty in received network bandwidth is modelled as a Markov chain with its own bandwidth state. In [69], authors have jointly optimized scheduling and error correction in layered videos, using *MDP*.

2.4 Performance Metrics

The efficacy of the proposed resource allocation strategies presented in this dissertation have been evaluated using a comprehensive set of performance metrics. For generic delay sensitive variable bit rate flows, the following *QoS* measurement metrics have been used:

1. *Packet Loss Rate (PLR)*: It is defined as the fractional number of data packets that fail to reach destination per unit time. More formally,

$$PLR = 1 - \frac{Packet\ Reception\ Rate}{Packet\ Transmission\ Rate}$$

2. *Average Goodput (goodput)*: It is the application level throughput; that is, the actual useful information delivered per unit time, excluding protocol overhead bits as well as retransmitted data packets. It is measured in bits per second (bps).

3. *Spectral Efficiency (SE)*: This may be defined as the average information rate achieved by a scheduler per unit bandwidth. It is a measure of how efficiently a limited frequency spectrum is utilized by the physical layer protocol. Unit of *SE* is bits per hertz per second (bits/Hz/sec).

4. *Fairness Index (FI)*: This measure helps to evaluate the relative fairness in the transmitted data among all active flows. Fairness is said to have been maintained among a set of flows if all of them receive the same share of the bandwidth over a given interval. In this work, the degree of fairness has been measured using the Jain's fairness index [70].

$$J(b_1, b_2, ..., b_n) = \frac{\left(\sum_{i=1}^{n} b_i\right)^2}{n \times \sum_{i=1}^{n} b_i^2} \tag{2.8}$$

where, n is the total number of flows and b_i is throughput of the i^{th} user. The range of fairness index is in between $\frac{1}{n}$ (in worst case) to 1 (in best case) and it is maximum when all the users receive the same bandwidth share.

Performance of systems consisting of real-time and streaming video flows are more suitably evaluated using metrics that allow the measurement of delivered *QoE*. This dissertation has employed the following *QoE* metrics:

- $\overline{\boldsymbol{Buffering}}\%$: It is defined as the average buffer outage percentage over the entire simulation duration.

$$\overline{Buffering}\% = \frac{\frac{1}{N}\sum_{N}(Buffer\ outage\ duration)}{Simulation\ time} \times 100 \tag{2.9}$$

- $\overline{\boldsymbol{PSNR}}$ (Unit: dB): This indicates the average $PNSR$ value over all flows during the simulation interval and measures the efficiency of a RB allocation scheme in terms of the overall video quality delivered to the UEs.

- $\overline{\textbf{\textit{Switching}}}$: Estimate of the average switching per second per video flow over the simulation length.
- $\overline{\textbf{\textit{Speed-up}}}$: Ratio of the average quality level selection times between two strategies per adaptation interval.

2.5 Summary

This chapter starts with providing a brief overview of the evolution of cellular network followed by a short description of the *LTE* architecture. Then, we have discussed the generic problems associated with traditional resource allocation strategies, especially with respect to the demands of contemporary wireless network traffic. Subsequently, we have presented a comprehensive state-of-the-art survey on various in-network/client-side bandwidth allocation and video adaptation approaches. The chapter concludes by presenting the important metrics used to evaluate the performance of the proposed scheduling approaches presented in later chapters. In the next chapter, we discuss two scheduling strategies applicable towards the effective management of real-time variable bit rate traffic. The devised scheduling strategies endeavour to deliver satisfactory quality of service to all active flows while incurring low computational overheads.

2. RESOURCE ALLOCATION STRATEGIES IN CELLULAR NETWORKS: BACKGROUND, FLAVOURS, TRENDS

Chapter 3

Low Overhead LTE Downlink Scheduling Frameworks for RT VBR Traffic

3.1 Introduction

In the last chapter, we studied various scheduling methodologies for *RT* traffics over *LTE* networks. From the insights gained through the analysis of the existing works, the following important observations and inferences may be derived:

- The primary cause for the high scheduling complexity in all the existing works is the metric based flow selection and mapping procedure at each *TTI*. This results in an overhead of at least $O(N \times R)$ per *TTI*, where N denotes the total number of active flows and R denotes the total number of available RBs. However, neither do most *RT* flows require *TTI* level scheduling granularities to meet their QoS demands, nor do channel conditions typically vary drastically over one or a few milli seconds. As an example, for a typical video application being encoded at 25 frames/sec (sampling period $SP = 40$ ms), it is sufficient for all packets of a particular video frame to respect a single delay bound that corresponds to the delay bound of the entire video frame (here 40 ms). Authors in [71] state that delays even upto 200 ms may be considered acceptable for interactive video applications. For Internet telephony, a delay of 100 ms is considered

as the limit for a good perceived quality, while even up to 300 ms delays are considered satisfactory. This indicates that it is possible to device carefully designed mechanisms that allow resource allocation at relatively coarser time scales (as compared to per *TTI* allocations) while still not significantly degrading overall system performance.

- Ignorance of spectral efficiency not only results in poor resource utilization but also causes real-time performance degradation beyond a certain system load value. Therefore, awareness of spectral efficiency is essential (to allow multiuser diversity gains) while simultaneously maintaining a stipulated level of delay sensitivity.

- In order to improve the performance of *RT VBR* traffic, special mechanisms to handle its inherent variability and burstiness are required. This may be achieved by employing efficient but low overhead *VBR* traffic prediction methodologies. Along with this, frequent periodic inspection of the input queues for instantaneously arrived data bursts also helps to obtain better overall real-time performance.

With these insights, we have developed two low-overhead resource allocation frameworks which have ability to satisfy pre-specified *QoS* requirements of *RT VBR* flows in most realistic scenarios, while simultaneously providing high resource utilization and graceful *QoS* degradation in times of overload. Firstly, we have developed a low overhead three-level resource allocation framework (called *TLS*) with the ability to satisfy pre-specified *QoS* requirements of *RT VBR* flows in most realistic scenarios, while simultaneously providing high resource utilization and graceful *QoS* degradation in times of overloads. At the outermost layer, time is divided into a sequence of fixed sized intervals called super-frames. At each super-frame boundary, this layer calculates the amount of data that each *RT* flow should transmit in the following super-frame in order to satisfy their delay constraints. The system overload handling mechanism is also embedded at this layer.

A super-frame is further divided into a specific number of frames whose duration may be modified dynamically (typically in the range 5 to 20 ms) as a trade-off between required scheduling accuracy and *RB* selection overheads. It may be noted here that as opposed to all the currently known scheduling mechanisms, *TLS* does not undertake an expensive metric optimization based *RB* allocation policy at each *TTI*. Instead, all *RBs* within a given frame are statically allocated at the beginning of each frame. Finally, at the third and innermost level, the RBs are physically allocated to the flows for transmission at each *TTI* through a constant time look-up process.

In the second scheduling strategy, we have attempted at a combined hybrid offline-online approach in order to minimize overall resource allocation overheads while maintaining a minimum satisfactory level of *QoS*. In this scheme, the flows are firstly classified into priority buckets based on real-time criticality factors. During the offline phase, the scheduler attempts to maintain the system load within a pre-specified safe threshold value by selecting an appropriate number of buckets. This selection procedure makes use of *Supervisory Control Theory* of *Discrete Event Systems* to synthesize an offline scheduler. Then, we have devised an online resource allocation strategy which runs on top of the offline policy and attempts to refine the offline solution in oder to minimize the impact of the inherent variability in cellular networks.

The next section presents an overview of *TLS*' working principle describing each of its constituent layers in detail.

3.2 The TLS Framework

The *TLS* is a flexible resource allocation framework aimed at enabling mobile operators to effectively achieve good *QoS* and high cell spectral efficiency while incurring low overall scheduling overheads. As shown in Figure 3.1, the proposed framework has been designed as a three layered split architecture which interacts together in order to dynamically allocate *RBs* to the active flows, taking into

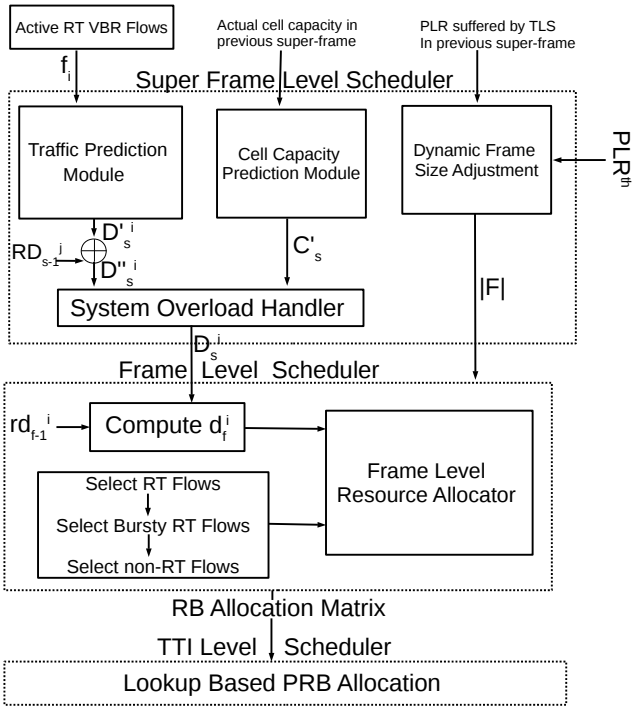


Figure 3.1: *The Proposed Three Level Scheduling Framework (TLS)*

account the instantaneous channel states, system load and maximum tolerable delays of *RT* flows. The three layers of *TLS*, namely, the *Super-Frame Layer*, the *Frame Layer* and the *TTI* Layer actually represent three distinct time domains at which *TLS* operates. As shown in Figure 3.2, a super-frame is typically composed of numerous frames, while a frame in turn is composed of one to several TTIs. We now discuss in detail the functions of each of these layers.

3.2.1 Super-Frame Level Scheduler

The super-frame level represents the outermost layer of *TLS*. From Figure 3.1, it is observed that the super-frame level scheduler takes active *RT VBR* flows f_i (where i denotes index of the flow) as input and estimates the data $D_s^{\prime i}$ to be transmitted for the i^{th} flow in the commencing super-frame SF_s such that *QoS* requirements are satisfied. The estimated value of $D_s^{\prime i}$ is calculated within the *Traffic Prediction Module* (section 3.2.4 presents an overview of the working

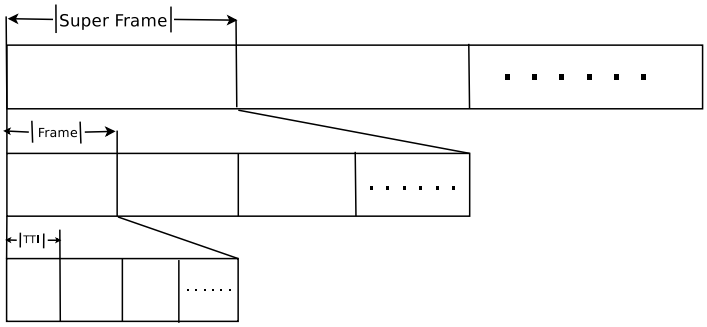


Figure 3.2: *Three distinct levels of scheduling granularities in TLS*

principle of the traffic prediction module). After $D_s^{'i}$ is calculated, it is added to the remaining non-transmitted data for the i^{th} flow (RD_{s-1}^i) in the previous super-frame to obtain $D_s^{''i}$, the total amount of data that is to be transmitted for the i^{th} flow in SF_s. Thus, the total estimated data for SF_s over all flows becomes,

$$D_s^{Total} = \sum_i D_s^{''i} \tag{3.1}$$

The system is considered to be susceptible to overload in super-frame SF_s, if the estimated cell capacity $C_s^{'}$ for the interval SF_s is less than D_s^{Total}. $C_s^{'}$ is computed in the *Cell Capacity Prediction module* (a discussion on this prediction methodology has been provided in section 3.2.5). In such a situation, it is not feasible for the scheduler to satisfy the *QoS* requirements of all flows. Hence, the *System Overload Handler* is called to proportionally scale down the amount of data for each flow (using equation 3.2) such that the total demand can be equal to the total capacity.

$$D_s^i = \begin{cases} D_s^{''i} \times \frac{C_s^{'}}{D_s^{Total}}, & \text{if } C_s^{'} < D_s^{Total} \\ D_s^{''i}, & \text{Otherwise} \end{cases} \tag{3.2}$$

This simple graceful degradation mechanism attempts to ensure that fairness in the data rates achieved by *QoS* sensitive flows is maintained even in the presence of overloads.

It may be noted that the size of the super-frame is an important parameter of the algorithm as it governs the quality of its real-time performance versus schedul-

ing overheads. Although, a very low super-frame size may provide high real-time accuracy, the associated scheduling complexity may exhibit higher overhead. The number of *frames* within a super-frame is decided at super-frame boundaries using a procedure described later in Algorithm 3 (refer section 3.2.7).

3.2.2 Frame Level Scheduler

Each super frame is divided into a number of equal sized frames. The frame duration (typically in the range *5 ms* to *20 ms*) is chosen based on the available computational resources (processing power available at eNodeB) and there exist a trade off between the required scheduling accuracy versus the RB allocation overheads. The frame level scheduler is called at each frame boundary within a super-frame in order to calculate the data d_f^i (f denotes the frame index) to be transmitted by each *RT VBR* flow f_i in the current frame F_f and also to statically pack the calculated data within the RBs of the frame. d_f^i is computed as follows:

$$d_f^i = (D_s^i/N) + d_{f-1}^{'i} \tag{3.3}$$

where, D_s^i represents the total data to be sent by the i^{th} flow in the s^{th} super frame, $d_{f-1}^{'i}$ is the amount of non-transmitted data (if any) for the i^{th} flow from the previous frame (F_{f-1}) within the current super-frame and N denotes the number of frames in the super frame.

As shown in Figure 3.1, the obtained d_f^i values are passed to the *FLRA*. *FLRA* gives the highest priority to *RT* flows, followed by *BRT* flows and *NRT* flows. This is because, the *RT* flows have pre-specified *QoS* requirements with real-time urgency bounds. On the other hand, *NRT* flows do not have such strict *QoS* demands. To exemplify, let us compare the nature of resource demands for real-time video traffic and non-real-time best-effort file downloads.

The authors in [71] have stated that in order to provide acceptable quality of service to an interactive video application, delay must be upper-bounded by 200-300 ms. On the other hand, file download traffic is best-effort in nature and

does not have such strict delay requirements. *BRT* traffic are a result of dynamic structural changes in *RT VBR* flows which cause sudden modifications in their *QoS* requirements. For example, in a typical video stream (*RT VBR* flow), a scene change may often cause significant variations in the data volumes of two consecutive video frames, effecting a corresponding modification in its data rate demand. Due to this *FLRA* checks the presence of *BRT* data before considering *NRT* flows when the system has residual unallocated RBs after the consideration of all *RT* flows. In the third round, *FLRA* allocates the remaining RBs (if any) to the *NRT* flows using a proportional fair [9] mechanism. A pseudo-code for the above scheme is presented in Algorithm 1.

ALGORITHM 1: Frame Level Resource Allocator (FLRA)

Input: Frame duration ($|F|$), Total number of sub-channels ($|sch|$)
Output: RB allocation Matrix for the k^{th} frame

```
Let Rem_RB_r denote the remaining unallocated RBs in the r^th
  sub-channel at any given instant;
for each sub-channel do
    Initialize Rem_RB_r = |F| ;
Let Q_sch denote the set of currently available sub-channels having
  unallocated RBs;
Let S_RT, S_BRT and S_NRT be the set of RT flows, bursty RT flows and NRT
  flows respectively;
Call function Schedule_Flows (S_RT) ;          /* Refer Algorithm 2 */
if Q_sch ≠ ∅ then
    Call function Schedule_Flows (S_BRT) ;     /* Refer Algorithm 2 */
else
    Exit;
if Q_sch ≠ ∅ then
    Call function Schedule_Flows (S_NRT) ;     /* Refer Algorithm 2 */
```

Corresponding to each sub-channel, all *RT VBR* flows are assigned distinct metric values based on two criteria: (i) Spectral efficiency of the flow on the sub-channel (ii) urgency of the flow based on its Head-of-line Delay (HOLD) value. The metric value m_{ir} for the i^{th} *RT* flow on the r^{th} sub-channel is defined as

follows:

$$m_{ir} = \begin{cases} SE_{ir}, & \text{if } HOLD < (SP - th) \\ \frac{\gamma \times HOLD_i}{MaxDelay_i} \times SE_{ir}, & \text{Otherwise} \end{cases} \tag{3.4}$$

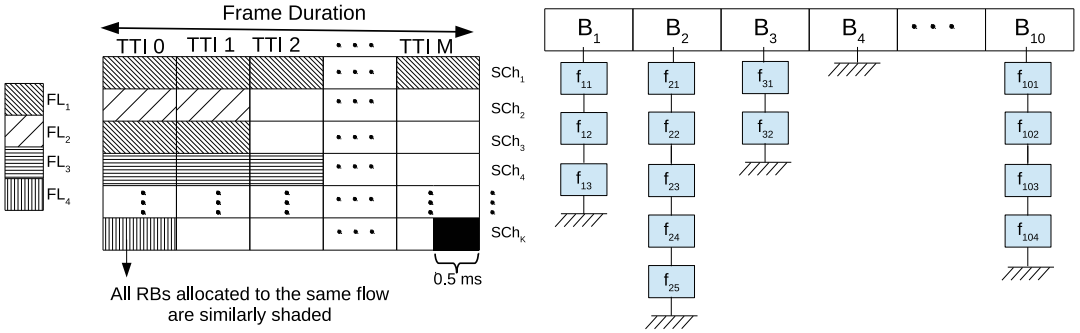
where, γ and th are tunable constants which denote the urgency factor and urgency threshold respectively (a sensitivity analysis of these tunable constants on the performance of *TLS* has been included in subsection 3.3.1.1), $HOLD$ is the head-of-line packet delay, SP is the sampling period of the *RT VBR* flow, $MaxDelay$ denotes the maximum allowable waiting time of a packet after which it is dropped from the queue (this represents the real-time criticality of the flow given predictable packet delivery through the core network) and SE denotes the spectral efficiency of the flow on the current sub-channel. For a particular flow f_i, if the first condition ($HOLD < SP - th$) in equation 3.4 is satisfied, f_i is categorized as a *normal flow*. Otherwise, it becomes an *urgent*. In order to prioritize urgent flows over normal ones in a given bucket, all normal flows are inserted into the bucket linked list at the tail while all urgent flows are inserted at the head. Higher values of γ and th implies higher priority to packet delay urgencies over spectral efficiency.

The metric used for *NRT* flows is the same as that employed by PF schedulers [9]. Metric value for the i^{th} *NRT* flow on the r^{th} sub-channel is defined as follows:

$$m_{ir} = SE_{ir}/\overline{R_i} \tag{3.5}$$

where, $\overline{R_i}$ is running average throughput for *NRT* flows and SE denotes the spectral efficiency. The obtained metric value helps assignment of priority to a flow for RB allocation within a given sub-channel at a frame boundary.

At each round of *FLRA*, a specific selected set of flows (S_{RT} or S_{BRT} or S_{NRT}) are allocated to all the sub-channels on a round-robin basis using the procedure *Schedule_Flows()* as described in Algorithm 2. For a particular sub-channel, at each allocation step of a given round (lines 6 to 16; each iteration within the *for loop* at line 5), the approximately best unscheduled flow (obtained by bucket



(a) An intermediate stage in the flow-to-RB mapping process. The frame consists of M TTIs. SCh_k and FL_i denote the k^{th} sub-channel and i^{th} flow respectively.

(b) The Bucket data structure

Figure 3.3: *Frame level allocation of resource block chunks to flows*

sorting the flows on the metric values presented in equations 3.4 and 3.5) for the sub-channel is selected and allocated either as many RBs as required by the flow, or until all RBs of the sub-channel are exhausted. One round in this round-robin algorithm completes after the allocation of one or more RBs of all sub-channels using the *for loop* (lines 5 to 16). That is, one round corresponds to one iteration of the *while loop* (lines 5 to 16). Figure 3.3(a) depicts an intermediate partially allocated stage in the *FLRA* flow-to-RB mapping process. In a given iteration of the *for loop*, if all RBs of a sub-channel gets exhausted, the remaining data for the flow must be transmitted through a different sub-channel. If such a sub-channel is not available, the remaining data $d_f^{'i}$ for the flow must be carry-forwarded to the next frame F_{f+1} (refer equation 3.3). In Figure 3.3(a), we depict a situation in which a flow FL_1 has been partially allocated in sub-channel SCh_1 using all its RBs and partially in SCh_3 using its first two RBs. We now present a description of our low overhead bucket sort based traffic flow ordering mechanism.

Corresponding to each sub-channel at the beginning of a frame, the flows are partitioned into a constant number of buckets $|B|$ based on their metric values for that sub-channel. The value of $|B|$ is a design parameter and must be appropriately chosen. In this work, we have used $|B| = 10$ (A sensitivity

ALGORITHM 2: Function: *Schedule_Flows*(S)

Input:
(i)Data to be transmitted for each flow f_i in F_f, the ensuing frame (d_f^i),
(ii) Metric value for the i^{th} flow on the r^{th} sub-channel (m_{ir}),
(iii) Set of sub-channels currently having unallocated RBs (Q_{sch}),
(iv)Remaining unallocated RBs in the r^{th} sub-channel (Rem_RB_r)
Output: RB Allocation Matrix for the k^{th} frame
`/* All variables stated above under input and output are global */`

```
for Each sub-channel r in Q_sch do
    Bucket sort queue of selected flows (S) in non-increasing order of m_r^j
     values and store in queue FQ_r;
Let FQ = {FQ_1, FQ_2, ..., FQ_|FQ|}, denote the set of all flow queues ;
while Q_sch ≠ ∅ and S ≠ ∅ do
    for Each sub-channel r in Q_sch do
        Select the next unscheduled flow f_i from FQ_r ;
        Let RB_r^i be the total number of RBs required to transmit d_f^i
         through the r^th sub-channel;
        if Rem_RB_r > RB_r^i then
            Allocate RB_r^i RBs to f_i; Update RB Allocation Matrix;
            Update Rem_RB_r = Rem_RB_r − RB_r^i;
            Remove f_i from all sub-channel queues FQ and from the set of
             flows S;
        else
            Allocate Rem_RB_r RBs to f_i; Update RB Allocation Matrix;
            Remove sub-channel r from Q_sch;
            Let d denote the amount of f_i's data that may be transmitted
             through the Rem_RB_r RBs of sub-channel r;
            Update d_f^i = d_f^i − d;
Return ;
```

analysis on the effect of the number of buckets on the performance of TLS has been included in subsection 3.3.1.1). The bucket data structure for each sub-channel has been implemented using an array B of linked lists $B_1, B_2, ..., B_{10}$ as depicted in Figure 3.3(b). Each linked list B_k of B forms the bucket of flows with corresponding metric range MR_k as shown in Table 3.1. The lower and

Table 3.1: *Range of metric values used for bucket sorting*

Bucket	Metric Range (MR_i)
B_1	$0 \leq MR_1 < 0.494$
B_2	$0.494 \leq MR_2 < 0.883$
B_3	$0.883 \leq MR_3 < 1.438$
B_4	$1.438 \leq MR_4 < 1.711$
B_5	$1.711 \leq MR_5 < 2.016$
B_6	$2.016 \leq MR_6 < 2.405$
B_7	$2.405 \leq MR_7 < 2.811$
B_8	$2.811 \leq MR_8 < 3.211$
B_9	$3.211 \leq MR_9 < 3.555$
B_{10}	$3.555 \leq MR_{10} < metric_{Max}$

upper bounds for the metric range of a given bucket is approximately equal to the spectral efficiencies corresponding to its lowest and highest supported *CQI* level. For example, the lower bound for bucket B_3's metric range (MR_3) is 0.883, which corresponds to *CQI* level 7 while its upper bound 1.438 corresponds to *CQI* level 9. After all the flows have been allocated to appropriate buckets, a linear scan of these buckets starting from B_{10} to B_1 sorts the flows into a single queue arranged in order of approximately non-increasing priorities.

3.2.3 TTI Level Scheduler

FLRA allocates RBs to the flows for the entire frame duration at the frame level. After this, as Figure 3.1 depicts, an O(1) lookup operation on the *RB Allocation Matrix* obtained from the *FLRA* module is used to physically allocate RBs at each *TTI*. During such RB allocation at any given *TTI*, the appropriate Modulation and Coding Scheme (MCS) is selected based on the instantaneous *CQI* feedback received for a flow-RB pair at that *TTI*. An overview of the traffic prediction methodology, the cell capacity prediction mechanism and an analysis of the overall computational complexity of TLS are presented in the next three sub-sections.

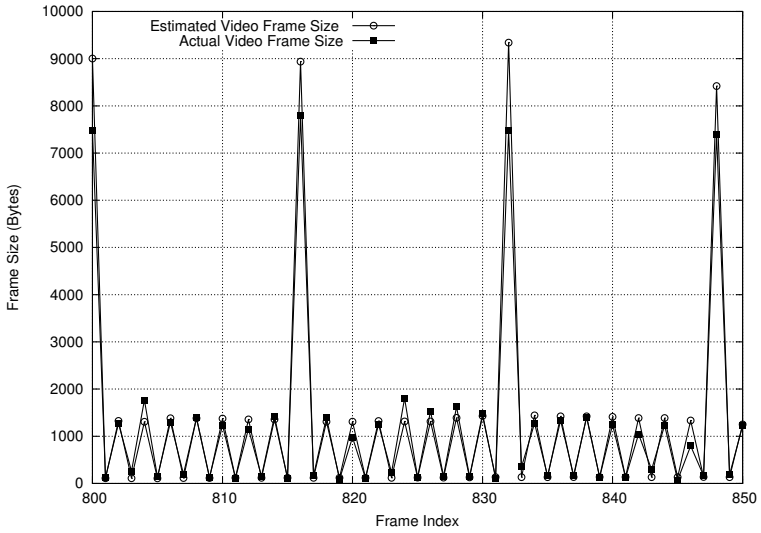


Figure 3.4: *ATSAES: Comparison of estimated Vs actual video frame sizes*

3.2.4 Traffic Prediction Methodology

As mentioned in section 3.2.1, the data to be transmitted for each flow is estimated within the *Traffic Prediction Module (TPM)* at the super-frame level. This module estimates the data for the different *VBR* flows by exploiting their slowly decaying auto-correlation properties and thus circumvents bandwidth over-provisioning problems effected by conservative allocation [72]. Although, a plethora of different *VBR* traffic prediction algorithms [73, 74] are available, a light weight algorithm must be used here as TLS employs the estimation mechanism on-line at each super-frame boundary. The implementation of exact *TPM* will vary depending on the nature of flows being predicted. In addition, any such strategy must gradually learn and tune specific parameters over time to allow accurate prediction for a given type of flow. Thus, within the traffic prediction module, a separate instance if the traffic estimator used, must run for each flow, as shown in Figure 3.1. This mechanism allows a given estimator instance to customize itself with respect to its corresponding flow and thereby deliver better prediction accuracy over time.

This work uses *Adaptive Trend and Seasonality Adjusted Exponential Smooth-*

ing (ATSAES) [75], a simple and reasonably accurate time-series forecasting technique with prompt responsiveness to process changes and only a constant time computational overhead. Although, the fundamental methodology remains same, *ATSAES* must be adapted according to the type of the traffic flow being estimated. Trace based H.264-encoded video traffic has been used as a representation of *RT VBR* flows during the experimental analysis of TLS. At each super-frame boundary, the *Traffic Prediction Module* (TPM) estimates the total amount of data to be transmitted by the flows. In this section, we present an overview of the steps followd by *TPM* for H.264-encoded videos.

In video coding, a group of pictures (or GOP structure), specifies the order in which intra (I)- and inter (P and B)-frames are encoded. I, P, and B-frames of H.264-encoded video sequences are encoded with different degrees of compression and possess varying statistical characteristics which result in short-term bit rate variations. Figure 3.4 depicts the three levels of distinct video frame sizes for I-frames (approx 7000 to 8000 bytes), P-frames (approx 1000 to 2000 bytes) and B-frames (approx 0 to 100 bytes) which rehashes itself in a cycle with a fixed frequency. For example, the video frame sequence represented in Figure 3.4 follows a GOP(16,2) structure, and hence I-frames with similar frame sizes are observed every 16 frames with alternating P and B-frames in between. To effectively estimate in the prediction process, the data contained in this three level GOP structure, *TPM* uses an independent seasonality index S_I, S_P and S_B for the I, P, and B-frames respectively, which normalizes the three distinct levels video traffic to the same level (deseasonalized level). The deseasonalized video frames (all of whose I, P and B-frames are at the same level) is then used to evaluate the current trend of the data content in the video traffic. The prediction module then predicts the amount of data in future video frames by using the obtained trend and the deseasonalized data content in previous video frames. Moreover, *TPM* attempts to detect structural changes in the underlying sequence (e.g., scene change) and updates the components of the prediction module (level, trend, seasonality index) accordingly in each super-frame. The ATSAES based

prediction mechanism progresses in three steps:

- Initialization: At this step, initial estimates of level (L_0), trend (T_0), and seasonality factors (S_I, S_P, and S_B) for I, P and B-frames are computed. The prediction module develops initial components, namely, level, trend and seasonality index for the video traffic with the help of the first K GOPs of the video flow. This is carried out only once when the flow starts.

- Forecasting: This step is executed at the boundary of each super-frame SF_s, to estimate the amount of data to transmit ($D_s^{'i}$) for each RT flow f_i in the next super-frame duration. For this, first the total number of predicted video frames (p) within a super-frame is computed as:

$$p = \frac{|SF_s|}{SP_i} \tag{3.6}$$

  where, $|SF_s|$ is the duration of the s^{th} super-frame and SP_i is the sampling period for the i^{th} RT flow. For example, the value of p is 3 for a super-frame duration of 120 ms and sampling period is 40 ms (video traffic transmitted at 25 frames/secs). For an arbitrary video frame (say, the t^{th} video frame index), the size of the $(t+q)^{th}$ (where, $q = 1, 2, ..p$) video frame is predicted using the following equation:

$$L_{t+q} = L_t + q \times T_t, \tag{3.7a}$$

$$F_{t+q} = L_{t+q} \times (S_I \; or \; S_P \; or \; S_B)_{t+q} \tag{3.7b}$$

  where, L_{t+q} is the deseasonalized level and F_{t+q} is the predicted video frame size for the $(t+q)^{th}$ frame index.

- Estimate Updation: After observing the actual video frame size for the $(t+1)^{th}$ frame index (D_{t+1}) during the s^{th} super-frame, the third step modifies the estimation of level L_{t+1}, trend T_{t+1}, and seasonality factors S_I,

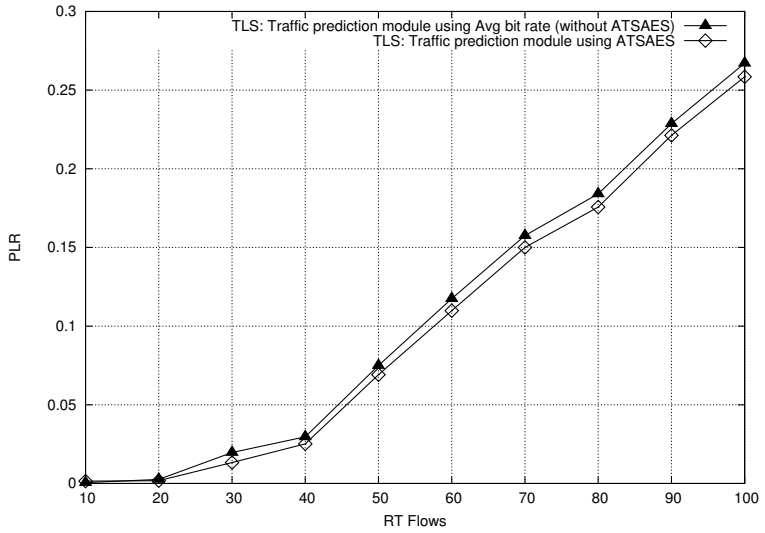


Figure 3.5: *RT VBR Packet Loss Rate Vs # RT flows (With and without ATSAES based prediction)*

S_P, and S_B at the start of the $(s+1)^{th}$ super-frame as given below:

$$L_{t+1} = a \times \left(\frac{D_{t+1}}{S_{t+1}}\right) + (1-a) \times (L_t + T_t) \tag{3.8a}$$

$$T_{t+1} = b \times \left(L_{t+1} - L_t\right) + (1-b) \times T_t \tag{3.8b}$$

$$S_{t+i+1} = c \times \left(\frac{D_{t+1}}{L_{t+1}}\right) + (1-c) \times S_{t+1} \tag{3.8c}$$

Where, a, b, and c are the smoothing constants for level, trend and seasonality factors respectively, and D_{t+1} is the actual data for the $(t+1)^{th}$ video frame index.

Figure 3.4 presents a comparison between the obtained *ATSAES* estimated video frame size values with respect to their actual values for the *nbc H.264 AVC trace* (QP =28) [1]. It may be observed from this figure that our generated estimated values are reasonably accurate in almost all cases. In Figure 3.5, we show plots to compare the packet loss rates of TLS with and without *ATSAES* based estimation. As expected, the results are always better with *ATSAES* based prediction as it reduces bandwidth over provisioning problems.

3.2.5 Cell Capacity Prediction Methodology

The total estimated capacity of the cell C'_s, for the duration of the commencing super-frame SF_s, is computed by the *Cell Capacity Prediction module* (refer Figure 3.1). The actual capacity is a function of numerous non-uniform parameters like resource allocation strategy, total number and position of *UEs* (this affects the gain due to diversity), *SINR* of *UEs* etc. Although a very accurate prediction procedure must consider all these parameters, the computational overheads involved in prediction will become higher and higher as the number of parameters are increased. This additional overhead may potentially become unacceptable towards online application. Therefore, this work uses the light weight (O(1)) yet reasonably efficient *Adaptive Response Rate Simple Exponential Smoothing (ARRSES)* [76] algorithm and adapts it for cell capacity prediction. Given C_{s-1}, the actual cell capacity and C'_{s-1}, the predicted capacity for the $(s-1)^{th}$ super-frame, estimated cell capacity for the s^{th} super frame may be obtained as:

$$C'_s = C_{s-1} + \alpha_s \times (C_{s-1} - C'_{s-1}) \tag{3.9}$$

Here, α is a positive constant whose value is important in determining the smoothing characteristic. It is customary to use values of α in the range 0 to 0.2 to be able to effectively filter out noise. However, a problem with such low values often arise when the system encounters sudden genuine changes in cell capacity. The forecasting system then takes a long time to home in to a new level. *ARRSES* handles this by making α automatically adaptive as follows:

$$\alpha_s = \left|\frac{E_{s-1}}{M_{s-1}}\right|, \tag{3.10a}$$

$$E_{s-1} = \beta \times e_{s-1} + (1-\beta) \times E_{s-2}, \tag{3.10b}$$

$$M_{s-1} = \beta \times |e_{s-1}| + (1-\beta) \times M_{s-2}, \tag{3.10c}$$

$$e_{s-1} = C_{s-1} - C'_{s-1} \tag{3.10d}$$

where, (i) e_{s-1} is the actual prediction error at the currently completed super-frame SF_{s-1}, (ii) β is a choice variable ($\beta = 0.4$ has been used in our experiments

as *ARRSES* was found to achieve the minimum root-mean-square error with this value), (iii) E_{s-1} is the smoothed estimate of the actual error observed in the predicted cell capacity in super-frame SF_{s-1} (calculated as a weighed average of E_{s-2} and the last prediction error (e_{s-1})) and (iv) M_{s-1} is the smoothed estimate of the absolute error observed in the predicted cell capacity in super-frame SF_{s-1} (calculated as a weighted average of M_{s-2} and the last absolute prediction error $|e_{s-1}|$). When the prediction (c'_{s-1}) is consistently under/over estimating, correspondingly, E_{s-1} also increases/decreases monotonically. Then, α_i will take larger and larger values progressing towards its maximum value of 1, thus making the method more responsive to change in the time series. However, if the prediction module is forecasting accurately (i.e. e_s is consistently very small), the parameter α_s will tend towards zero, making the method *smooth out* random variations in the signal. Our experimental analyses also show that *ARRSES* performs about 50% better on an average when compared to *Simple Exponential Smoothing (SES)*.

3.2.6 Computational Overhead

In this section, we provide an analysis of the per *TTI* computational overhead of the *TLS* framework. *TLS TTI* level complexity has three components: (i) effect of the scheduling overhead at the boundary of each super-frame at a *TTI*, (ii) effect of the frame level scheduling overhead on a *TTI* and (iii) RB allocation overhead at each *TTI*. The effect of the first component has been determined by first calculating the overall complexity at the super-frame level and then dividing it by the length of the super-frame to obtain amortized complexity. Amortized complexity for the second component is obtained in a similar fashion as the first component. No amortization is actually involved for the third component, this being the actual overhead of the *TTI* level *RB* allocator.

Assuming that a super-frame and frame is composed of p and q *TTIs* respectively, a measure of the per *TTI* computational complexity of *TLS* (*T(TLS)*)

may be obtained as shown in equation 3.11.

$$T(TLS) = \frac{T_1}{p} + \frac{T_2}{q} + T_3 \tag{3.11}$$

where, T_1, T_2, T_3 are overheads corresponding to the super-frame, frame and TTI level schedulers, respectively.

Super-frame level scheduling overhead (T_1):

$$\begin{aligned} T_1 = N_{rt} \times O(&\textit{Overhead of traffic prediction model}) \\ &+ O(\textit{Overhead of Cell capacity prediction}) \\ &+ N_{rt} \times O(\textit{Overhead of graceful degradation}) \end{aligned}$$

where, N_{rt} is total number of active *RT VBR* flows. As traffic prediction, cell capacity estimation and graceful degradation can all be performed in constant time, the overall complexity of T_1 reduces to $O(N_{rt})$.

Frame level scheduling overhead (T_2):

At this level, the overhead is dominated by the complexity of the *FLRA*. The *FLRA* scheme (refer Algorithm 1) calls function *Schedule_Flows (S)* thrice - first for *RT* flows, followed by bursty *RT* flows and then for *NRT* flows. Before discussing the overall computational complexity of *FLRA*, we first describe here the computational overhead for *Schedule_Flows(S)*.

The function *Schedule_Flows()* in *FLRA* bucket sorts the flows in non-increasing order of their metric values, for each sub-channel (refer Algorithm 2, lines 1-2). The bucket sorting procedure takes $O(S)$ time (where, S denotes the set of flows) to order the input queues for each sub-channel. Assuming that there are $|Q_{sch}|$ sub-channels, the complexity for bucket sorting all sub-channel queues becomes $O(S \times |Q_{sch}|)$. At each round (refer Algorithm 2, *while* loop at line 4) and for each sub-channel (refer Algorithm 2, lines 5-16), one unscheduled flow is selected for RB allocation. All the steps between lines 6-16 of Algorithm 2 may be executed in $O(1)$ time. Even the removal of a flow from all sub-channel queues (line 11) may be accomplished in constant time by employing a global queue that

maintains the currently remaining data to be allocated for each flow and allowing each sub-channel queue node to point to its corresponding index on the global queue. If we assume that all flows require at-most one RB to transmit their data (in the worst-case), and there are enough flows to consume all RBs in a frame ($S \geq |F| \times |Q_{sch}|$), then the *while* loop (lines 4-16) will execute $|F| \times |Q_{sch}|$ times. As the complexity for bucket sorting all sub-channel queues (lines 1-2) is $O(S \times |Q_{sch}|)$ and the overhead for the *while* loop (lines 4-16) is $O(|F| \times |Qsch|)$, the complexity incurred by the function *Schedule_Flows(S)* will be $O(S \times |Q_{sch}|)$.

Let us now discuss the overall overhead for the *FLRA* procedure. As $|Q_{sch}|$ is the total number of available sub-channels, the complexity of the loop in lines 2-3 of the *FLRA* scheme (refer Algorithm 1) is $O(|Q_{sch}|)$. Further, the overheads in Algorithm 1 due to the function calls *Schedule_Flows(S)* in lines 6, 8 and 12 is $O(S_{RT} \times |Q_{sch}|)$, $O(S_{BRT} \times |Q_{sch}|)$ and $O(S_{NRT} \times |Q_{sch}|)$ respectively. Therefore, the overall complexity of the *FLRA* procedure will be $O(N \times |Q_{sch}|)$, where $N = S_{RT} + S_{BRT} + S_{NRT}$.

TTI level scheduling overhead (T_3):

The *TTI* level scheduler uses a $O(1)$ overhead look-up on the *RB Allocation Matrix* (obtained from the *FLRA* module) to physically allocate RBs at each *TTI*. Hence, the Overhead T_3 is $O(1)$.

Therefore, the amortized per *TTI* complexity of TLS becomes:

$$T(TLS) = \frac{O(N_{rt})}{p} + \frac{O(N \times |Q_{sch}||)}{q} + O(1) \tag{3.12}$$

In our framework, as super-frame durations are generally of the order of hundreds of milliseconds, the super-frame level scheduling overhead may typically be contained within amortized $O(1)$ time. However, frames within a super-frame occur at much higher frequencies and as discussed above, their overheads at frame boundaries is also high. Therefore, lower frame rates/higher frame durations often allow significant reduction in computational overheads. On the other hand, it has been shown through the experiments conducted in section 3.3.1.2, higher

ALGORITHM 3: Dynamic Frame Size Adjustment Scheme

Input: Running average PLR (PLR^{avg}), Threshold PLR (PLR^{th}), Guard PLR (PLR^{gd}), Minimum frame duration ($|F|_{min}$), Maximum frame duration ($|F|_{max}$)

Output: Adjusted frame duration

Update the running average PLR (PLR_i^{avg}) for the ensuing super-frame (SF_s);
if $(PLR_s^{avg} - PLR^{th}) > PLR^{gd}$ && $|F| > |F|_{min}$ **then**
  $|F| = |F| - 1$;
else if $(PLR_s^{avg} - PLR^{th}) < PLR^{gd}$ && $|F| < |F|_{max}$ **then**
  $|F| = |F| + 1$;

frame durations degrade TLS' performance in terms of packet loss rates (and also other related parameters). Therefore, maintaining just as much frame rate as is essential to keep PLR bounded within a stipulated threshold (PLR^{th}) often helps contain computational overheads. The slack computational capacity thus scavenged (if any) through such dynamic frame duration adjustment process may be utilized in efficiently accomplishing various other important activities at eNodeB. We now describe the dynamic frame size adjustment procedure used in this work.

3.2.7 Dynamic Frame Size Adjustment Scheme

In this scheme, when the difference between the running average PLR (PLR_i^{avg}) over last z super-frames and the threshold PLR (PLR^{th}) is higher/ lower than a guard PLR (PLR^{gd}), the super-frame level scheduler increments/decrements the current frame duration by one (within a range $|F|_{min}$ to $|F|_{max}$). The running average of PLR (PLR_s^{avg}) over the last z super-frames is calculated as follows:

$$PLR_s^{avg} = \frac{1}{z} \sum_{p=s-1}^{s-1-z} PLR_p \tag{3.13}$$

where, PLR_p denotes the packet loss rate during the p^{th} super-frame. z, PLR^{th} and PLR^{gd} are design parameters which are taken as input from the service provider. A step wise description of the *Dynamic Frame Size Adjustment Scheme* is presented in the Algorithm 3.

3.3 Experiments and Results

The performance of the proposed TLS framework has been experimentally evaluated against various parameters and compared with five popular QoS aware downlink scheduling schemes considered to be useful especially with delay-sensitive applications [14]. These schemes are *New Two Level Scheduler* [22], *LOG-Rule* [11], *EXP-Rule* [12], *EXP-VT-SH* [13] and *FLS* [14]. The evaluation methodology is based on simulation studies carried out using *LTE-Sim* [34], an open source simulator for *LTE* networks.

LTE-Sim has allowed us to create a realistic single-cell with interference scenario working in Frequency Division Duplex (FDD) mode with a total available bandwidth of 10 MHz per cell. Each cell (of radius 0.5 km) contains a variable number of mobile UEs (10 to 100 UEs have been considered) which travel following the *random direction mobility model* [77]. We have used 120 ms as the super-frame duration while inner frame sizes have been varied within the range of 4 to 20 ms. Each UE receives one *RT VBR* video flow (H.264 AVC video traces ($QP = 28$) [1] obtained from several video test sequences have been used) along with one infinite buffer flow (best effort). The sampling period (SP) and maximum allowed transmission delay (MaxDelay; refer equation 3.4) for the video flows are 40 ms and 80 ms in all cases. Each data point is an average over 25 instances. All simulations run for 120 secs. A summary of the main simulation parameters are presented in Table 3.2.

3.3.1 Results

The performance of *TLS* for both *RT* and *NRT* traffic flows have been evaluated. Figure 3.6, 3.8, 3.9 and 3.11 summarize the performance results for *TLS* (with frame size = 5ms) against *LOG-Rule*, *EXP-Rule*, *EXP-VT-SH*, *New Two Level Scheduler* and *FLS* as the number of UEs vary between 10 and 100. Super-frame size is an important aspect of *TLS* because it determines the frequency at which the important operations such as *Cell Capacity Estimation*, *Traffic Prediction* and

Table 3.2: *Simulation Parameters*

Simulation Time	120s
Bandwidth	10MHz
Number of RBs	50
Frame Structure	FDD
Cell Radius	0.5 km
Number of Cells	19
Carrier Frequency	2GHz
Schedulers Evaluated	*LOG-Rule*, *EXP-Rule*, *EXP-VT-SH*, *New Two Level Scheduler*, *FLS*, *TLS*
Super-frame Duration	120 ms
Number of UEs	10 to 100
Applications per UE	1 *RT* flow and 1 *NRT* flow
Traffic Generator	Trace Based
Types of Traces	nbc, silence, sony, starwars, tokyo
Sampling Period	40 ms for *RT* flows
Max Delay	80 ms for *RT* flows
RLC ARQ	Maximum 5 retransmissions

Dynamic Frame Size Adjustment are conducted. Shorter the time scales (super-frame sizes) over which these operations are periodically invoked, better becomes their real-time accuracy. In order to measure the effect of super-frame duration on execution time and *PLR*, we have conducted a set of experiments with fixed number of *UEs* (=100). Table 3.3 shows the obtained values of execution time per TTI of the super-frame level scheduler along with the *PLR* at five distinct values of super-frame duration (= 1ms, 10ms, 50ms, 120ms, 1000ms). It may be observed from the table that as the super-frame size increases, although the average RB allocation over-head at the super-frame boundary decreases, the Packet Loss Rates (PLR) increase progressively. We have conducted all the experiments for *TLS* with the super-frame duration 120 ms.

The value of the constants γ, th, $MaxDelay$ and SP in equation 3.4 has been taken to be 2.22, -0.005, 0.08 secs, and 0.04 secs, respectively. From Figure 3.6 and 3.8, it may be observed that *TLS* shows significantly better *PLR* and goodput as compared to all the other schedulers. Figure 3.7 presents separate plots depicting

Table 3.3: *Results for execution time per TTI and PLR with varying super-frame duration $|SF|$ (milliseconds)*

$\|SF\|$	Execution Time	PLR
1	0.0560	0.290
10	0.0075	0.329
50	0.0019	0.330
120	0.0010	0.332
1000	0.00034	0.334

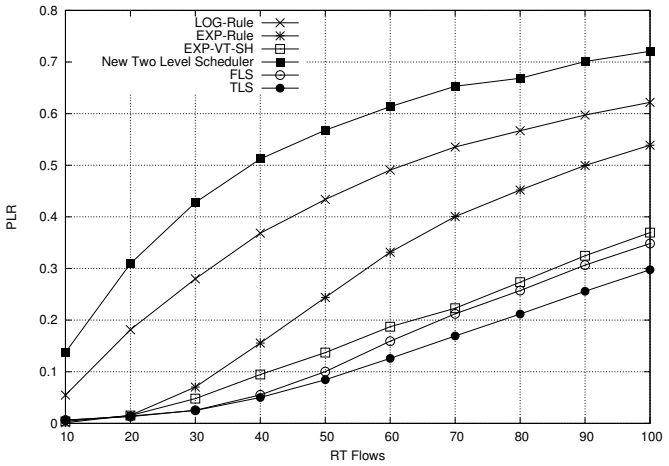


Figure 3.6: *RT VBR Packet Loss Rate Vs. #RT Flows*

95% confidence intervals [78] for the PLR suffered by TLS. These confidence intervals have been generated for a sample size of 25 (i.e 25 distinct network instances were considered for generating each interval). It may be seen from the figure that TLS and FLS show similar mean $PLRs$ for less than $\sim$ 40 RT flows. Between 50 and 90 RT flows, the worst-case performance of TLS approximately matches FLS' mean PLR. However, as the number of RT flows grow further, TLS may be observed to be outperforming FLS even in the worst case. Other schedulers perform even worse (refer Figure 3.6) with respect to TLS and hence have not been included in Figure 3.7. Such better performance may be attributed primarily to the appropriate prioritization of packet delay urgency over spectral efficiency at the frame layer (refer equation 3.4) of TLS, when the urgency crosses a given threshold th. Better PLR and goodput values are also obtained through

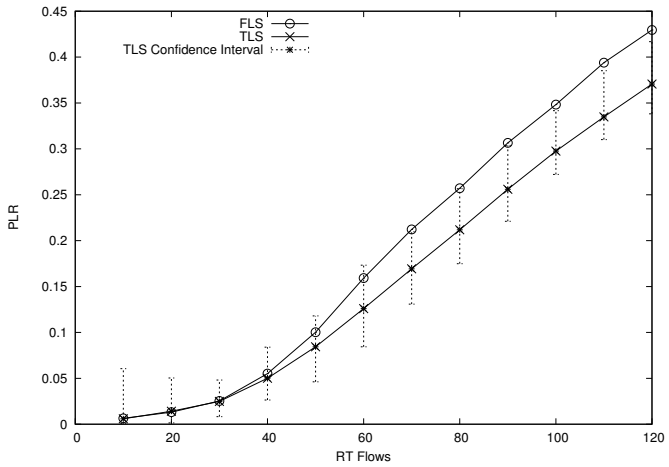


Figure 3.7: *PLR Vs. #RT Flows*

the reconsideration of bursty *RT* flows after scheduling the stipulated *RT* flows at each frame boundary to take care of sudden instantaneous data bursts.

However, as may be seen from Figure 3.9 that as a natural consequence of such heavy prioritization of *RT* flows over *NRT* ones, the overall spectral efficiency (comprising both *RT* and *NRT* flows) of *TLS* becomes poor especially for a low number of flows (The same reason may be attributed with respect to *TLS*' comparatively poor performance in terms of *NRT* goodput as shown in Figure 3.11). Figure 3.10 presents separate plots depicting 95% confidence intervals for the spectral efficiency achieved by *TLS*. These confidence intervals have been generated for a sample size of 25. To help comparative study, the figure also contains a plot showing the mean spectral efficiency for *LOG-Rule*. Spectral efficiencies achieved by other schedulers are generally poorer than *LOG-Rule* (refer Figure 3.9) and hence has not been included in Figure 3.9. Figure 3.9 and Figure 3.10 also show that the spectral efficiency achieved by *TLS* steadily increases with increasing flows and becomes better than all the other schedulers beyond about 200 flows. From Figure 3.11, it may be seen that this boost in spectral efficiency helps the *NRT* goodput of *TLS* to become comparable to that of *FLS* as the number of flows increase beyond about 70.

The reason behind *TLS*' better spectral Efficiency with increasing flows is

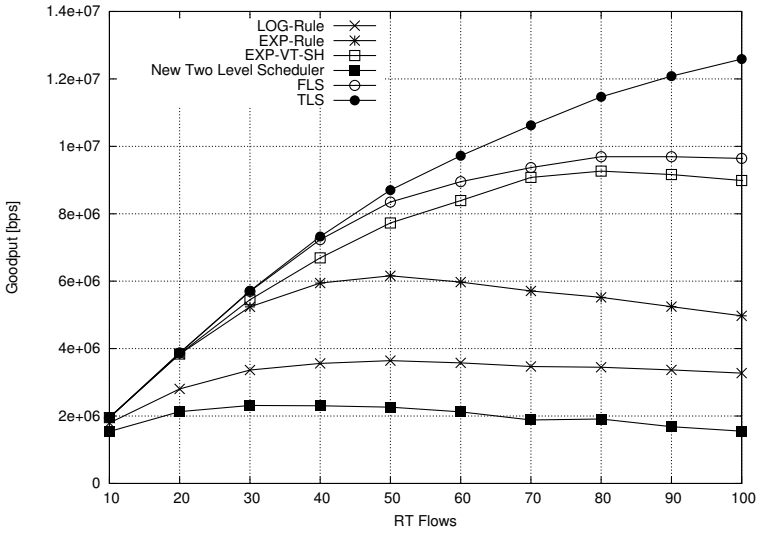


Figure 3.8: *Goodput Vs. #RT Flows*

that, at a given time and frequency, the probability to transmit flows experiencing good channel conditions increases at a higher rate for *TLS* as compared to other schedulers. *TLS* achieves such better multi-user diversity gain due to three principal reasons: i. Spectral efficiency has a very significant contribution on the overall metric value (refer equations 3.4) that *TLS* uses to allocate RBs at all *TTIs* within a frame both for *normal* and *urgent* flows. ii. For the *RT* flows, *TLS* only strives to satisfy the *QoS* demands and do not attempt to directly balance their bandwidth consumption through fairness measures. On the other hand, all the other schedulers attempt at fair distribution of the bandwidth among all flows comprising both *RT* and *NRT* (*FLS* uses a *PF* scheduler at the *TTI* level; for *EXP-Rule* and *Log-Rule*, the constant b in equation 2.7 represents the inverse of the expected spectral efficiency of a flow and hence acts as a fairness factor [10]). It is obvious that a stress on fairness restricts the maximum achievable diversity gain. iii. As the number of flows increase, a majority of the flows in the *TLS* framework become *urgent* ($HOLD \geq (SP - th)$) due to system overload and this helps to extract further diversity gain among all the urgent flows. Fairness is however indirectly achieved among the *RT* flows through the tunable constants

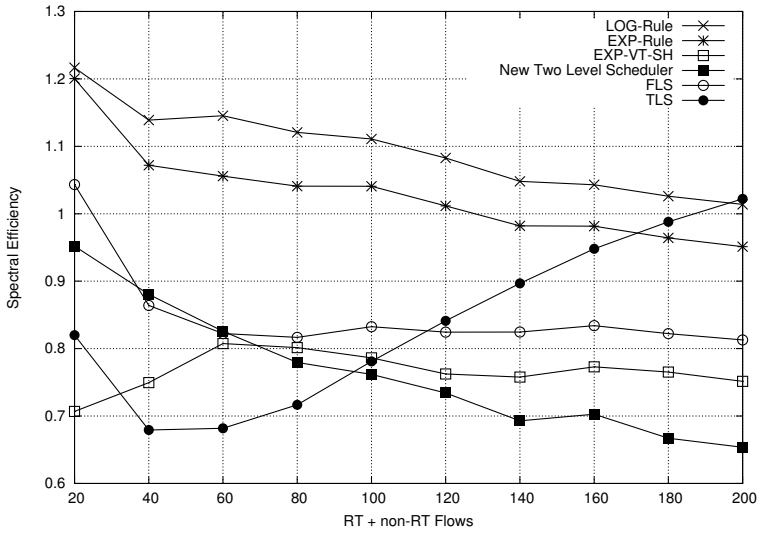


Figure 3.9: *System Spectral Efficiency Vs. #RT + non-RT Flows*

γ and th by controlling the rate at which urgent flows switch from lower towards higher priority buckets as their $HOLD$ values increase. A detailed discussion on the tunable constants is given in the next subsection.

3.3.1.1 Results With Varying γ, th and $|B|$

This section examines the impact of variation of the tunable constants γ, th, and $|B|$ (total number of available buckets) on the performance of TLS.

Impact due to variation in th and γ values:

As discussed in section 3.2.2, th is alluded as urgency threshold and is used to classify RT flows into two categories, namely *normal* ($HOLD < SP - th$) and *urgent* ($HOLD \geq SP - th$), based on their current head-of-line packet delay ($HOLD$) and Sampling Period (SP). To analyze the impact of th on TLS' performance, experiments have been carried out for two distinct values of th (th= -0.04, 0.005). H.264 encoded video traffic transmitted at a rate of 25 frames/secs has been chosen to represent RT flows. The values for the parameters $MaxDelay$ and SP for such RT flows are approximately 0.08 secs and 0.04 secs respectively.

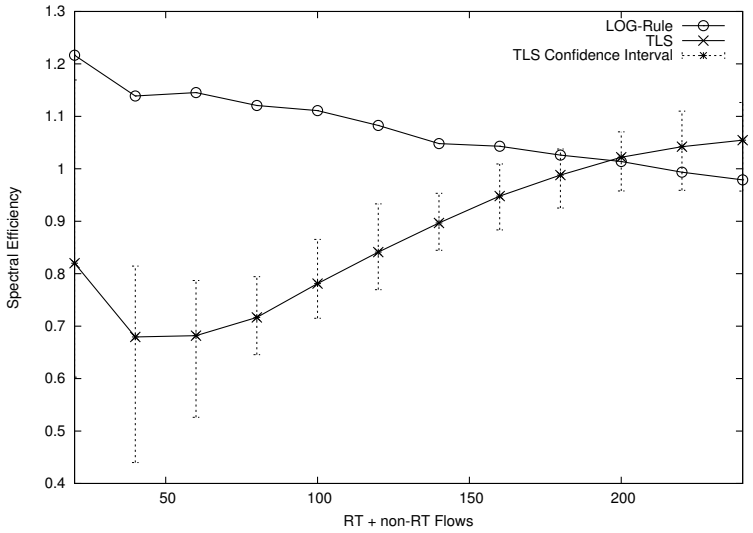


Figure 3.10: *Spectral Efficiency Vs. #RT + non-RT Flows*

When the th value is set to -0.04, $SP - th$ becomes 0.08 (0.04 - (-0.04)). In this case, a flow will be classified as *urgent* only when its $HOLD$ value is at least 0.08 secs. However, this never happens because $MaxDelay$ (which denotes the maximum waiting time of a packet ($HOLD$) before it is dropped) has been taken to be 0.08 secs. Therefore, the packets of a flow having waiting time equal to 0.08 secs will be dropped, and the $HOLD$ value of the flow will be accordingly updated. Hence, the flow will never be classified as urgent. When $th = 0.005$, the value of $SP - th$ equals 0.035 (0.04-0.005); hence, flows with $HOLD$ values greater than ~ 0.035 will becomes *urgent*.

Table 3.4: *Comparative results for Fairness Index, Spectral Efficiency with varying th and γ values*

RT Flows	Fairness Index				Spectral Efficiency			
	th= -0.04	th= 0.005			th=-0.04	th=0.005		
	$\gamma = NA$	$\gamma = 2.5$	$\gamma = 3.0$	$\gamma = 3.5$	$\gamma = NA$	$\gamma = 2.5$	$\gamma = 3.0$	$\gamma = 3.5$
20	0.82168	0.82264	0.82238	0.82252	0.31405	0.31383	0.31382	0.31415
40	0.8095	0.81046	0.81198	0.81261	0.60764	0.60884	0.60755	0.60385
60	0.7437	0.7657	0.76967	0.77425	0.82526	0.82159	0.81374	0.80349
80	0.69548	0.72537	0.73353	0.7418	0.98562	0.9716	0.95262	0.92002
100	0.6394	0.68053	0.69002	0.69459	1.07414	1.04805	1.0128	0.97181

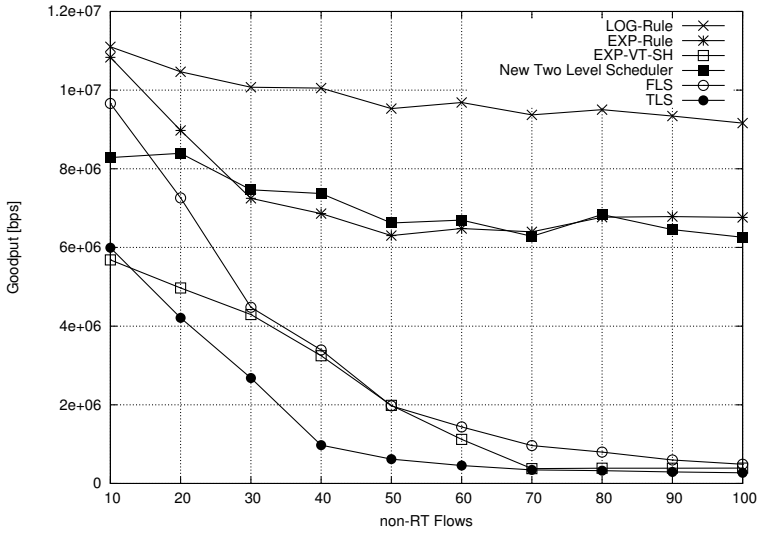


Figure 3.11: *Goodput Vs #non-RT Flows*

We refer the constant γ in equation 3.4 as the *urgency factor* which controls the rate at which flows switch from lower towards higher priority buckets as their HOLD values increase. When a flow is designated as urgent (i.e., $(HOLD > SP - th)$ in equation 3.4, the constant of proportionality γ adds an urgency factor in enhancing the metric value of the flow and therefore increases its relative priority by placing it in a higher priority bucket as compared to where it would be placed based on its original spectral efficiency value corresponding to a given sub-channel. Figure 3.12 depicts the average number of buckets skipped per flow per sub-channel due to the urgency threshold th and urgency factor γ. For example, given 80 *RT* flows, the average number of buckets skipped per flow per sub-channel is obtained as 0.464 when the th and γ values are equal to 0.005 and 3.5 respectively. It may be noted from Figure 3.12 that when the th value equals to -0.04, the average number of buckets skipped per flow per sub-channel is zero because no flows ever become *urgent* in this case. Table 3.4 shows the results for the performance metrices *fairness index* and *spectral efficiency* (defined in section 3.3.1.1, page 66) for three different values of γ ($\gamma = 2.5$, 3.0, 3.5 has been considered for $th = 0.005$). The table also shows the results for $th = -0.04$.

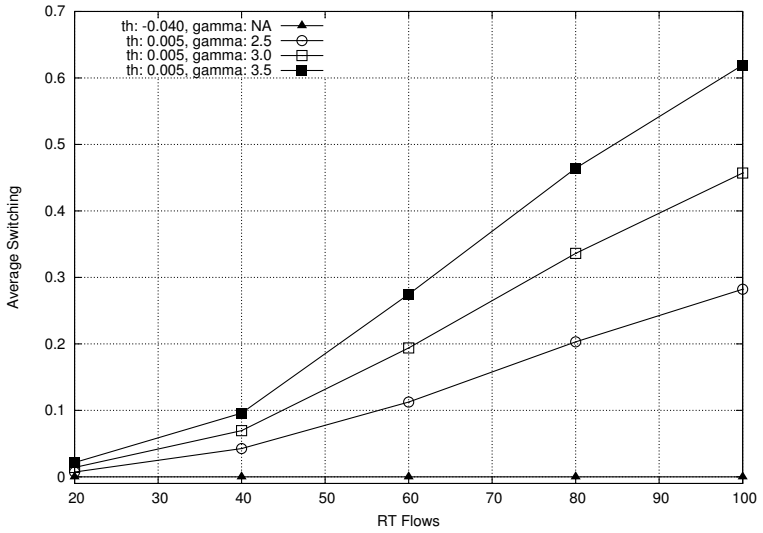


Figure 3.12: *Average number of bucket skipping per sub-channel per flow per frame due to th and* γ

This prioritization of urgent flows due to γ in our bucket allocation mechanism also indirectly helps to improve the fairness in the amount of transmitted data among *RT* flows. Table 3.4 shows that the fairness index (based on Jain's fairness index) achieved by TLS increases with increasing th and γ values. This happens because prioritization of urgent flows indirectly also promotes starved *RT* flows. At the same time, as it may also be observed from Table 3.4, that as a natural consequence of allocating higher priority buckets to urgent flows, which may be suffering from low *CQI* feedbacks, the average spectral efficiency drops as th and γ increases. It may be noted that when $th = -0.04$, there is no impact due to the variation of γ as flows do not becomes urgent. In this case, as spectral efficiency is the only prioritization factor in the selection of flows for *FLRA*, the spectral efficiencies achieved are the highest and correspondingly the fairness indices obtained are the lowest.

Impact due to variation in total number of buckets ($|B|$):

In the *FLRA* process, corresponding to each sub-channel at the beginning of a frame, all selected flows are partitioned into a constant number of buckets B,

Table 3.5: *Comparative results for PLR and Spectral Efficiency with varying $|B|$ values*

RT	PLR				Spectral Efficiency			
Flows	$\|B\|$= 1	$\|B\|$= 5	$\|B\|$= 10	$\|B\|$= 15	$\|B\|$= 1	$\|B\|$= 5	$\|B\|$= 10	$\|B\|$= 15
20	0.0409	0.0158	0.0147	0.01431	0.5016	0.7093	0.7195	0.72512
40	0.2574	0.0514	0.0461	0.04721	0.4881	0.6847	0.7264	0.73621
60	0.4722	0.1336	0.1274	0.12482	0.4554	0.835	0.8573	0.86519
80	0.5682	0.2092	0.1977	0.19507	0.4938	0.9637	0.9922	1.0013
100	0.6634	0.2905	0.2833	0.28096	0.481	1.0451	1.0718	1.07448
Spectral Efficiency in Bits/sec/Hz								

based on their metric values for that sub-channel. Table 3.5 shows the results for the *PLR* and spectral efficiency achieved by *RT VBR* flows for four distinct values of the total numbers of buckets ($|B|$ = 1, 5, 10, 15 have been considered). It may be observed from the table that TLS performs poorly (high *PLR* and low spectral efficiency) when $|B| = 1$. However as $|B|$ increases, the performance is improved as *FLRA* is able to obtain a better classification of flows based on metric values and thereby achieve better relative prioritization among them. Obviously, the complexity of bucket sorting increases as the number of available buckets increases. It may be noted from Table 3.5 that the performance gain obtained by increasing the number of buckets is not linearly proportional as $|B|$ increases from 1 to 15. For example, the performance gain obtained by incrementing $|B|$ from 1 to 5 is quite significant; but this is not so when $|B|$ is increased from 10 to 15. Values of $|B|$ at around 10 has been seen to produce good results in almost all cases.

3.3.1.2 Trade-off Between Scheduling Accuracy Versus Frame Duration

In this section, we first evaluate the performance of TLS at different fixed frame size values ($|F|$ = 1 ms, 10 ms, 15 ms has been considered). Table 3.6 (for RT flows) and 3.7 (for *NRT* and RT flows) shows the *PLR* and spectral efficiency values as a function of the number of flows with different frame sizes. It may be observed that although, as expected, TLS' performance degrades with increasing

Table 3.6: *Comparative results for packet loss rate with varying frame sizes*

RT Flows	PLR					
	TLS			FLS	EXP rule	LOG rule
	$\|F\|$ =1	$\|F\|$ =10	$\|F\|$ =15			
20	0.002	0.006	0.132	0.004	0.010	0.166
40	0.024	0.050	0.177	0.035	0.141	0.358
60	0.110	0.159	0.256	0.144	0.326	0.487
80	0.176	0.232	0.309	0.231	0.439	0.552
100	0.258	0.317	0.386	0.330	0.531	0.614

Table 3.7: *Comparative results for spectral efficiency with varying frame sizes*

RT + non-RT Flows	Spectral Efficiency					
	TLS			FLS	EXP rule	LOG rule
	$\|F\|$ =1	$\|F\|$ =10	$\|F\|$ =15			
40	0.836	0.661	0.649	0.893	1.027	1.068
80	0.837	0.783	0.764	0.869	1.038	1.097
120	0.910	0.829	0.765	0.854	1.007	1.069
160	1.046	0.950	0.859	0.892	1.011	1.061
200	1.134	1.021	0.920	0.854	0.957	1.012

frame sizes, the degradation rate is not drastic. In fact, its performance even at 10 ms is at par with the other schedulers. This indicates that scheduling granularities considerably higher than 1 ms may be safely used by TLS in most realistic scenarios to allow lower computational overheads (as discussed in section 3.2.6 (Computational Overhead)).

Given an expected threshold PLR (PLR^{th}) that the system requires to satisfy at any time, TLS includes a dynamic frame adjustment procedure (described in section 3.2.7) which attempts to maintain just as much frame rate as is essential to keep PLR bounded within PLR^{th}. The aim of this procedure thereby, is to contain the computational overhead incurred by the *TLS* while allowing only a bounded performance degradation. In this section, we explore the impact of the variation of system load (in terms of the number of RT flows) on frame sizes achieved and corresponding PLRs, for different PLR threshold bounds. Figures. 3.13 and 3.14 portrays receptively the plots for PLR achieved and the aver-

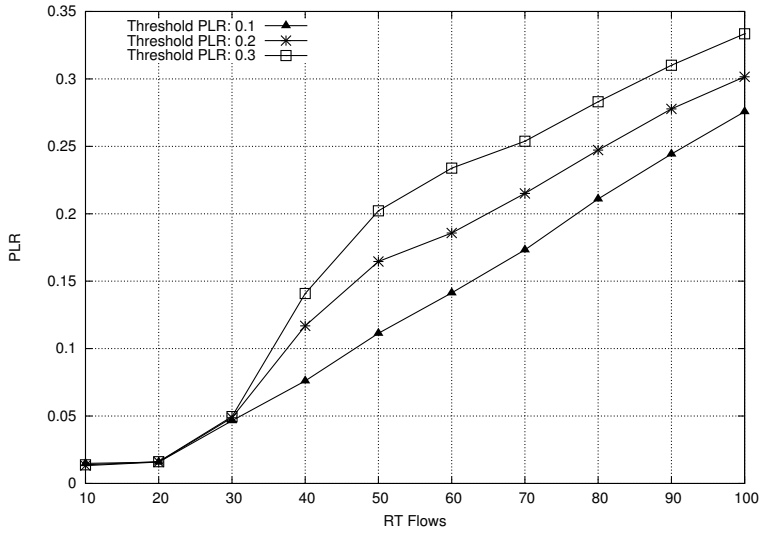


Figure 3.13: *PLR Vs. #RT Flows*

age frame duration over the entire simulation span for three distinct PLR thresholds (PLR^{th}= 0.1, 0.2 and 0.3, refer Algorithm 3), as the number of RT flows varies from 10 to 100. The constants z, PLR^{gd}, $|F|_{min}$ and $|F|_{max}$ in Algorithm 3 have been set to 10, 0.01, 4 ms and 20 ms, respectively. As expected, when PLR^{th} is relaxed, the average frame duration obtained (in Figure 3.14) increases (which allows a reduction in computational overhead), however the PLR suffered (in Figure 3.13) also increases simultaneously.

From the above discussion it may be observed that: (i) A value for $|B|$ in the range of $\sim$ 5 to 10 may be considered to be suitable in most scenarios. In the work, we have used $|B| = 10$ in the experiments conducted with the TLS framework. The comparative results have been quite handsome in favor of TLS. (ii) With $|F| = 10$ms, performance of TLS is almost at par with its nearest competitor FLS. In this work, we have used $|F| = 5$ms for the comparative analysis of TLS and obtained better performance with respect to other schedulers in most cases. This indicates that scheduling granularities considerably higher than 1 ms may be safely used by TLS in most realistic scenarios to allow lower computational overheads (as discussed in section 3.2.6). (iii) Given an expected workload and

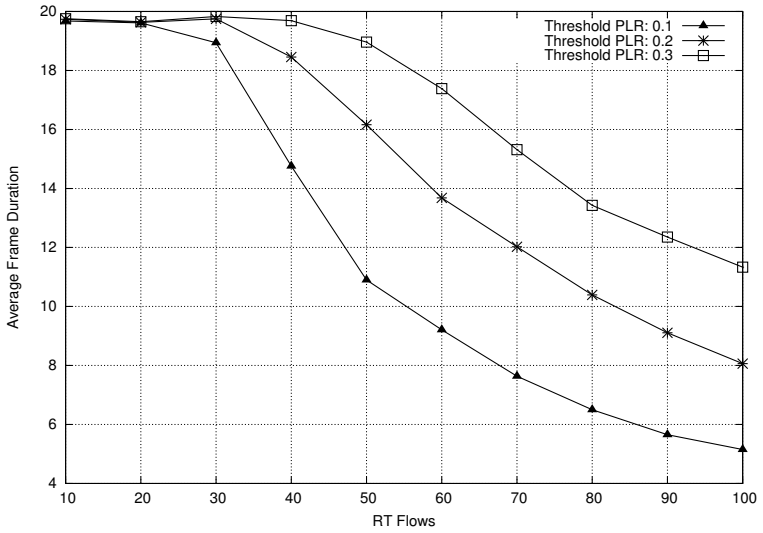


Figure 3.14: *Average Frame Duration*

required performance demand, the value of the *threshold PLR* (PLR^{th}) may be chosen appropriately and (iv) The values of th and γ depend on the types of traffic as well as the targeted services (fairness index Vs spectral efficiency) being handled by an operator must be adjusted according to the scenario at hand.

3.3.1.3 Comparative results for average execution time (in millisecs)

In this subsection, we have evaluate the average execution times of the *TLS*, *FLS*, *LOG-Rule* and *EXP-Rule* strategies. Figure 3.15 depicts the average run time per *TTI* taken by the various schedulers with the number of *UEs* varying from 10 to 100. It may be observed that the split architecture based strategies (*TLS* and *FLS*) take less execution time compared to conventional scheduling strategies (*EXP-Rule* and *LOG-Rule*; where all the scheduling decisions taken at each *TTI* boundaries). *FLS* takes scheduling decisions at two distinct time granularities by considering real-time flows and non-real time flows separately and thus, have less average run time compared to *EXP-Rule* and *LOG-Rule*. *TLS* takes the lowest execution time due to its temporally hierarchical multilevel scheduling infrastructure. The multilevel split architecture allows *TLS* to appropriately

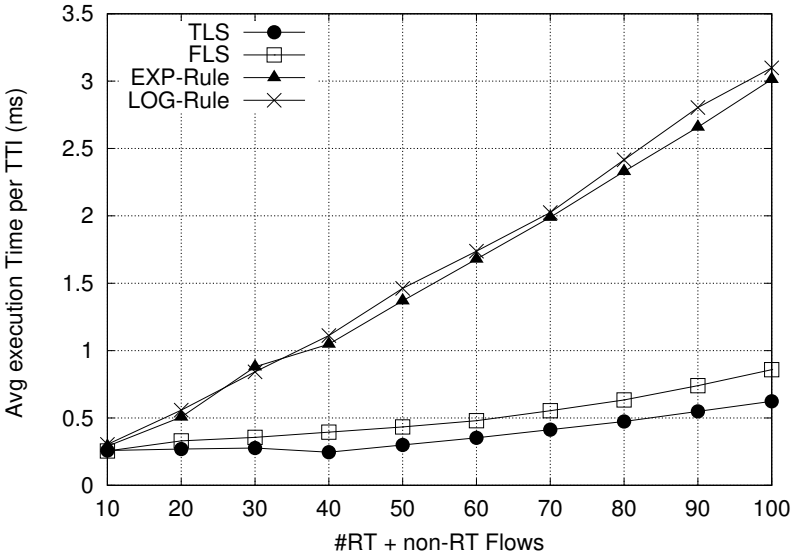


Figure 3.15: *Comparative results for average execution time (in millisecs)*

set the frequency at which different components of the scheduler like "traffic prediction for each real time flow", "cell capacity estimation", "resource block allocation" etc. should be performed.

Next section presents a detailed description of our second resource allocation strategy known as *Hybrid Resource Allocation Framework*.

3.4 Hybrid Resource Allocation Framework

The *Hybrid Resource Allocation Framework (HRAF)* is aimed at enabling *LTE* service providers to efficiently achieve good *QoS* while incurring low overall scheduling overheads. As depicted in Figure 3.16, *HRAF* has been designed as a hybrid architecture which integrates offline and online techniques together in order to allocate resource blocks toRT flows. The objective of the offline phase is to maintain system load within a given threshold value in each scheduling interval (or *TTI*). The online resource allocation scheme runs on top of the offline policy and conducts the physical mapping of one or more RBs to a set of selected flows at any given *TTI*. This online RB to flow mapping procedure captures in-

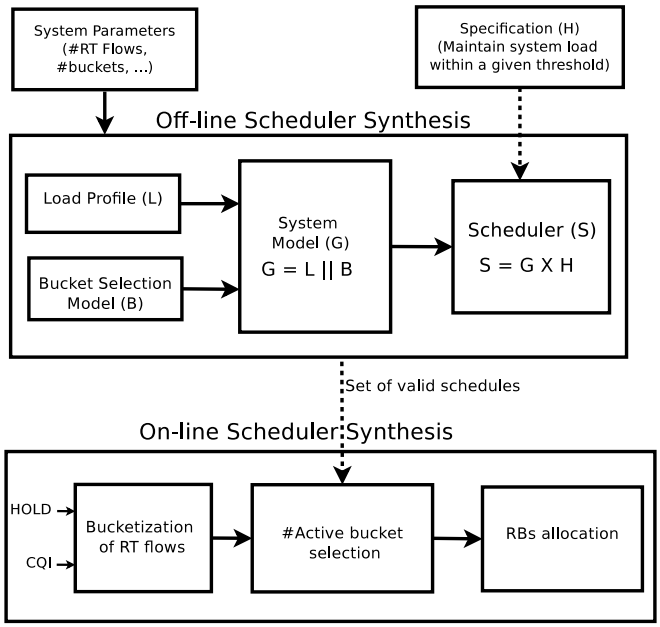


Figure 3.16: *Hybrid Resource Allocation Framework*

Table 3.8: *Notations*

Variable	Explanation
$L_b(t)$	System load at time instant t
$C(t)$	System capacity at time instant t
$B_k(t)$	Total number of buckets selected for RB allocation at time instant t
AB	Total number of available buckets for flows classification
th	Safe operating threshold value (measured in % of $C(t)$)

herent variability in system parameters by considering instantaneous CQI of the flows, instantaneous system capacity, actual end-to-end delay bounds of RT flows etc. in an endeavour to maximize achieved QoS. Now, we present the detailed description of each scheduling phase.

3.4.1 Offline Supervisor

Typically, the downlink scheduling infrastructure in LTE deals with the physical allocation of available RBs among active flows such that the QoS demand of each flow is satisfied. However, it may not be feasible to satisfy the demand of all flows

at all times due to the limited available bandwidth [79]. A typical instantaneous system load profile at eNodeB ($L_b(t)$) is depicted in Figure 3.21 (refer Section 4.3). It may be observed from the figure that the system load goes beyond 100% of the available capacity for a significant length of time. Such load profile needs to be managed so that at any time t, $L_b(t)$ can stay within its allowable limit (which is also known as safe operating threshold and denoted by th) for a given instantaneous available capacity $C(t)$. In order to maintain $L_b(t)$ within the given threshold th, there is a need to develop a mechanism which will select and remove some of the less critical active flows temporarily until the system recovers to the manageable load condition. The decision to removing a particular set of active flows can actually be decided online. However, the average number of flows that requires to be removed so that system load may be maintained within stipulated threshold th, can be pre-computed offline to minimize online overheads. Therefore, the objective of the offline scheduler can be stated as follows: *Given $L_b(t)$, $C(t)$ and th, design a supervisor which decides the number of active flows to be selected for RB allocation such that the load induced by the selected flows stays within the safe operating threshold (th) in each scheduling interval.*

In the endeavor to achieve the objective mentioned above, *HRAF* classifies the flows into constant number of priority buckets. The priority bucket to which a particular flow is assigned depends on a combination of the flow's head of line delay (*HOLD*) and its *CQI*. Information about the number of *RT* flows, number of buckets and specification of the safe threshold limit (th) are fed as input to the offline supervisor synthesis mechanism. The synthesis supervisor selects the appropriate number of buckets so that $L_b(t)$ remains within the threshold th at any given *TTI* under consideration. However, a good Such a selection strategy must consider $L_b(t)$, $C(t)$, possible urgencies of flows, potential channel condition of each flow/UE, etc. All possible combinations of the above mentioned parameters must be analyzed in order to obtain a good solution for the problem under consideration. This requirement can be guaranteed by enumeration techniques which can explore the entire solution space. *Supervisory Control Theory* of *Discrete*

Event Systems (*SCT* of *DES*) is an important formal state-space enumeration mechanism. A SCT of DES based design enjoy the advantage that being based on formal languages and automata theory the resulting supervisor is guaranteed to be *correct-by-construction*. In this work, we formulate the problem of selecting the number of priority buckets using the *SCT* of *DES* framework.

Figure 3.16 depicts the steps involved in the synthesis of an offline supervisor (denoted by S) using *SCT* of *DES*. It starts with the modeling of individual components of the system. Since the objective is to control $L_b(t)$ by selecting an appropriate number of buckets $B_k(t)$, the discrete event models for $L_b(t)$ and $B_k(t)$ are constructed. The composite system model G is derived through the parallel composition over these individual models. Correspondingly, we construct model H which provides the specification for the legitimate number of buckets for different load conditions. Finally, the supervisor S is obtained using the product composition of G and H. With this overview, let us proceed towards the offline supervisor synthesis by constructing the automata for individual components in the system.

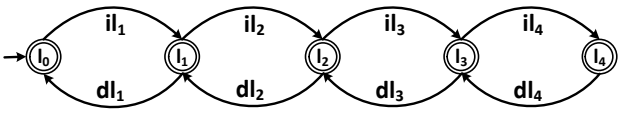


Figure 3.17: *Model L for System Load at eNodeB.*

Modeling $L_b(t)$: In order to model $L_b(t)$, we need to identify the set of possible values that can be taken by $L_b(t)$. It may be noted that the range of values taken by $L_b(t)$ typically varies between 50% and 200%. To represent this continuous range using DES, we divide this entire interval into a discrete number of sub-intervals, each of which represents a distinct range of values. In the model in Figure 3.17, we have divided the range of instantaneous load values into five sub-intervals each of which is denoted by a distinct state in the model. It is obvious that, more the number of sub-intervals into which the range of $L_b(t)$ is divided, finer will be the representation of the system load, but correspondingly

larger will be the state-space generated by the resulting model. The discrete event model L representing different instantaneous system load conditions ($L_b(t)$) at eNodeB (shown in Figure 3.17) is defined as follows:

$$L = (Q_L, \Sigma_L, l_0, Q_{mL}, \delta_L, \Gamma_L),$$

where, $Q_L = \{l_0, l_1, l_2, l_3, l_4\}$. A state in Q_L depends on the instantaneous load and is defined as:
l_0: $L_b(t) \leq th$,
l_1: $th < L_b(t) \leq 120\%$ of th,
l_2: 120% of $th < L_b(t) \leq 140\%$ of th,
l_3: 140% of $th < L_b(t) \leq 160\%$ of th,
l_4: 160% of $th < L_b(t)$,
Here, all states are *marked*, i.e., $Q_{mL} = Q_L$, since the instantaneous load may take the system to any of these states. The event set $\Sigma_L = \{il_1, il_2, il_3, il_4, dl_1, dl_2, dl_3, dl_4\}$. A description of the events in Σ_L are:
il_1: $L_b(t)$ exceeds 100% of th,
il_2: $L_b(t)$ exceeds 120% of th,
il_3: $L_b(t)$ exceeds 140% of th,
il_4: $L_b(t)$ exceeds 160% of th,
dl_1: $L_b(t)$ falls below 100% of th,
dl_2: $L_b(t)$ falls below 120% of th,
dl_3: $L_b(t)$ falls below 140% of th,
dl_4: $L_b(t)$ falls below 160% of th.
The instantaneous system load $L_b(t)$ is measured based on two parameters: (i) Aggregate data-rate demands from all flows, (ii) Instantaneous cell capacity. It may be noted that the flows which we have considered in this work are *real-time* and may have *variable bit-rate* demands. This makes the resource demands of individual flows variable and beyond the control of the supervisor. On the other hand, given an available bandwidth, instantaneous cell capacity varies depending

on CQIs experienced by the user equipments at a given time. Thus, variability in overall available cell capacity is also beyond the control of the supervisor. Consequently, the values taken by $L_b(t)$ are induced by the operating environment and are *uncontrollable*. Therefore, all events Σ_L are modeled as *uncontrollable*.

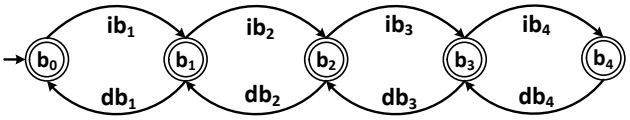


Figure 3.18: *Model B for #Buckets Selected for RB allocation.*

Modeling $B_k(t)$: The discrete event model B representing the number of buckets to be selected for RB allocation is shown in Figure 3.18. Model B is defined as follows:

$$B = (Q_B, \Sigma_B, b_0, Q_{mB}, \delta_B, \Gamma_B),$$

where $Q_B = \{b_0, b_1, b_2, b_3, b_4\}$, $\Sigma_B = \{ib_1, ib_2, ib_3, ib_4, db_1, db_2, db_3, db_4\}$. The states in Q_B may be described as:
b_0: $B_k \leq 20\%$ of AB,
b_1: 20% of $AB < B_k \leq 40\%$ of AB,
b_2: 40% of $AB < B_k \leq 60\%$ of AB,
b_3: 60% of $AB < B_k \leq 80\%$ of AB,
b_4: 80% of $AB < B_k \leq 100\%$ of AB,
Here, $B_k(t)$ represents the total number of buckets to be selected for RB allocation at eNodeB and AB represents the total number of available buckets. All states are *marked*, i.e., $Q_{mB} = Q_B$, since any number of buckets can be selected for the RB allocation in a given scheduling interval. The description of events in Σ_B is as follows:
ib_i: the number of selected buckets becomes greater than $3i$, where $i = 1, 2, 3, 4$.
db_i: the number of selected buckets becomes less than or equal to $3i$, where $i = 1, 2, 3, 4$.

Since, the scheduler has the flexibility to restrict the total number of buckets that may be used for RB allocation, all events in Σ_B are modeled as *controllable*.

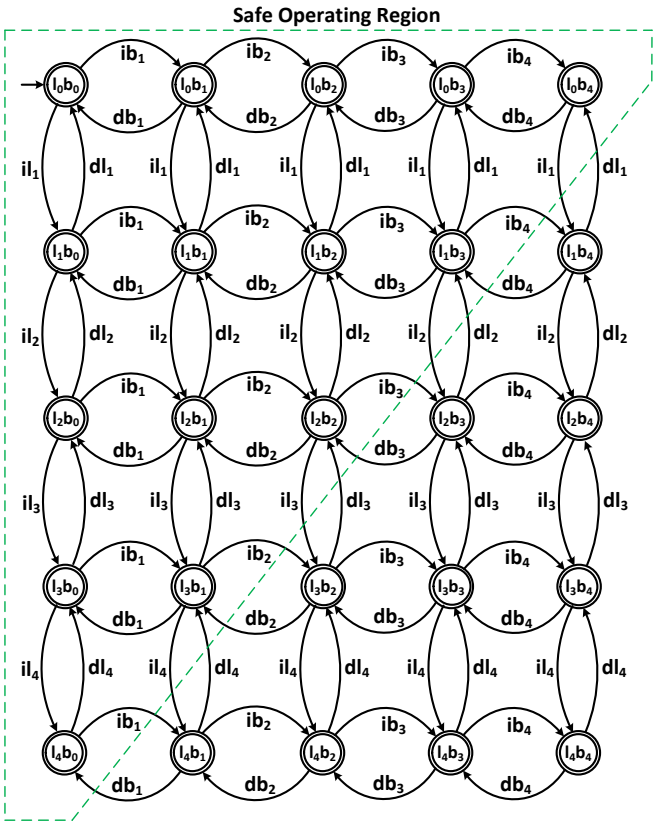


Figure 3.19: *System model* $G = L||B$.

Composite System Model G: Given the models for individual system components, i.e., L and B, the composite system model G is obtained using parallel composition and it is shown in Figure 3.19. It may be observed that G contains all possible states in which the system may be present during online operation. That is, *State* l_0b_0 represents the *underloaded* scenario in which $L_b(t)$ is less than th and less than 20% of the available buckets are being used. On the other hand, *State* l_0b_4 represents the *underloaded* scenario in which at least 80% of the available buckets are under consideration. Similarly, *State* l_4b_4 represents the scenario in which at least 80% of the buckets are in use in a highly *overloaded* situation ($L_b(t) > 160\%$ of th). In addition, G is *non-blocking*, i.e., any state

Table 3.9: *Specification for bucket selection*

Instantaneous System Load ($L_b(t)$)	**Number of Selected Buckets** ($B_k(t)$)
$L_b(t) \leq th$	$B_k(t)$ is equal to 100% of AB
$th < L_b(t) \leq 120\%$ of th	$B_k(t)$ cannot exceed 80% of AB
120% of $th < L_b(t) \leq 140\%$ of th	$B_k(t)$ cannot exceed 60% of AB
140% of $th < L_b(t) \leq 160\%$ of th	$B_k(t)$ cannot exceed 40% of AB
160% of $th < L_b(t)$	$B_k(t)$ cannot exceed 20% of AB

in G can be reached from any other state in G. Hence, *the system behavior* $L_m(G)$ *contains all possible sequences that allow the selection of any number of buckets irrespective of system load conditions* (within defined limits). Suppose G is at *State* l_4b_0, the sub-string $ib_1ib_2ib_3ib_4$ allows the potential selection of all the available options for the number of buckets.

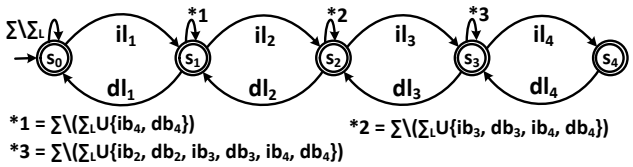


Figure 3.20: *The specification model H*

Modeling Specification: In order to maintain system load within the given threshold value (th), we develop the specification model which enforces selection of the appropriate number of buckets in each scheduling interval. In this work, we consider the specification shown in Table 3.9 and its corresponding discrete event model H is shown in Figure 3.20. At *State* S_0 of H, the self-loop $\Sigma \setminus \Sigma_L$ contains the event set Σ_B which allows the selection of any number of buckets. The transition on event il_1 takes H from *State* S_0 to *State* S_1. At this state, the self-loop ^{*1} $(= \Sigma \setminus (\Sigma_L \cup \{ib_4, db_4\}))$ restricts the number of buckets to be upper bounded by 80% of AB. When $L_b(t)$ goes above 120% of th and remains below the 140% mark, transition on event il_2 takes H from *State* S_1 to *State* S_2. Here, the self-loop ^{*2} $(= \Sigma \setminus (\Sigma_L \cup \{ib_3, db_3, ib_4, db_4\}))$ restricts the number of buckets to be not more than 60% of AB. After $L_b(t)$ crosses the 140% mark but remains

below 160% of th, transition on event il_3 takes H from S_2 to S_3. Here, the self-loop ^{*3} $(= \Sigma \setminus (\Sigma_L \cup \{ib_2, db_2, ib_3, db_3, ib_4, db_4\}))$ ensures the number of selected buckets to be less than 40% of AB. For $L_b(t)$ beyond 160% of th, il_4 takes H from S_3 to S_4. The absence of any self-loops at this state guarantees that the number of selected buckets is limited to 20% of AB. Following a similar sequence of transitions as discussed above corresponding to the progressive increase in $L_b(t)$, H moves to appropriate states based on decrease in $L_b(t)$.

It may be observed that the marked behavior $L_m(H)$ restricts the total number of buckets that may be selected corresponding to any potential system load condition. Therefore, a product composition $S = G \times H$ restricts the system model G to select the appropriate number of buckets corresponding to system loads. This model S acts a supervisor of the system (shown using dotted lines in Figure 3.19). Now, we have to ensure that the synthesized scheduler is controllable. Suppose s denotes a sequence of events encountered by the system G under control of the supervisor S. Then, the next event (say, σ) must always be allowed if it belongs to the set of uncontrollable events, i.e., $s \in \overline{L_m(G)}$, $s \in \overline{L_m(H)}$ and $\sigma \in \Sigma_{uc}$, then $s\sigma \in \overline{L_m(G)}$ implies $s\sigma \in \overline{L_m(H)}$. Otherwise, S is not controllable. It may be noted that in our proposed framework, $\Sigma_{uc} = \Sigma_L$.

Theorem 1: *The offline supervisor S is not controllable.*

Proof: In order to verify the controllability of S, let us consider the scenario in which the system starts its operation from the initial state (i.e., l_0b_0 of G). Since $L_b(t) \leq C(t)$, the scheduler S does not restrict the usage of all available buckets. Suppose all buckets have been used, then G will reach *State* l_0b_4 by traversing on the events $ib_1ib_2ib_3ib_4$ (denoted by seq). Here, it may be observed that $seq \in \overline{L_m(G)}$ as well as $seq \in \overline{L_m(H)}$. Due to the usage of all buckets, the instantaneous system load $L_b(t)$ may exceed $C(t)$, i.e., G will move out of the safe operating region defined by the specification. Such a violation cannot be controlled by the supervisor S since the occurrence of events in Σ_L cannot be prevented by the supervisor. For example, the extension of sequence seq $(=ib_1ib_2ib_3ib_4)$ by il_1 cannot be restricted by the supervisor since $il_1 \in \Sigma_{uc}$. Such a extension leads

to the scenario in which $ib_1ib_2ib_3ib_4il_1 \in \overline{L_m(G)}$, but $ib_1ib_2ib_3ib_4il_1 \notin \overline{L_m(H)}$. Therefore, the supervisor S is not controllable. □

Since the supervisor S is not controllable, it cannot provide a guarantee that the system load will always be maintained within the threshold th. Although the supervisor S does not have direct control over $L_b(t)$, it has control over the number of buckets to be selected at any scheduling interval. Whenever the system G moves out of the safe operating region (shown using dotted lines in Figure 3.19), the supervisor S can bring G back to the safe region by decrementing the number of buckets that are allowed. That is, $ib_1ib_2ib_3ib_4il_1 \notin \overline{L_m(H)}$, but $ib_1ib_2ib_3ib_4il_1db_4 \in \overline{L_m(H)}$. It may be noted that the decision on increasing / decreasing the total number of buckets selected for RB allocation can be taken only at TTI boundaries. Hence, the supervisor S cannot instantaneously change $B_k(t)$. Therefore, the scheduler will control $L_b(t)$ within the safe threshold th only at TTI boundaries. In order to ensure such control, the supervisor S partitions the state set Q of G into two disjoint sets, i.e., $Q = Q_s \cup Q_u$. Here, Q_s denotes the set of safe states in G and Q_u denotes the set of unsafe states in G. Using this information, the scheduler will take necessary corrective action online to bring the system back to safe operation region whenever it moves to the unsafe region.

3.4.2 Online Scheduling

The online scheduler runs on the top of the offline policy in order to capture the inherent variability in the system. Additionally, the online scheduler also bring the system G back to the safe region (by decrementing the number of buckets that are allowed during RB allocation.) whenever G moves out of the safe-operating zone (shown using dotted lines in Figure 3.18). For example, let us consider a system with state Q_t being its current state at the start of the t^{th} TTI. Q_t belongs to the set of safe states Q_s. Subsequent to the elapse of the t^{th} TTI it may so happen that the system state gets changed to q_{t+1} from q_t, where Q_{t+1} belongs to the set of unsafe states Q_u. In such a scenario, the system state will

be revised/re-initialized to its nearest safe state by the online scheduler [80, 81]. A step wise description of the online scheduler is presented in Algorithm 4. The proposed scheduler dynamically allocates resource blocks to the *RT* flows, taking into account the instantaneous channel quality index, system load and maximum end-to-end tolerable delays of *RT* flows.

In the first step, the scheduler calculates the key values for all flows in order to uniformly divide them into available number of buckets (AB). The key values for all the flows are assigned on the basis of two parameters: (a) Spectral efficiency of the flow based on its wide-band *CQI* (b) *Head-of-line Delay (HOLD)* which represents the urgency of the flow. The key value key_i for the i^{th} flow is calculated as follows:

$$key_i = \begin{cases} SE_i, & \text{if } HOLD < HOLD_i^{th} \\ \frac{\gamma \times HOLD_i}{MaxDelay_i} \times SE_i, & \text{Otherwise} \end{cases} \tag{3.14}$$

where, $HOLD$ is the head-of-line packet delay, SE_i denotes the spectral efficiency corresponding to the i^{th} flow based on wide band *CQI* and $MaxDelay_i$ represents the upper bound on the time a flow should wait before being dropped from the queue ($MaxDelay$ provides a measure of a flow's criticality towards real-time packet delivery). $HOLD_i^{th}$ and γ are the tunable parameters which indicate the urgency factor and urgency threshold, respectively. Larger the values of $HOLD_i^{th}$ and γ, higher becomes the relative priority of packet delay urgencies compare to spectral efficiency.

In the second step, the scheduler sorts all the flows in non-decreasing order based on the computed key values and divides them uniformly into the priority buckets. That is, the flows having higher key values are enqueued into the higher priority buckets. Then, the scheduler computes the instantaneous system load $L_b(t)$ at eNodeB in step 3. In order to maintain the system load within the given safe threshold value (th), the online scheduler selects the appropriate number of buckets with the help of the offline supervisor. For this purpose, it invokes the function `Update-System-State` $(q_{t-1}, L_b(t))$ (refer step 4) to identify the current system state. Here, q_{t-1} represents the system state at the end of last

ALGORITHM 4: The Online Scheduler

```
1  Calculate key value for each active flow;
2  Allocate flows uniformly among the available buckets based on their key
     values;
3  Compute the current system load L_b(t);
4  q_t = Update-System-State (q_{t-1}, L_b(t));
5  Select the appropriate number of buckets B_k(t) for RB allocation based on
     the obtained q_t;
6  Compute the RBs demand D^b_RB(t) of the b^th bucket;
7  Compute the total RBs demand over all selected buckets as:
     D^total_RB(t) = sum_{b in B_k(t)} D^b_RB(t);
8  Let C(t) be the instantaneous system capacity available at eNodeB;
9  if C(t) > D^total_RB(t) then
10     while (C(t) - D^total_RB(t)) > 0 do
11         Update B(t) = B(t) + 1 ;
12         Update D^total_RB(t) based on current B(t) ;
13 else
       /* System is overloaded condition                              */
14     Fairly distribute the available capacity C(t) among selected buckets;
15 Allocate RBs to each flow which belongs to the selected buckets B_k(t) ;
```

TTI (scheduling interval) and a check is made to determine whether it is part of the set of unsafe states $Q_u \in G$. Step 1 of Algorithm 2 initializes q_{t-1} to the temporary variable q_{temp}. Suppose $q_{temp} \in Q_u$, then q_{temp} is appropriately adjusted by taking the transition on events of type db_j (where $j = \{1, 2, 3, 4\}$) (refer to steps 3 to 5) until q_{temp} becomes the part of the set of safe states Q_s. On the other hand, $q_{temp} \in Q_s$, then q_{temp} is appropriately adjusted by taking the transition on events of type ib_j (where $j = \{1, 2, 3, 4\}$) (refer to steps 7 to 8) until q_{temp} reaches the boundary of safe operating region. Finally, q_{temp} has been updated as q_t (step 9, Algorithm 2) and the total number of buckets $B_k(t)$ allowed at this state is given as an input to the online scheduler.

It may be noted that based on the solution provided by the offline policy, a certain number of buckets $B_k(t)$ is selected based on instantaneous system state

ALGORITHM 5: Update-System-State

Input: System state in last scheduling interval (q_{t-1}), Instantaneous System Load ($L_b(t)$), Composite System Model $G = (Q,\ \Sigma,\ q_0,\ Q_m,\ \delta,\ \Gamma)$

Output: Current System State (q_t)

`/* Compute` q_t `from` q_{t-1} `s.t.` $q_t \in Q_s$ `*/`

1 Initialize $q_{temp} = q_{t-1}$;
2 **if** $q_{temp} \in Q_u$ **then**
3 **repeat**
4 $q_{temp} = \delta(q_{temp}, db_j)$, where $db_j \in \Gamma(q_{temp})$;
5 **until** $q_{temp} \in Q_s$;
6 **else**
7 **while** $\delta(q_{temp}, ib_j) \in Q_s$ **do**
8 $q_{temp} = \delta(q_{temp}, ib_j)$, where $ib_j \in \Gamma(q_{temp})$;
9 $q_t = q_{temp}$;
10 **return** q_t

at a given time. However, inherent system dynamics such as instantaneous RB demands, instantaneous sub-band channel conditions, etc. of each flow have not been considered in the offline supervisor. Such inefficient utilization of the above mentioned dynamic parameters may result in poor resource utilization and may lead to performance degradation for the *RT* flows beyond a certain system load value. Therefore in steps 9 to 15 (Algorithm 1), the online scheduler appropriately alters the offline decisions at each *TTI* to improve resource utilization. In order to carryout this modification, the online scheduler must consider both underutilization as well as over-utilization of resources. The underutilization occurs when the total RB demand over all the selected buckets (calculated offline) is less than the available instantaneous capacity $C(t)$. In such a situation, the algorithm increments the total number of selected buckets by one until the total RB demand over all selected buckets is approximately equal to the instantaneous capacity (In steps 9 to 12). On the other hand, if the total RB demand over all selected buckets overshoots the instantaneous capacity ($C(t)$), the online scheduler fairly scales down the total #allocated RBs for each bucket in step 14. The

number of allocated RBs $A^b_{RB}(t)$ for the b^{th} bucket is calculated as:

$$A^b_{RB}(t) = D^b_{RB}(t) \times \frac{C(t)}{D^{total}_{RB}(t)} \tag{3.15}$$

where, $D^b_{RB}(t)$ denotes the RB demand of the b^{th} bucket, $D^{total}_{RB}(t)$ denotes the aggregate RB demand over all selected buckets and $C(t)$ represents the instantaneous capacity.

Each bucket (having its corresponding allocated bandwidth chunk) must divide the resource among their flows. Physical allocation of RBs is done by the online scheduler in step 15. Resource allocation progresses one bucket at a time and goes from higher priority buckets to lower priority buckets. The scheduler allocates the available RBs among flows based on the comparison of per-RB metrics. For a given RB (say RB_r), a currently unallocated flow (say f_i) is selected for allocation, if the metric value m_{ir} for this flow-RB pair is maximum. That is:

$$m_{ir} = \mathbf{max}_k \ \{m_{kr}\} \ \ \forall k \in \{1, N_b\} \tag{3.16}$$

where, N_b denotes the total number of flows available in the b^{th} bucket. m_{ir} can be interpreted as the transmission priority of each flow on a specific RB. The metric value (m_{ir}) for the i^{th} flow on the r^{th} RB is calculated based on two parameters: (i) spectral efficiency of the i^{th} flow on the r^{th} RB (or sub-channel) and (ii) urgency of the i^{th} flow based on its head-of-line delay. In this work, the same equation (equation 3.14) which was used to calculate key values during flow prioritization for bucket allocation has been used to generate m_{ir} values. However in case of metric value generation for flow-RB pairs, sub-band CQIs have been used instead of wide-band CQIs (which was used for the calculation of key values).

3.5 Experiments and Results

A set of experiments have been conducted in order to measure the performance of the *HRAF* resource allocation framework. The performance metrics which

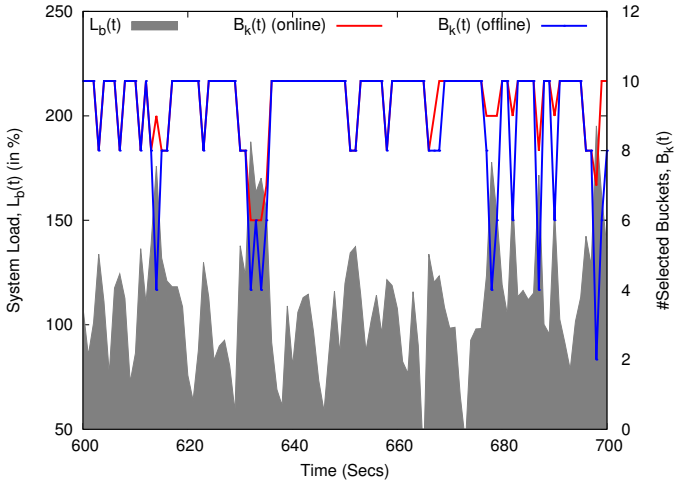


Figure 3.21: *Variation in $B_k(t)$ against $L_b(t)$*

have been considered for evaluation are: (i) #selected buckets with system load variation and (ii) *Packet Loss Rate (PLR)*.

Figure 3.21 depicts the instantaneous system load along with the number of buckets selected (offline and online) by the scheduler at eNodeB. At *TTI* 600, we can observe that $L_b(t)$ is within 100% and consequently, offline scheduler selects all available buckets for RB allocation. Since, the maximum number of buckets have already been selected by the offline scheduler, there is no scope for online improvement. After about 5 TTIs, $L_b(t)$ goes beyond 100% and during the ensuing *TTI*, offline scheduler reduces the number of buckets $B_k(t)$ to 8. As an illustration of the online moderation conducted by *HRAF* over obtained offline solutions, we may observe the online up-gradation of $B_k(t)$ at *TTI* 615. Here, $L_b(t)$ is around 175% and therefore, our offline scheduler selects 40% of the available buckets (i.e., 4). However, the online scheduler analyzes that the resource utilization with the selected buckets is less than 100% and subsequently, it increases $B_k(t)$ until the surplus capacity is effectively utilized. Finally, the online scheduler selects 9 buckets. Similar online upgradations can be observed at the *TTIs*: 630, 670, 680, 690, 700 etc. From the above discussion, it may be concluded that offline decision holds for most of the cases. However, in certain

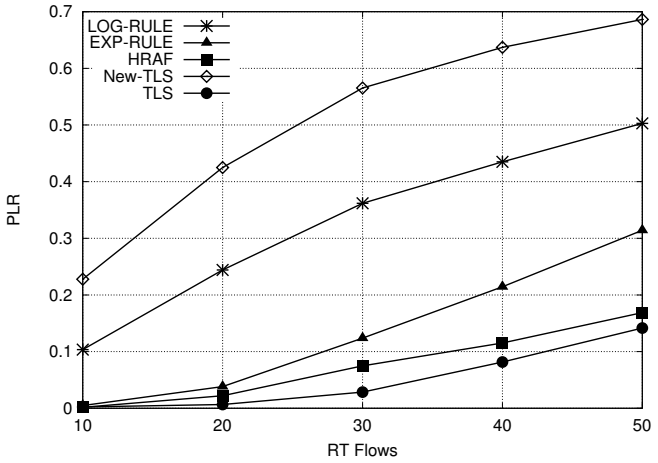


Figure 3.22: *Packet Loss Rate (PLR) Vs. #RT Flows*

scenarios when the instantaneous available resources are not fully utilized by the selected buckets, the online scheduler alters the offline decisions to ensure the effective utilization of system resources.

Figure 3.22 compares the performance of *HRAF* against *New Two Level Scheduler* (*New-TLS*), *LOG-Rule*, *EXP-Rule* and *TLS* as the number of flows vary between 10 and 50. It may be observed that the *HRAF* strategy shows better *PLR* for the *RT* flows as compared to the *LOG-Rule*, *EXP-Rule* and *New-TLS* schedulers. Such better performance may be attributed primarily to the hybrid resource allocation policy embedded at eNodeB which endeavors to maintain the system load within the given safe threshold (th) by selecting an appropriate number of buckets offline and tuning it effectively online (if needed) in each *TTI*. Better performance of the proposed strategy is also achieved through the appropriate prioritization of packet delay urgency over spectral efficiency in the calculation of key values and metric values of the flows, when the urgency crosses a given threshold $HOLD_{th}$. However, as observed from the figure, the performance of *HRAF* is in general slightly poorer than TLS in all scenarios.

3.6 Summary

In this chapter, two new flexible downlink resource allocation frameworks for the *LTE* systems have been presented. The framework is aimed at enabling mobile operators to effectively achieve good *QoS* and cell spectral efficiencies while incurring low overall scheduling overheads. The first scheduling strategy (known as *TLS*) is designed as a three layer split-architecture. At the highest level (super-frame boundaries) of this three layered scheduling framework, an estimation of the amount of incoming data traffic for each active *RT* flow and expected overall cell capacity in the next super-frame, is determined. These estimates are then used to calculate the amount of data to be transmitted for each flow such that proportional fairness in the degree of *QoS* provided to all flows may be maintained. At each frame boundary within a super-frame, a low overhead mechanism is employed to allocate all RBs of the next frame to flows, in three rounds; *RT* flows in the first round, instantaneously arriving *RT* data bursts in the second round, and *NRT* flows in the third round. At each *TTI* in a frame, RBs are physically allocated to the flows by looking-up a RB allocation matrix obtained from the frame level scheduler. The three level scheduling granularity of the proposed work provides flexibility to allow significant reduction in computational overheads while suffering low and bounded degradation in its performance in terms of packet loss rates, spectral efficiency etc., provided the system is not uncontrollably overloaded. Then, we have attempted at a combined hybrid offline-online approach in order to minimize overall resource allocation overheads while maintaining a minimum satisfactory level of *QoS*. The offline strategy is based on *Supervisory Control Theory* of *Discrete Event Systems*. Simulation results reveal that the proposed schemes outperform the existing schedulers in most scenarios.

Chapter 4

Buffer-Aware Resource Allocation for Video Streaming over LTE

4.1 Introduction

Latency sensitive high bandwidth multimedia data is expected to capture more than 70% of the total traffic in *LTE* and future wireless networks. This is expected to pose enormous challenges on the radio resource and management mechanisms in these networks. One of the most important factors that diminish the quality of viewing experience of delivered videos is frequent client side rebuffering events. Two resource allocation strategies for generic real-time flows have been presented in the previous chapter. Both the strategies are able to deliver satisfactory quality of service to the real-time flows. However, client-side buffer awareness has been not considered in their resource allocation policy and this makes them susceptible to frequent buffering events leading to stutters during video playback which ultimately pulls down the quality of viewing experience.

In this chapter, we have presented a downlink resource allocation framework called Buffer-aware Three Level Scheduler (*BA-TLS*) which endeavors to deliver smooth viewing experience to each active end user even during transient network overloads. It may be noted that seamless playout experience in the face of temporally varying wireless bandwidths, may only be ensured by continuously

maintaining playout buffer size above a specific threshold [23]. Founded on the basic architecture of Three Level Scheduler (TLS), *BA-TLS* inherits all its salient facets including low resource block (RB) allocation overheads, minimum guaranteed delay bounds for the flows, high spectral efficiency etc. The design of the *BA-TLS* framework has first been modeled as an optimization problem. Three different solution approaches to the formulated problem have been devised, thereafter. The first solution approach is based on *DP* and provides an optimal solution given discrete bandwidth unit. Utilizing analytical properties of the problem, the second approach constructs a *Genetic Algorithm (GA)* based stochastic strategy which provides good solution in almost all cases while consuming far lower temporal overheads compared to *DP*. The last approach is a fast and efficient deterministic heuristic known as *Proportionally Balanced Robustness-level Allocator (PBRA)*. *PBRA* is orders of magnitude faster than *GA* but delivers solution qualities which are comparable to it.

Next section presents the scheduling architecture of *BA-TLS*.

4.2 The BA-TLS Framework

According to a set of recent statistics by Conviva [82], viewer interruption from re-buffering affects ∼20.6% of video streams while ∼19.5% of users are impacted by slow video startups. In an endeavor to alleviate this problem, we have proposed an efficient radio resource allocation framework called *Buffer Aware Three Level Scheduler (BA-TLS)*. The *BA-TLS* framework provides a certain degree of robustness against re-buffering events and slow startups by attempting to quickly ramp-up and maintain playout buffer sizes of each flow above a threshold value. Such a protective shield against buffer outages enables the service provider to effectively deliver smooth viewing experience to all UEs under varying wireless channel conditions.

Typically, flows encountering poor channel qualities tend to suffer higher packet loss rates which in turn effects increased packet retransmissions. In such a

situation, a flow naturally undergoes through a higher risk of buffer outages. One of the possible ways in which a flow may avoid buffer outages (improve robustness) during such transient durations of low CQIs, is to maintain comparatively larger playout buffer sizes. With this insight, we have designed buffer threshold (BS^{th}) for each flow to be a dynamic quantity which is proportional to the instantaneous value of the CQI feedback for the flow. Buffer threshold for the i^{th} flow (BS_i^{th}) is calculated as:

$$BS_i^{th} = 2 \times (CQI_{max} - CQI_i + 1) \tag{4.1}$$

where, CQI_{max} (= 15) is the maximum possible CQI feedback in LTE and CQI_i is the current CQI feedback received by the i^{th} flow. As equation 4.1 shows, the value of BS_i^{th} for the i^{th} flow increases as its channel quality degrades. In this work, lower and upper bounds on BS_i^{th} has been set to be 2 secs (putting $CQI_i = CQI_{max} = 15$) and 30 secs (putting $CQI_i = 1$), respectively.

As discussed above, a targeted degree of robustness against varying channel conditions may only be guaranteed by maintaining the playout buffer size of each flow above the threshold value BS_i^{th}. Therefore, *BA-TLS* calculates the amount of data to be transmitted for each flow in the ensuing super-frame such that the playout buffer achieves its threshold size BS_i^{th}. However, achieving threshold buffer sizes may not always be possible for all flows due to limited instantaneous wireless bandwidth. In such a situation, the framework attempts to provide a fraction of the targeted threshold robustness to each flow such that the total data transmission demand of all flows remains less than the expected cell capacity. The value of the fractional robustness to be chosen for a given flow depends on its relative urgency towards buffer replenishment. For the flow f_i, this urgency is proportional to the difference ($BS_i^{th} - BS_i^{cur}$) between its targeted threshold buffer size (BS_i^{th}) and its current buffer size (BS_i^{cur}). However, it may be noted that, choosing an optimal fractional robustness value for each flow such that the overall system level urgency is maximally reduced while simultaneously avoiding resource capacity overloads, is a very hard problem. Therefore, in order

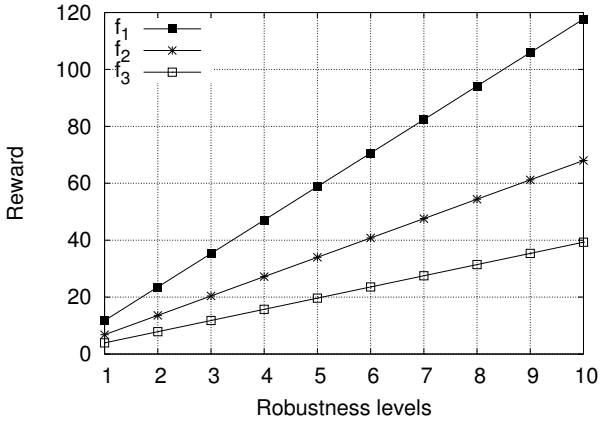


Figure 4.1: *Reward for the flows f_1, f_2 and f_3 at all the available robustness levels*

to control its complexity, the problem has been discretized as follows. The interval $(BS_i^{th} - BS_i^{cur})$ for each flow f_i has been partitioned into a constant number K of robustness levels (In this work, we have considered 10 levels, i.e. $K = 10$). The degree of urgency of a flow has been quantitatively represented in the form of an exponential reward function. The system obtains a reward R_{il} if it is able to successfully transmit the i^{th} flow at the l^{th} robustness level. R_{il} is given by:

$$R_{il} = exp\left(\frac{(BS_i^{th} - BS_i^{cur})}{1 + \sqrt{1/N \sum_i (BS_i^{th} - BS_i^{cur})}}\right) \times l \tag{4.2}$$

where, N is the total number of active flows. The reward obtained by the scheduler by delivering a certain amount of data to a client, depends on the buffer-state robustness the client achieves by receiving that data. The scheduler/system can obtain higher reward if it can maintain buffer sizes (for a flow) closer to the targeted threshold buffer size (BS^{th}). It may be noted from the above equation that reward R_{il} is a linearly increases function of the selected robustness level l. In the *BA-TLS* framework, the *System Overload Handler* of *TLS* is replaced by the *Robustness Level Selection module* which judiciously selects an appropriate robustness level for each flow.

We now present a simplistic example in order to unfold the concealed properties of the reward function. Assume a system consisting of three flows f_1, f_2

and f_3 with 10 robustness levels for each flow. Let the current playout buffer size (BS_i^{cur}) for all the flows be same and equal to 5 secs. Also, let the *CQI* feedbacks received by f_1, f_2 and f_3 be 9, 10 and 11, respectively. Hence, their corresponding buffer threshold values (refer equation 4.1) become 14 secs, 12 secs and 10 secs. The differences between their threshold and current playout buffer size ($BS_i^{th} - BS_i^{cur}$) are 9, 7 and 5, respectively. Figure 4.1 depicts plots for the reward value obtained by f_i, f_2 and f_3 as the robustness levels l varies from 1 to 10. It may be observed from the figure that for any given robustness value, higher the value of the difference $BS^{th} - BS_i^{cur}$, higher is the obtained reward. Thus, the designed reward function implicitly auto-tunes the system towards quick buffer ramp-up and thereby endeavours to mitigate possibilities of buffer outages.

4.2.1 Problem Formulation

Assume that the length of a super-frame interval is t TTIs and the number of RBs available in a TTI is p. Then, the total number of RBs available in a super-frame duration becomes $B = t \times p$. These RBs are required to be distributed among the N active flows ($Q = \{f_1, f_2, .., f_N\}$) at a given super-frame boundary. Let K be the total number of available robustness levels. Also, let b_{il} be the total number of RBs required to transmit the i^{th} flow at the l^{th} *robustness level*. x_{il} is a binary variable which is equal to 1 if flow f_i is selected for transmission at robustness level l. We then formulate the resource allocation problem as:

$$\text{maximize} \sum_{i=1}^{N} \sum_{l=1}^{K} R_{il} \times x_{il} \tag{4.3a}$$

subject to

$$\sum_{i=1}^{N} \sum_{l=1}^{K} b_{il} \times x_{il} \leq B, \tag{4.3b}$$

$$\sum_{l=1}^{K} x_{il} \leq 1, \; x_{il} \in \{0, 1\}, \forall i \tag{4.3c}$$

ALGORITHM 6: The Dynamic Programming based Robustness-level Allocator (DPRA)

Input: N : Number of active video flows,
B : Total number of resource block in a super-frame interval,
K: Total number of robustness levels available for each video flow,
b_{il} : RBs demand for the i^{th} video flow at the l^{th} robustness level,
R_{il} : Reward value for the i^{th} video flow at the l^{th} robustness level
Output: Selected robustness level for each video flow

```
1  begin
2      for β from 1 to B do
3          R(1, β) = 0;
4      for i from 2 to N do
5          for β from 1 to B do
6              R(i, β) = −∞
7      for i from 1 to N do
8          for β from 1 to B do
9              R(i, β) = U(i − 1, β);
10             for l from 1 to K do
11                 R(i, β) = max (R(i, β), R(i − 1, β − b_il) + r_il)
12     Output the solution that gives R(N, B);
```

The first constraint in Equation 4.3b guarantees that the total number of RBs allocated to all flows do not surpass the total available number of RBs (B) at eNodeB. The second constraint as given in Equation 4.3c forces each flow to select at most one robustness level.

4.2.2 Proposed Resource Allocation Schemes

The above formulation (refer equation 4.3) may be classified as a *Multiple Choice Knapsack Problem (MCKP)* [83], where each video flow is analogous to a class and the distinct robustness levels of each video flow are the items within each class. *Multiple Choice Knapsack Problem* can be optimally solved through a *DP* procedure. We first propose a standard dynamic programming procedure (described in Algorithm 6) to solve this *MCKP*.

4.2.2.1 The Dynamic Programming Robustness-level Allocator (DPRA)

This strategy selects a robustness level for each flow in such a manner that the aggregate reward over all flows is maximized. A step by step analysis of the working principle of the algorithm is as follows: Steps 1 to 12 select robustness levels for all flows. Let $R(i, \beta)$ be the optimal aggregate reward value for flows 1 through i given β, the total number of available RBs. The initialization of the optimal aggregate reward matrix ($R(i, \beta)$) is done in steps 2 to 6. The matrix $R(i, \beta)$ is built iteratively for all the flows, and for each flow the problem is solved for all the available RBs in a super-frame interval (B). The optimal aggregate reward value $R(i, \beta)$ depends on the robustness level selected for the i^{th} flow. Hence, for each robustness level (1 to K), the allocator checks the optimal aggregate reward value obtained by the first $i-1$ flows given $(\beta - b_{il})$ resource blocks, where b_{il} is the total additional number of *RBs* required to transmit the i^{th} flow at the l^{th} robustness level. To obtain optimal aggregate reward value $R(i, \beta)$, the allocator selects the robustness level for the i^{th} flow that gives the highest reward.

The computational complexity of the dynamic programming solution is $\mathcal{O}(N \times L \times B)$ where, N is the number of active users, L is an upper bound on the number of robustness levels corresponding to a flow and B is the total number of resource blocks in a super-frame. This overhead proves to be quite expensive as the number of RBs to be scheduled (B) is typically high even for moderately sized super-frames. For example, in a system with 20 MHz bandwidth (i.e. 100 RBs per TTI), the value of R is 100000 for a super-frame duration of just one second. Our experimental results show that given an *LTE* bandwidth of 20 MHz in a system with 100 active users and 2.5 GHz processing capacity, conventional *DP* takes $\sim$ 1.5 secs on average to generate a solution for a super-frame interval of size of 1 sec. The above overhead estimates indicate that to be practically useful, we require lower overhead heuristics for the online buffer aware RB allocation problem. Therefore, a *GA* based fast and efficient resource allocation heuristic has been proposed over the *BA-TLS* framework. For the example system mentioned

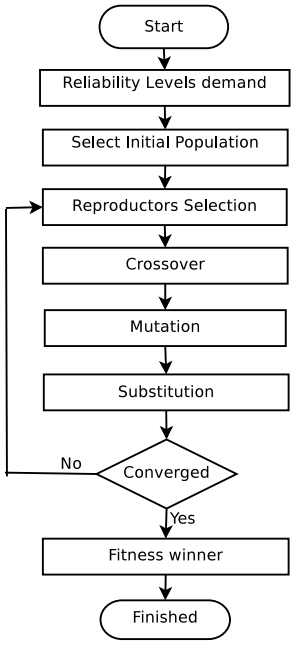


Figure 4.2: *Optimization strategy based on Genetic Algorithm*

above, our *GA* strategy is able to produce good and acceptable solutions in ∼0.02 secs which is ∼51 times faster than conventional *DP* (which takes ∼1.5 secs). A detailed description of the proposed strategy is presented in the next subsection.

4.2.3 Genetic Algorithm Strategy for Robustness Level Selection

Genetic algorithm [24, 84] is an optimization search procedure which samples the solution space of the problem iteratively and tries to locate a globally optimal solution after examining a limited number of candidates in the solution space. Effectiveness and efficiency of genetic algorithm comes from its potential to analyze the search space and maintain the traits of the best candidates already found while searching through the state space.

As depicted in Figure 4.2, the genetic algorithm selects a solution providing candidate robustness levels for each flow in five steps, namely, (1) Initialization, (2) Reproductors Selection, (3) Crossover, (4) Mutation Operator and (5) Sub-

stitution. The steps 2-5 are repeated until the solutions converge. Once, the solutions are converged, the algorithm determines a final solution which has the highest fitness value. Now, we discuss each step of our genetic algorithm strategy in detail.

Initialization: In this step, initial candidate solutions are generated. Here, a solution represents a distinct set of selected robustness levels for the flows. Initialization may be done through a *random* strategy or using the *first fit* scheme. However, with the *random* strategy, the candidate solutions are selected without investigating their feasibility. On the other hand, *best-fit* scheme only selects feasible candidate solutions. A solution is feasible if the total number of RBs required to transmit all flows do not surpass the total available RBs (B) at eNodeB. It may be noted that the *first fit* process is myopically greedy in choosing feasible candidate solutions and may induce poor overall performance of the genetic algorithm. On the other hand, random generation may provide good candidate solutions, but the chances for their infeasibility are also high. To ensure a good variety of selected robustness levels in the initial solutions, we have selected 30% of the candidate robustness levels in the initial configuration by the *first-fit* method while robustness levels for the rest of the candidates have been generated randomly.

Reproductors Selection: This step identifies the better solutions within a given generation using a metric called *Fitness Value (FV)*. These identified solutions act as reproductors for the creation of a new population. Feasible candidate solutions with relatively higher fitness values are closer to the optimal solution. The fitness value (FV_z) for the z^{th} candidate solution is defined as the sum of the rewards of all flows at their selected robustness levels i.e.

$$FV_z = \sum_{i=1}^{N} \sum_{l=1}^{L_i} R_{il} \times x_{il}^z \tag{4.4}$$

It may be noted that FV is same as the expression for the objective function of the ILP in equation 4.3.

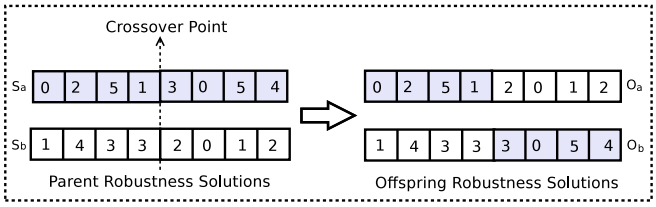


Figure 4.3: *Single-point Crossover procedure*

Crossover: One of the principal motivations towards the use of a genetic algorithm approach is to transfer the best traits in the parent solutions to the subsequent candidate generations. This is achieved through the mechanism of crossover where new child solutions are generated by combining parts of the configurations from two different parent solutions. For example, in the scenario shown in Figure 4.3, two offspring solutions O_a and O_b have been generated through a crossover operation on the parent solutions S_a and S_b. Here, both O_a and O_b are obtained by combining half of the robustness levels from S_a and remaining from S_b. The Primary goal of the crossover operation is to generate a good variety for a new population. This is typically referred to as "*Survival of the fittest*". However, good offspring candidates can only be generated by intelligently choosing parents having the best traits. This is achieved by assigning a selection probability to each parent solution based on its goodness/fitness. This work calculates selection probability SP_z of the z^{th} candidate using its *fitness value* as:

$$SP_z = \frac{FV_z}{\sum_{z=1}^{P} FV_z} \tag{4.5}$$

where, P denotes the size of the population.

It may be observed from the above equation that higher the fitness value of a candidate solution, higher will be its selection probability. Then, roulette wheel [24] selection scheme has been used to choose the parents based on their selection probability and traits of the parents are combined to generate new candidates. As shown in Fig 4.3, the algorithm calculates a single crossover point [85]

on both parents' and then, a new candidate solution is generated using robustness levels from the beginning of the chromosome to the crossover point from one parent, the rest being copied from the second parent. After that, a second child solution is generated from the remaining robustness levels of both the parent solutions.

Mutation Operator: It may be noted that just selecting the best solutions based on their fitness values is not enough for obtaining the global maximum reward because the approach may get stuck at a local maxima. The main purpose of the mutation operator is to maintain diversity by perturbing the solutions within a population and avoid premature convergence. The mutation operator randomly selects a flow (also called mutation flow) from the candidate solutions and assigns all possible robustness levels to it. Then, the algorithm calculates the fitness values corresponding to all newly assigned robustness levels to the mutation flow. The robustness level which corresponds to highest fitness value is assigned to the mutation flow.

Substitution: The improved solutions which are obtained by the crossover and mutation steps are included in the population, and the solutions which are infeasible are removed. Then, the algorithm repeats steps 2-5 until the solutions are converged.

GA thus produces good solutions at appreciably lower computational cost with respect to *DP*. However, insight obtained through a deeper look into the structure of the problem revealed that it is possible to design a poised proportionally balanced step by step heuristic solution approach which is capable of providing comparable performance with that of *GA* while incurring drastically lower computational overheads. This approach, which we call the *Proportionally Balanced Robustness-level Allocator (PBRA)* is discussed next.

4.2.4 Proportionally Balanced Robustness-level Allocator

The underlying philosophy of the *Proportionally Balanced Robustness-level Allocator (PBRA)* strategy is to select the robustness levels of a subset of the flows in each super-frame boundary, such that the robustness of the system is optimize. However, in order to obtain good and acceptable solutions, the heuristic strategy must be aware of the constraint on the available number of RBs. Hence, the algorithm must not only consider the individual reward gains during robustness level selections, but also the number of incremental RBs incurred during such a selection process. Therefore, we have transformed the original objective function, i.e *Reward (R)* into *Reward per RB (RpR). RpR* is calculated as:

$$RpR_{il} = \frac{R_{il}}{RB_{il}} \tag{4.6}$$

where, R_{il} is the reward value obtained by the system when it transmit the i^{th} flow at the l^{th} robustness level and RB_{il} is the total number of RBs required to transmit the i^{th} flow at the l^{th} robustness level. *RpR* proves to be a better performance metric than *Reward* as the former takes into account the spectral efficiencies of different flows and is able to provide higher overall Reward per super-frame interval.

At any given time during the *robustness level* allocation process of the *PBRA* heuristic, the flows are maintained in a priority queue organized as *max heap*. The priority of a flow within the *max-heap* is decided on the basis of a *key* which is defined for each flow as follows:

$$key_i = \mathbf{max}\{d[RpR^i_{lK}],\ d[RpR^i_{l(l+1)}]\} \tag{4.7}$$

Here, $d[RpR^i_{lK}]$ denotes the difference in *Reward per RB* values between its current (l) and highest (K) *robustness levels* and $d[RpR^i_{l(l+1)}]$ represents the *Reward per RB* difference of its current and immediately higher *robustness levels*.

It may be observed from equation 4.7 that key_i is able to provide an appropriate balance between the immediate gain obtained through a *robustness level*

ALGORITHM 7: The Proportionally Balanced Robustness-level (PBRA)

Input: N : Number of active video flows,
B : Total number of resource block in a super-frame interval,
K: Total number of robustness levels available for each video flow,
b_{il} : RBs demand for the i^{th} video flow at the l^{th} robustness level,
R_{il} : Reward value for the i^{th} video flow at the l^{th} robustness level

Output: Selected robustness level for each flow

1 Assume the current and highest *robustness levels* of flow f_i are l and K, respectively;
2 Initialize $l = 0$ for all flows;
3 Remaining Excess Capacity $REC = B$;
4 Compute the $d[RpR_{l(l+1)}]$ and $[RpR_{lK}]$ values for each active flow f_i;
5 Compute key_i: $\mathbf{max}\{d[RpR^i_{lK}],\ d[RpR^i_{l(l+1)}]\}$ for each flow f_i ;
6 Create a *max-heap* of the flows with the *key* values obtained;
7 **begin**
8 Extract the flow f_i from the root of the *max-heap*;
9 Let $RB^i_{l(l+1)}$ be the RBs required to increment the robustness level of the flows f_i by one;
10 Increment the *current robustness level* l of f_i by one if the incremental RBs ($RB^i_{l(l+1)}$) is at most equal to REC;
11 Update $REC = REC$ - $RB^i_{l(l+1)}$;
12 If ($REC \leq \epsilon$) Exit;
13 If ($l < K$), update key_i, reheapify f_i, go to step 9;

shift from l to $l + 1$ and the overall obtainable gain for the flow $d[RpR^i_{lK}]$. A situation (shown for robustness level upgradation here) where the consideration of such overall gains may be useful is as follows: Let us consider the relative prioritization of two flows f_i and f_j, currently allocated *robustness levels* x and y respectively (say), at an arbitrary intermediate stage of the *robustness level* allocation process using *PBRA*. Assume $d[RpR^i_{x(x+1)}]$ is lower than $d[RpR^j_{y(y+1)}]$. However, $d[RpR^i_{xK}] >> d[RpR^j_{yK}]$. In such a situation, if overall gain values are not considered as part of the *key*, f_j will be selected for the *robustness level* upgradation by one level over f_i even if its overall gain ($d[RpR^i_{xK}]$) is much greater than $\mathbf{max}\{d[RpR^i_{yK}],\ d[RpR^i_{y(y+1)}]\}$. A more severe case is that, if f_i's immediate gain $d[RpR^i_{x(x+1)}]$ is relatively very low, f_i may be indefinitely starved in spite of poten-

Table 4.1: *Simulation Parameters*

Simulation Time	120 secs
Bandwidth	20MHz
Number of RBs	100
Frame Structure	FDD
Cell Radius	1.0 km
Number of cells	19
Carrier Frequency	2GHz
Number of Video Flows	10 to 100
Video Traffic Generator	Trace Based
RLC ARQ	Maximum 5 retransmissions

tially handsome overall gains. Defining the *key* as $\mathbf{max}\{d[RpR^i_{lK}],\ d[RpR^i_{l(l+1)}]\}$ appropriately handles the situation.

The algorithm proceeds by repeatedly extracting the flow at the root of the *max heap*, incrementing its *robustness level* by 1, updating its *key* value and reheapifying it, until either the system capacity is completely exhausted, or all flows have been assigned their maximum possible *robustness levels*. The asymptotic worst-case time complexity of this heuristic is $\mathcal{O}\big(N\big)+\mathcal{O}\big(\sum_i L_i logN\big)$ where, N is the total number of active flows. The complexity $\mathcal{O}\big(N\big)$ is for the formation of the heap data structure for the N active flows. The $\mathcal{O}\big(\sum_i L_i logN\big)$ overhead is for the reheapify operation and updating the remaining excess/deficient capacity at each *robustness level* modification. A step wise description of the *Proportionally Balanced Robustness-level Allocator (PBRA)* algorithm is presented in Algorithm 7.

4.3 Experiments and Results

The performance of the *BA-TLS* framework has been experimentally evaluated using LTE-Sim [34], an open source simulator for *LTE* networks. All simulations run for 120 secs. A summary of the main simulation parameters are presented

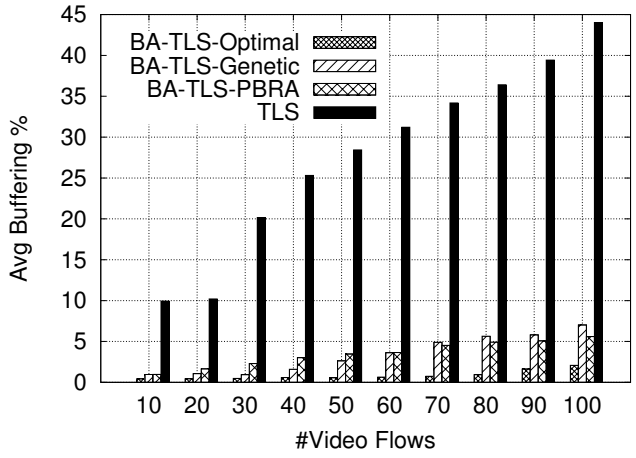
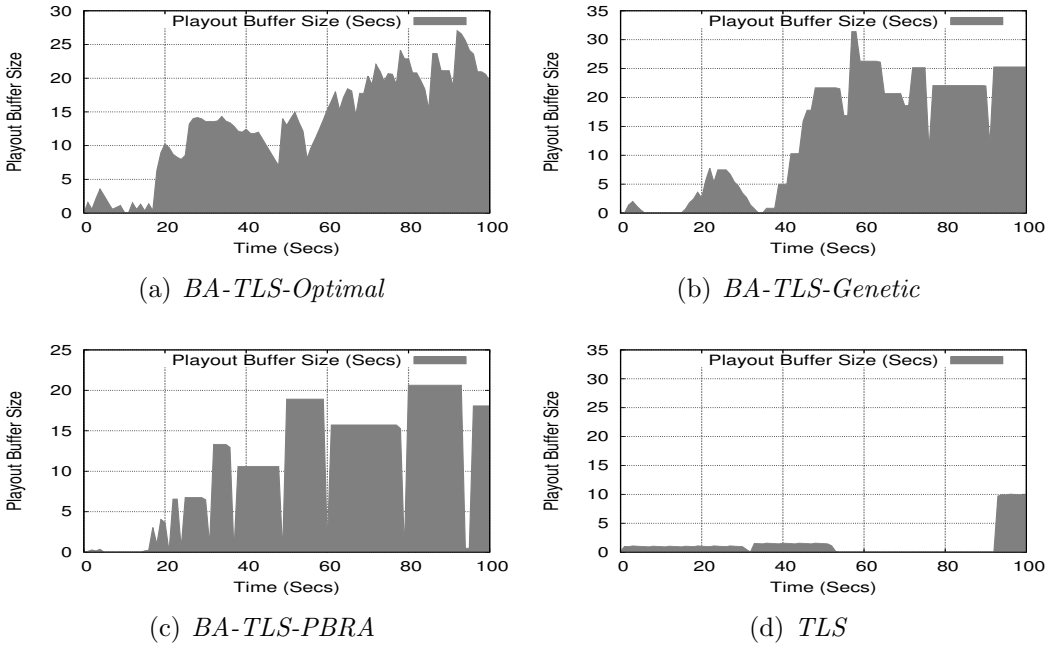


Figure 4.4: *Avg Buffering % Vs #Video Flows*

in table 4.1. A series of experiments have been conducted in order to measure the performance achieved by the proposed strategy. The obtained performance results have also been compared against the *Three Level Scheduling* (*TLS*) framework. The specific performance metrics which have been considered for evaluation are: (i) Average buffering (%), (ii) Instantaneous playout buffer status and (iii) Average execution time.

Figure 4.4 depicts plots for the average buffering suffered by the different resource allocation strategies against varying number of video flows (or system load) during the entire simulation length. It may be noted that average buffering (%) for all the strategies increases as the number of flows/system load increase. This is expected because the average number of RBs that may be allocated for a video flow reduces as the total number of video flows increases under a fixed resource block budget within a super-frame interval. Although, trends for all the methodologies in Figure 4.4 are similar, *BA-TLS-Optimal* is seen to encounter less buffering events compared to both the heuristic strategies, namely, *BA-TLS-Genetic* and *BA-TLS-PBRA*, while the basic *TLS* algorithm performs poorly in all cases. The reason for the poor performance of basic *TLS* originates from its ignorance of client side buffer status during the resource allocation process. On the other hand, the endeavour to maintain stable playout buffer sizes for each flow



(a) *BA-TLS-Optimal* (b) *BA-TLS-Genetic*

(c) *BA-TLS-PBRA* (d) *TLS*

Figure 4.5: *Instantaneous playout buffer size achieved by BA-TLS-Optimal, BA-TLS-Optimal and TLS strategies*

considerably reduces rebuffering events in the proposed buffer aware schemes. In Fig 4.4, *BA-TLS-Genetic* and *BA-TLS-PBRA* are seen to suffer slightly higher average buffering with respect to *BA-TLS-Optimal*, due to their inherent heuristic nature.

Figures 4.5(a) to 4.5(d) shows instantaneous buffer sizes achieved by *BA-TLS-Optimal*, *BA-TLS-Genetic*, *BA-TLS-PBRA* and TLS strategies respectively, for a single flow (namely, Star Wars) over the entire simulation duration. The scenario considers a cell with 100 active flows. It may be observed from the figures that the buffer-aware strategies, namely *BA-TLS-Optimal*, *BA-TLS-Genetic* and *BA-TLS-PBRA* are able to maintain approximately stable buffer sizes for the flow during the entire simulation duration. Stable playout buffer sizes in *BA-TLS* is achieved by two principle mechanisms: (i) Providing a certain degree of robustness to each flow against varying channel conditions and (ii) Auto tunning the priority of the flow during resource allocation based on its instantaneous

Table 4.2: *Comparative results for average run time (in millisecs)*

# Flows	*BA-TLS-Optimal*	*BA-TLS-Genetic*	*BA-TLS-PBRA*
10	144.2	4	0.016
20	297.7	6.7	0.024
30	447.3	9.3	0.025
40	600.8	12.4	0.029
50	744.8	15.3	0.031
60	902.1	18.6	0.035
70	1061	21.5	0.038
80	1210.2	24.3	0.04
90	1360.2	27.4	0.041
100	1523.4	29.6	0.049

playout buffer size and received *CQI* feedback (i.e. assigning relatively higher reward values to flows having comparatively lower playout buffer sizes and/or CQIs).

Table 4.2 shows the average execution time (in millisecs) taken by *BA-TLS-Optimal*, *BA-TLS-Genetic* and *BA-TLS-PBRA* as the number of video flows vary from 10 to 100. It may be observed from the table that the average execution time required for *BA-TLS-Optimal* is comparatively much higher than the *BA-TLS-Genetic* and *BA-TLS-PBRA* strategies. This happens because *BA-TLS-Optimal* calculates partial solutions for all possible bounds on number of flows ($\forall i \in [0, N]$), robustness levels ($\forall l \in [0, L_i]$) and RBs ($\forall \beta \in [0, B]$). On the other hand, the stochastic strategy *BA-TLS-Genetic* is observed to generate good and acceptable solutions much quicker as compared to the optimal strategy. This is expected because the *BA-TLS-Genetic* tries to locate a globally optimal solution after examining a limited number of candidates in the solution space. Therefore, the *BA-TLS-Genetic* strategy may be seen to achieve good speed-ups ($\sim$ 40 to 50 times) with respect to the *BA-TLS-Optimal* strategy. On the other hand, it may be observed that *BA-TLS-PBRA* perform far better than *BA-TLS-Optimal* and *BA-TLS-Genetic* in terms of computational overhead. This is because the com-

Table 4.3: *Comparative results for speed-ups achieved by the proposed heuristics*

# Flows	*Optimal Vs Genetic*	*Optimal Vs PBRA*	*Genetic Vs PBRA*
10	36	9013	250
20	44	12404	279
30	48	17892	372
40	48	20717	428
50	49	24026	494
60	49	25774	531
70	49	27921	566
80	50	30255	608
90	50	33176	668
100	51	31090	604

putational complexity of *BA-TLS-PBRA* only depends on the available number of flows and the number of available robustness level. Therefore, *BA-TLS-PBRA* is able to achieve drastic speed-ups with respect to the other two strategies. Comparative results for achieved speed-ups are shown in Table 4.3.

4.4 Summary

In this chapter, a new *Buffer Aware Three Level Scheduling* framework (called *BA-TLS*) has been proposed. *BA-TLS* is based on *TLS* and hence, inherits all its salient facets. The framework is aimed at enabling the service provider to deliver smooth viewing experience to the end users even during transient overloads through a buffer aware adaptive scheduling strategy that attempts to minimize client-side re-buffering events. The resource allocation model has been formulated as an optimization problem for which conventional dynamic programming solution is shown to impose substantial overheads. Thus, two fast and efficient solution strategies, namely *BA-TLS-Genetic* and *BA-TLS-PBRA* have been designed with the endeavor to achieve efficient but low-overhead resource adaptability. Experimental results show that the genetic algorithm *BA-TLS-Genetic*

produces good solutions and at the same time is about *40* to *50* times faster than the optimal strategy *BA-TLS-Optimal*. The deterministic heuristic strategy *BA-TLS-PBRA* in-turn is comparable in performance while being about *300* to *600* times faster on average compared to *BA-TLS-Genetic*.

It may be noted form the experiments section that although *BA-TLS* is able to significantly restrict re-buffering events, buffering increases as the number of flows/system load increase. Therefore, only playout buffer awareness may not be enough to restrict re-buffering events, especially during system overloads. The next chapter presents a video bit-rate adaptation framework whose objective is to deliver smooth viewing experience to all end users even during overloaded conditions by appropriately adjusting the qualities of adaptive video flows.

Chapter 5

A Resource Allocation Framework for Adaptive Video Streaming Over LTE

Video streaming over wireless is expected to be one of the main revenue generators for the current and future mobile broadband networks. However, meeting the ultimate objective of delivering all clients/User Equipments (UEs) with high *QoE* pivots around several key factors including: (i) Transmitting all video streams with satisfactory quality over a limited and temporally varying wireless bandwidth, (ii) Minimizing buffer outage at the *UE* in order to guarantee seamless video viewing experience and (iii) Stabilizing bit-rate switches to avoid user annoyance due to flickering. In this effort, *DASH* has emerged as a key technology that enables enhancement of the overall transmission quality of a service provider by allowing online video bit-rate adaptation over time. In *DASH*, the content delivery server maintains the video as segments of fixed duration with each segment being stored at multiple quality levels to permit dynamic quality adaptations during transmission. However, power of the *DASH* streaming technology can only be harnessed to its fullest by using an efficient online radio resource scheduler which appropriately adjusts the quality levels and data transmission rates for all video flows from their respective *DASH* servers. In this chapter, we have presented a new video adaptation framework whose objective is to maximize the aggregate *QoE* over all flows by simultaneously considering the above mentioned

QoE parameters in a judiciously balanced manner.

However, selecting appropriate data-rates and efficiently multiplexing the available bandwidth among clients in both frequency and time in order to maximize *QoE* is a complex online scheduling problem. We first pose this problem as an *ILP* formulation and solve through a conventional *DP* strategy. However, conventional *DP* proves to be prohibitively expensive in terms of online computational overheads. Our experimental results show that given an *LTE* bandwidth of 20 MHz in a system with 50 active users, conventional *DP* takes 14.4 secs on average to generate a solution even for a moderate adaptation interval size of 1 sec on a 2.5 GHz computing core. Therefore, in order to design *AVRC* as an effective online mechanism which can be implemented with moderate computational resources and applied at each adaptation interval boundary, we have modified conventional *DP* and devised a new strategy known as "*Streamlined DP-based Quality-level Allocator (SDQA)*". *SDQA* intelligently leverages the discrete nature in the data-rate scalability of video flows to retain a far lower number of non-dominating partial *DP*-solutions and allows the ultimate optimal solution to be generated much quicker. Further, we propose a tunable approximation scheme called *SDQA-AA* that may be employed to accelerate the speed of generating solutions (or limit necessary computational resources) by various optional degrees with distinct bounds on the degradation in solution quality.

Before providing the detail description of the proposed framework, we have presented the system overview which is used by the proposed architecture.

5.1 System Overview

The overall system considered in this work is based on *QoE*-aware *DASH* employing *LTE* as the cellular network protocol. Figure 5.1 represents an abstract system overview. The *MPEG-DASH* based encoded video to be transmitted to a given client is stored in the *DASH* server as shown in the figure. In order to achieve dynamic bit-rate adaptation, *DASH* fragments the entire video into

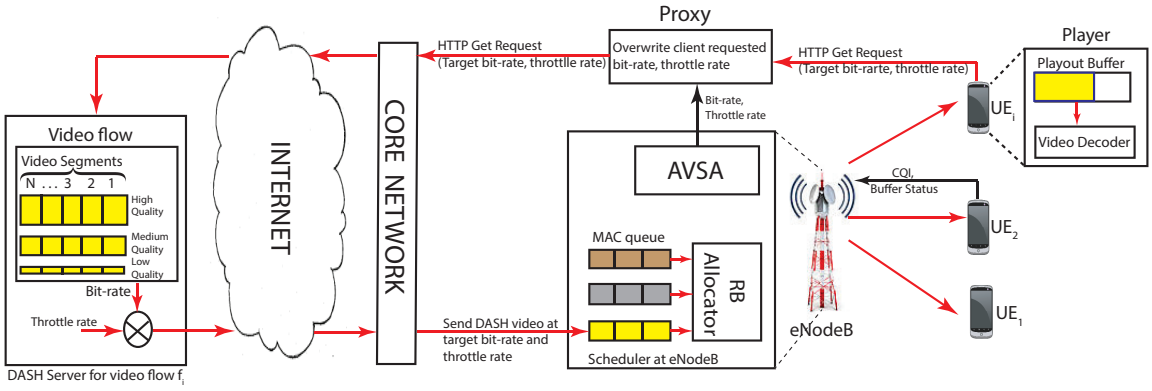


Figure 5.1: *System overview for DASH video delivery over LTE networks.*

multiple short duration *segments* of one to a few seconds, where each segment is encoded at various distinct bit-rates/quality levels. Each Segment of the desired video is downloaded by the client in a sequential manner using standard HTTP GET requests from the *DASH* server. Downloaded segments are buffered and subsequently decoded/played-back by the client/UE. While most of the framework is standardized as part of the *MPEG* based standard *DASH*, the adaptation algorithm to select the most appropriate bit-rate and throttle rate for future segments is left to the specific implementation. Here, throttle rate refers to the boost provided to the transmission rates of segments from the *DASH* server (with respect to their client-side decoding/playback rates). This technique is used to maintain adequate playout buffer sizes at the UE and thereby avoids buffer outages.

This work considers a scenario in which each client/UE_i receives a single *DASH* video f_i. As depicted in the figure, video segment packets (of the video flow say, f_i) from a *DASH* content delivery server are transmitted via the Internet and core network to its designated *MAC* queue at the serving eNodeB corresponding to UE_i. Similarly, the eNodeB receives video flows corresponding to other clients. Subsequently, the RB allocator within the scheduler at eNodeB assigns an appropriate number of RBs for each flow f_i in order to transmit video packets from its designated *MAC* queue to the client UE_i, over the air interface. In order to deliver high system-level *QoE* while ensuring that overloads are avoided, the

scheduler must judiciously control and balance the bit-rates and throttle rates for the future video segments of all active flows at eNodeB. This bit-rate and throttle rate adaptation for each flow is performed periodically by the *Adaptive Video Streaming Architecture (AVSA)* module within the scheduler and the generated output is fed back through a proxy to the *DASH* server for the flow. *Therefore, in contrast to classical DASH adaptation solutions where the client selects the quality level and throttle rate for each segment, this work has adopted a more proactive approach where a proxy overwrites the clients' HTTP requests* [49]. In order to achieve the objective of providing satisfactory *QoE* to all clients, *AVSA* estimates the overall instantaneous capacity of the system from channel condition information collected from all clients. In *LTE*, this information is generated by each *UE*/client by estimating the *SINR* corresponding to the currently received reference signals over the sub-channels. The *SINR* values are then mapped it to a set of *CQI* feedbacks (integers in the range 1 to 15) which are periodically forwarded to *eNodeB*. In addition, *AVSA* also collects the instantaneous buffer status information from all clients so that the selected bit-rates and throttle rates will be able to ensure uninterrupted playback.

5.2 Adaptive Video Streaming Architecture

The *Adaptive Video Streaming Architecture (AVSA)* is a *DASH* based flexible resource allocation framework which attempts to effectively maximize the overall *QoE* of the end-users by appropriately selecting the bit-rates/quality levels (l_i) and throttle rates (TR_i) of future video segments under varying network conditions. As shown in figure 5.2, the $< l_i, TR_i >$ values selected by AVSA are passed to the proxy which then feeds back these decisions to respective *DASH* servers as depicted in Figure 5.1. It may be observed from Figure 5.2 that the proposed architecture operates using three controllers: (i) *TRC*, (ii) *SSC* and (iii) *AVRC*. All the controllers cooperatively work in unison at each adaptation interval boundary to ensure high *QoE*. Typically, the duration of an adaptation interval ($|AI|$) is

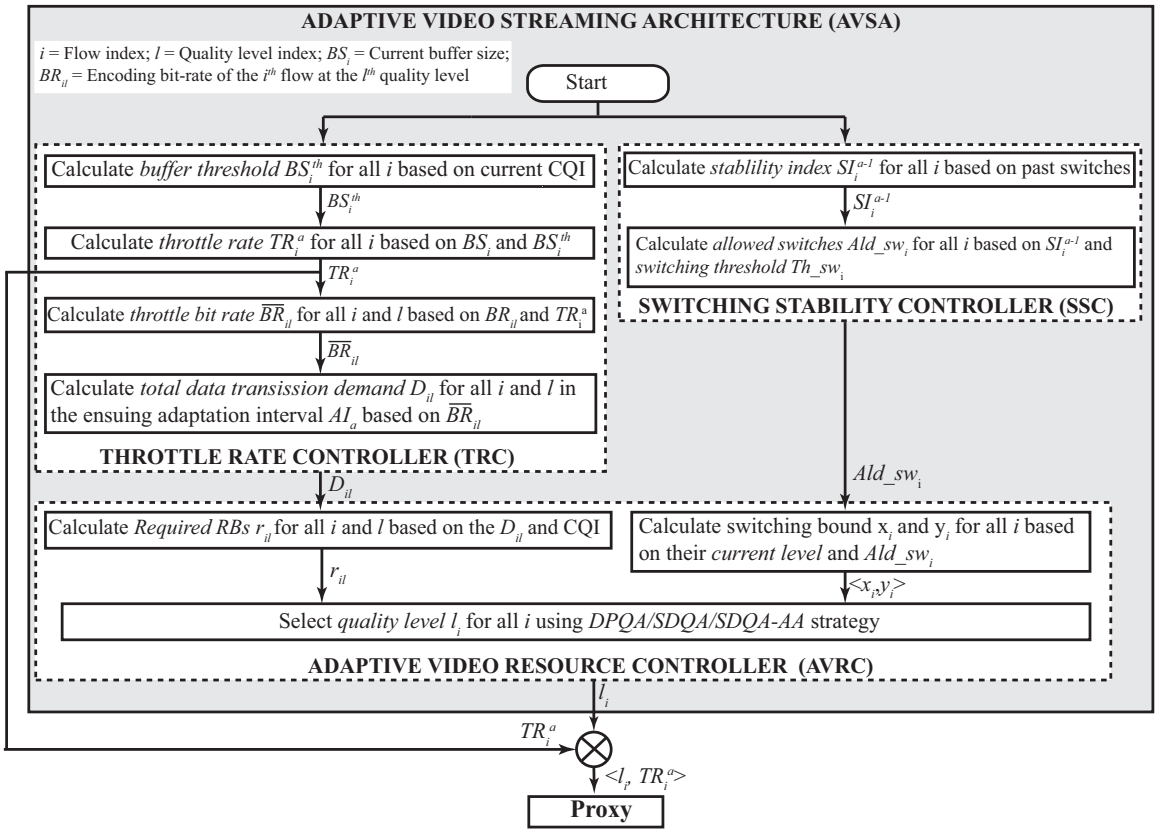


Figure 5.2: *Adaptive Video Streaming Architecture (AVSA).*

order of a few seconds. A few important terminologies which will be used in the current and later sections of the paper are presented in Table 5.1.

5.2.1 Throttle Rate Controller (TRC)

Stutter-free video playout is an important parameter which contributes to the overall *Quality of Experience* of a flow. Stuttering or playout interruptions is mainly caused by *outages of the playout buffer at the UE*. Throttling [86] is a well known technique of maintaining adequate playout buffer sizes at UEs by boosting transmission rates of flows (with respect to their encoding bit-rates) from the server. For example, transmission bit-rates for Dailymotion and Vimeo are statically throttled typically at 1.25 times the encoding rate in Flash and native players of Android devices [86]. However, such static throttling schemes being

Table 5.1: *Notations ($\mathcal{N}$) and their definitions*

$\mathcal{N}$	Explanation
i	Flow/Client/UE index
l	Video quality level/bit-rate index
α	Upper bound on #RBs per adaptation interval
N	# video flows; also denotes, # clients/UEs
R	# RBs in an adaptation interval
L_i	# quality levels available for the i^{th} flow
r_{il}	i^{th} flow's RB demand at l^{th} quality level
q_{il}	i^{th} flow's video quality at l^{th} quality level
x_i	Min quality level allowed for the i^{th} flow
y_i	Max quality level allowed for the i^{th} flow
ρS_i	List of partial solutions using at most i flows
ρS_{il}	List of partial solutions using upto i flows - with the i^{th} flow fixed at l^{th} quality level
$\rho SN_{ip} \rightarrow \alpha$	The bound on the number of RBs for the p^{th} node in ρS_i
$\rho SN_{ip} \rightarrow q$	The quality value corresponding to the p^{th} node in ρS_i
$\rho SN_{ip} \rightarrow QL$	List of selected quality levels for the flows $f_1, f_2, \ldots, f_i$ corresponding to the p^{th} node of solution ρS_i

oblivious of *UE*-buffer sizes tend to be conservative and often induces unnecessarily high traffic loads in the system which may create artificial overload situations at eNodeB and buffer overflows at *UEs*. Hence in this work, we propose an efficient *UE*-buffer aware *TRC* that dynamically adjusts the transmission rates of the flows at the boundary of each adaptation interval. The objective of this adjustment process is to replenish the playout buffer of each *UE* so that the system is driven towards buffer state stability for all flows, thus minimizing the probability of buffer outages in the process. Figure 5.2 pictorially depicts the major steps within the *TRC* module of *AVSA*.

Let BS_i^{th} (refer section 4.2, equation 4.1, page no. 93) denote the safe threshold buffer size for flow f_i above which the probability of buffer outages is negligible.

Based on the targeted playout buffer threshold (BS_i^{th}), *TRC* calculates a throttling rate for each flow (refer Figure 5.2). The throttling rate (TR_i^a) for the flow f_i at the a^{th} adaptation interval boundary is calculated as follows:

$$TR_i^a = \begin{cases} \max\left(\left(1 + \frac{BS_i^{th} - BS_i}{BS_i^{th}}\right), 0.5\right), & \text{if } BS_i < BS_i^{max} \\ 0, & \text{Otherwise} \end{cases} \tag{5.1}$$

where, BS_i and BS_i^{max} denote the instantaneous and maximum buffer size of f_i. The above equation shows that throttling rate of a flow may vary from 0.5 to 2 depending on its instantaneous buffer size (when the buffer is not yet fully filled). Based on the calculated throttling rate, *TRC* calculates the throttled bit-rate $\overline{BR_{il}}$ and total data transmission demands D_{il} for each flow f_i at all its distinct quality levels ($l = 1,\ 2, \ldots\ L_i$; where, L_i denotes the number of quality levels available for f_i) at each adaptation interval boundary. $\overline{BR_{il}}$ and D_{il} are calculated as:

$$\overline{BR_{il}} = BR_{il} \times TR_i^a \tag{5.2a}$$

$$D_{il} = \overline{BR_{il}} \times |AI_a| \tag{5.2b}$$

where, BR_{il} is the estimated encoding bit-rate of the i^{th} flow at the l^{th} quality level and $|AI_a|$ is the duration of the a^{th} adaptation interval. As Figure 5.2 depicts, D_{il} is fed to the *AVRC*.

5.2.2 Switching Stability Controller (SSC)

Another important vertical which contributes significantly to the overall *QoE* of a flow is known as *stability* (defined as the number of quality level switches per unit time). Frequent inter-level switching creates annoyance among users as the video image is perceived to continually flicker in quick successions, thus ultimately pulling down the overall *QoE*. In this work, we have defined a metric called *Stability Index (SI)* which computes the exponential moving average over the number of switches encountered in the past. If the quality level of a flow f_i

gets updated from l_i^{a-2} to l_i^{a-1} in the $(a-1)^{th}$ adaptation interval, the stability index SI_{il}^{a-1} for the i^{th} flow is defined as follows:

$$SI_{il}^{a-1} = \beta|l_i^{a-1} - l_i^{a-2}| + (1-\beta)SI_i^{(a-2)} \tag{5.3}$$

It may be observed that this metric assigns the highest weightage to the switching encountered in the a^{th} adaptation interval, with successively previous adaptation intervals obtaining exponentially lower weightages. Here, β is a positive choice variable (between 0 to 1) whose value governs the smoothing characteristic and $SI_i^{(a-1)}$ is the stability index of the i^{th} flow in the $(a-1)^{th}$ adaptation interval. A higher value of β discounts past switches faster.

As may be observed from Figure 5.2, *SSC* consists of two principal steps. The first step determines the stability index SI_i^{a-1} for all flows f_i after completion of the last adaptation interval $a-1$. Given SI_i^{a-1}, *SSC* calculates the allowable number of quality level switches (Ald_sw_i) for all flows in the ensuing adaptation interval a, such that the exponential average of Ald_sw_i and SI_i^{a-1} is at most a desired switching threshold Th_sw_i. That is,

$$Th\_sw_i \geq \omega \times Ald\_sw_i + (1-\omega) \times SI_i^{a-1} \tag{5.4}$$

Ald_sw_i may be derived from the above equation as,

$$Ald\_sw_i = \mathbf{max}\left\{\frac{\left(Th\_sw_i\right) - \left((1-\omega) \times SI_i^{a-1}\right)}{\omega}, 0\right\} \tag{5.5}$$

The value of Ald_sw_i calculated by *SSC* is fed to *AVRC*, as shown in Figure 5.2.

5.2.3 Adaptive Video Resource Controller (AVRC)

AVRC selects an appropriate quality level for each flow so that the overall video transmission quality is maximized in the ensuing adaptation interval. In this work, the quality (qil) of a video flow at level l is characterized in terms of a metric called *Peak Signal to Noise Ratio (PSNR)* which uses a distortion free version of the video as reference. For video streaming applications, *PSNR* is an

objective quantitative parameter that represents a high degree of correlation with user-perceived video quality [87].

It may be noted that such a quality maximization process must be carried out while ensuring that the total bandwidth demand over all the flows remains within the expected cell capacity. In order to achieve this, the controller first calculates the number of RBs required (r_{il}) to transmit each flow at each quality level based on its total transmission data demand D_{il} (received from *TRC*) and *CQI* feedbacks from the corresponding client (as shown in Figure 5.2). *AVRC* then calculates the lowest (x_i) and highest (y_i) quality levels between which f_i's quality should be selected so that the upper bound on allowable switching Ald_sw_i (received from *SSC*) is respected. x_i and y_i is calculated as:

$$x_i = \mathbf{max}\ \{(l_i^{a-1} - Ald\_sw_i), 1\}, \tag{5.6a}$$

$$y_i = \mathbf{min}\ \{(l_i^{a-1} + Ald\_sw_i), L_i\} \tag{5.6b}$$

Where, l_i^{a-1} and L_i denote the current quality level and the total number of available quality levels for the i^{th} flow. Finally, based on calculated RB demands (r_{il}) and switching bounds (x_i and y_i), *AVRC* selects the appropriate quality level (l_i^a) at which each flow f_i should be transmitted in the next adaptation interval a through a mechanism whose overall objective is to maximize aggregate video quality (refer Figure 5.2). In the next subsection, we pose this problem as an *ILP* formulation. Subsequently, in subsections 5.2.3.2, 5.2.3.3 and 5.2.3.4, three dynamic programming (DP) based solution strategies to the problem has been discussed.

5.2.3.1 Quality Level Selection

Let us assume that the length of an adaptation interval is t *TTIs* and the number of RBs available in a *TTI* is p. Then, the total number of *RBs* available in an adaptation interval becomes $R = t \times p$. These *RBs* are required to be distributed among the N active flows ($F = \{f_1, f_2, .., f_N\}$) at a given adaptation interval boundary. Let x_i and y_i be the minimum and maximum quality level threshold

bounds for the i^{th} flow. Let q_{il} be the video quality delivered and r_{il} be the total number of *RBs* required to transmit the data demand ($D_{il}(t)$; refer equation 5.2b) for the i^{th} flow at the l^{th} quality level. Also, let z_{il} be a binary variable which is equal to 1 if flow f_i is selected for transmission at quality level l. We then formulate the resource allocation problem as:

$$\text{maximize:} \quad \sum_{i=1}^{N} \sum_{l=x_i}^{y_i} q_{il} \times z_{il} \tag{5.7a}$$

$$\text{subject to:} \quad \sum_{i=1}^{N} \sum_{l=x_i}^{y_i} r_{il} z_{il} \leq R, \tag{5.7b}$$

$$\sum_{l=x_i}^{y_i} z_{il} \leq 1, \; z_{il} \in \{0,1\}, \forall i \tag{5.7c}$$

The first constraint in Equation 5.7b guarantees that the total number of RBs allocated to all flows do not surpass the total available number of *RBs* (R) at eNodeB. The second constraint as given in Equation 5.7c forces each flow to select at most one quality level. The objective of equation 5.7 is to maximize the aggregate *quality value* considering all video flows, for a given limited number of *RBs* (R) available and bounds on switching (x_i, y_i). The above definition is guaranteed to achieve targeted buffer sizes (as decided by *TRC*) for all flows as both D_{il} and r_{il} are obtained based on throttled demands.

5.2.3.2 DP Based Quality-level Allocator (DPQA)

In the above subsection, we formulated the bit-rate/quality level selection strategy as an *ILP* problem. This quality level selection has to be conducted on-line at every adaptation interval. An effective scheduling strategy should provide good solutions quickly. However, it is well known that although an *ILP* provides optimal solutions, it is inherently exponential in nature and is poor in terms of scalability.

A closer look into the scheduling problem reveals that use of the *LTE* framework (resource allocation in terms of RBs) makes it naturally discrete in nature.

Additionally, the problem has an optimal substructure. Hence, the optimal solution may be obtained as a composition of the optimal solutions to a set of its subproblems and therefore, *DP* provides a natural solution mechanism. This makes it possible to obtain optimal solutions in times which are pseudo-polynomial in the number of RBs. We first propose a conventional *DP* procedure to solve this problem. We have referred to this conventional *DP* based solution approach as *DP-based Quality-level Allocator (DPQA)*. A step-by-step algorithm along with discussion of *DP* is given in section 4.2.2, Algorithm 6, page no. 96 in previous chapter. *DP* is essentially an optimization procedure and the problem definition in equation 5.7 can be represented by the following *DP* recursive formulation:

$$Q(i,\alpha) = \begin{cases} 0, & \text{if } i = 0 \text{ or } \alpha = 0 \\ max_l\{Q(i-1,\alpha), q_{il} + Q(i-1,\alpha - r_{il})\}, & \forall l \in \{x_i, y_i\} | \alpha \geq r_{il} \end{cases} \tag{5.8}$$

The computational complexity of *DP Based Quality-level Allocator (DPQA)* is $\mathcal{O}(N \times L \times R)$ where, N is the number of active users, L is an upper bound on the number of quality levels corresponding to a flow and R is the total number of RBs in a adaptation interval. This overhead proves to be quite expensive as the number of RBs to be scheduled (R) is typically high even for moderately sized adaptation intervals. For example, in a system with 20 MHz bandwidth (i.e. 100 *RBs* per ms (*TTI*)), the value of R is 100000 for an adaptation interval duration of just one second. Our experimental results show that, given *50* active users in such a system, *DPQA* takes $\sim$ 14.4 secs on average to generate a solution on a 2.5 GHz computing core. It is easy to understand that this overhead is significantly high.

However, as discussed before, the optimization problem under consideration is inherently discrete in nature. Thus, we do not obtain continuous improvements in video quality of a flow with each additional RB allocated to it. Rather, the quality improvement has a step-wise nature with $(y_i - x_i)$ steps corresponding to each flow f_i. Now typically, $y_i - x_i << R$ and consequently, a majority of the optimal solutions $Q(i,\alpha)$ do not return distinct values. As a result, generic *DP*

which memoizes all partial solutions for each distinct value of i and α (in equation 5.8), may suffer from high and unnecessary computational overheads. This overhead may be reduced (often drastically) through a more efficient implementation which memoizes only those partial optimal solutions which provide distinct enhancements in quality with increment in the bound on the number of RBs α, for each value of i. Further, it may be observed that this reduced memoization do not cause solution optimality to be compromised. Here, we propose an efficient and optimal solution strategy called *SDQA*.

5.2.3.3 Streamlined DP-based Quality-level Allocator (SDQA)

Before providing a detailed description of *SDQA*, we first introduce the principal data structure used in SDQA.

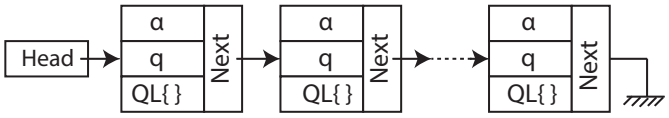


Figure 5.3: *General structure of lists ρS_i or any list in PS*

Data Structure: The principle data structure used by the *SDQA* algorithm is a set of linked lists each of which contain partial optimal solutions. Similar to i, the algorithm iterates over the N flows producing the final optimal solution after all flows have been considered. During any given iteration, say i, *SDQA* maintains: (i) A linked list ρS_{i-1}, which contains the list of distinct partial solutions corresponding to sub-problems considering at most $i-1$ flows $(f_1, f_2, \ldots, f_{i-1})$, and (ii) A set of linked lists $PS = \{PS_{il} \mid x_i \leq l \leq y_i\}$. Any linked list PS_{il} in PS contains partial solutions considering upto i flows, but restricting the i^{th} flow to be transmitted only at its l^{th} quality level. The p^{th} node of ρS_i and PS_{il} are denoted as ρSN_{ip} and PSN_{ilp}, respectively. Each node (ρSN_{ip} or PSN_{ilp}) of any given linked list (the fields in each node of ρS_{i-1} as well as PS_{il} are same) holds a solution corresponding to flows $f_1, f_2, \ldots, f_i$. Only those solutions for which the total number of *RBs* required (α) is at most the maximum number

ALGORITHM 8: Streamlined DP based Quality-level Allocator (SDQA)

Input: $q_{il},\ r_{il},\ R$
Output: Selected quality level for each video flow

1 Initialize the data structures with the partial solutions considering no flows and zero RBs. The corresponding solution list ρS_0 will have a single node ρSN_{00} with the following field values: (i) $\rho SN_{00} \rightarrow \alpha = 0$, (ii) $\rho SN_{00} \rightarrow q = 0$ and (iii) $\rho SN_{00} \rightarrow QL = \{\ \}$;
2 **for** i *from 1 to* N **do**
  /* Iterate over all flows; Let G be the number of nodes in ρS_{i-1} */
3 **for** l *from* x_i *to* y_i **do**
    /* Iterate over all quality levels of each flow */
4   **for** p *from* 1 *to* G **do**
      /* Iterate over nodes of ρS_{i-1} */
5      $PSN_{ilp} \rightarrow \alpha = \rho SN_{(i-1)p} \rightarrow \alpha + r_{il}$;
6   **if** $PSN_{ilp} \rightarrow \alpha > R$ **then**
7   Break (Go to step 3);
8      $PSN_{ilp} \rightarrow q = \rho S_{(i-1)p} \rightarrow q + q_{il}$;
9      $PSN_{ilp} \rightarrow QL = \rho S_{(i-1)p} \rightarrow QL \bigcup \{l\}$;
10   Enqueue the node PSN_{ilp} into PS_{il} ;
11 $\rho S_i =$ **Merge** $\left(PS_{ix_i},\ PS_{i(x_i+1)},\ \ldots,\ PS_{iy_i}\right)$;

of available *RBs* (R), are considered feasible and retained. The rest of the solutions are discarded and not accommodated in the linked-lists. The fields in any given node of a linked list (ρSN_{ip}, say) are: (i) The quality value of the solution, $\rho SN_{ip} \rightarrow q$, (ii) The bound on the number of RBs, $\rho SN_{ip} \rightarrow \alpha$, and (iii) An enumeration/list of selected quality levels $\rho SN_{ip} \rightarrow QL(= \{l_i, l_2, \ldots, l_i\}; l_k$ denotes the quality level selected for flow f_k) for the flows $f_1, f_2, \ldots, f_i$ corresponding to the solution. It may be noted that the nodes in any of the linked lists follow a strict ordering based on the values of its attributes $'q'$ and $'\alpha'$. For the linked list ρS_i, say, any two consecutive nodes always satisfy the following two conditions: (i) $\rho SN_{ip} \rightarrow \alpha < \rho SN_{i(p+1)} \rightarrow \alpha$, (ii) $\rho SN_{ip} \rightarrow q < \rho SN_{i(p+1)} \rightarrow q$. That is, all linked lists store only the non-dominating solutions and discard the rest.

Detailed Algorithm: The *SDQA* strategy is an iterative algorithm which memoizes only those partial optimal solutions which provide distinct enhancements in quality. A step wise description of the *SDQA* is presented in Algorithm 8. In step 1, the algorithm initializes the partial solution with a single node (ρSN_{00}) considering no flows and zero *RBs*. Then, from steps 2 to 11, the *SDQA* strategy iteratively builds the final optimal solution. In the i^{th} iteration (step 2), the algorithm includes one extra flow f_i in the sub-problem and first builds a set of partial optimal solutions $PS = \{PS_{il} \ \forall l \in [x_i, y_i]\}$ from ρS_{i-1} (refer steps 2 to 10). The for loop in steps 3 to 10 generates the partial optimal solution PS_{il} by including the i^{th} flow at the l^{th} quality level to the partial solution $\rho S_{(i-1)}$. The p^{th} iteration first populates the field $PSN_{ilp} \rightarrow \alpha$ with the sum of r_{il} (the number of RBs required to transmit the i^{th} flow at the l^{th} quality level) and $\rho SN_{(i-1)p} \rightarrow \alpha$ (step 5). If this total RB demand PSN_{ilp} results in an overload ($PSN_{ilp} \rightarrow \alpha > R$), such overheads will also happen for subsequent values of p, as ρS_{i-1} is sorted in strictly increasing order of α and q. Hence, in this situation, the control breaks out from the loop and goes back to step 3 and no further nodes are enqueued into the solution list PS_{il}. Otherwise ($PSN_{ilp} \rightarrow \alpha \leq R$), the algorithm calculates the other two attribute values of node PSN_{ilp}, namely, $PSN_{ilp} \rightarrow q$ and $PSN_{ilp} \rightarrow QL$ in steps 8 and 9, respectively. PSN_{ilp} is then enqueued at the current tail of PS_{il} in step 10. Finally in step 11, the partial optimal solution ρS_i is constructed by using a **Merge()** routine as described in Algorithm 9. All the calculated PS_{il} solutions are given as input to the **Merge()** function.

The **merge()** procedure merges the linked lists $PS_{ix_i}, \ldots, PS_{iy_i}$ to generate the partial optimal solution ρS_i that considers upto i flows $(f_1, f_2, \ldots, f_i)$. As discussed above, the generated solution is non-dominating with respect to both resource α and delivered quality q. The while loop in steps 5 to 13 iterates until PS becomes empty. In step 6, the algorithm determines the subset of partial solutions (with indexes $\{a_1, \ldots, a_\lambda\}$) which consumes the minimum amount of resource among all solutions over all linked lists in PS. It may be

ALGORITHM 9: Merge $\left(PS_{ix_i},\ PS_{i(x_i+1)}, \ldots, PS_{iy_i}\right)$

Output: ρS_i

1 Let $PS = \{PS_{ix_i},\ PS_{i(x_i+1)}, \ldots, PS_{iy_i}\}$;

2 Let k_{il} and σ_i be pointers to the current heads of the lists PS_{il} and ρS_i;

3 **for** $l = x_i \ldots y_i$ **do** $k_{il} = PS_{il}.head \rightarrow Next$;

4 $\sigma_i = \rho S_i.head$; $\rho S_i.head \rightarrow q = 0$; $\rho S_i.head \rightarrow \alpha = 0$;

5 **while** $PS \neq \varnothing$ **do**

6 Find $\{a_1, a_2, \ldots, a_\lambda\}$, such that,
$\left\{\{a_1, \ldots, a_\lambda\} \subseteq \{x_i, \ldots, y_i\},\ k_{ia_1} \rightarrow \alpha == k_{ia_2} \rightarrow \alpha == \ldots == k_{ia_\lambda} \rightarrow \alpha == min_{l=x_i}^{y_i}\{k_{il} \rightarrow \alpha\}\right\}$;

7 Find m, such that,
$\{m \in \{a_1, a_2, \ldots, a_\lambda\},\ k_{im} \rightarrow q == max_{l=a_1}^{a_\lambda}\{k_{il} \rightarrow q\}\}$;

8 **if** $\sigma_i \rightarrow q < k_{im} \rightarrow q$ **then**

9 $\sigma_i \rightarrow Next = new\_node()$; $\sigma_i = \sigma_i \rightarrow Next$; $\sigma_i \rightarrow q = k_{im} \rightarrow q$;

10 $\sigma_i \rightarrow \alpha = k_{im} \rightarrow \alpha$; $\sigma_i \rightarrow QL = k_{im} \rightarrow QL$; $\sigma_i \rightarrow Next = NULL$;

11 **for** $l = a_1 \ldots a_\lambda$ **do**

12 $k_{il} = k_{il} \rightarrow Next$;

13 **if** $k_{il} \rightarrow Next == NULL$ **then** $PS = PS \setminus PS_{il}$;

14 Return ρS_i ;

noted that as the all the linked lists in PS are sorted in increasing order of consumed RBs, this subset is obtained by comparing only the current heads of the linked lists in PS. Then, in step 7, the algorithm finds that solution node (pointed to by k_{im}) which fetches the maximum quality (q) among all nodes in the set $\{k_{ia_1}, k_{ia_2}, \ldots k_{ia_\lambda}\}$. In steps 9 to 10, the solution corresponding to k_{im} is appended at the tail of the partially generated linked list ρS_i, provided this addition preserves the non-dominance properties of ρS_i. This is checked through the condition in step 8. It may be noted that other than k_{im}, no other node provided by the set $\{k_{ia_1}, k_{ia_2}, \ldots k_{ia_\lambda}\}$ can possibly contribute a non-dominating solution in ρS_i. Hence, all these nodes are discarded from further consideration in step 12 of the for-loop in step 11 to 13. If the removal of a node (in step 12) causes a linked-list (say, PS_{il}) in PS to become empty, then PS_{il} is removed form PS (step 13). The final partial optimal solution (ρS_i) considering upto i

flows $(f_1, f_2, \ldots, f_i)$ and various distinct bounds on the number of usable RBs, is obtained in step 14.

5.2.3.4 Approximation Algorithm (SDQA-AA)

As discussed above, the *SDQA* strategy memoizes only non-dominant partial optimal solutions which provide distinct enhancements in quality with increasing values of α. However, it may be noted that when the enhancement in quality is small, the corresponding effect on the perceived viewing quality of the end-user will usually be negligible. A typical measure for representing perceptual video quality (recommended by International Telecommunications Union) is *Mean Opinion Score* (MOS) [88]. *MOS* partitions the range of perceptual video quality values measured in terms of *PSNR*, into five distinct classes; PSNR$>$ 37 representing *excellent quality*, PSNR range $31 - 36.9$: *good quality*, PSNRs $25 - 30.9$: *fair quality*, PSNRs $20 - 24.9$: *poor quality* and PNSR$<$ 19.9: *bad quality*. It may be observed from the table that each class consists of a broad range of *PSNR* values and small variations in a flow's quality (*PSNR*) will usually not result in a notable degradation in the perceived viewing experience of the end-user. However, by disregarding partial solutions that donot provide significant gains in quality in the *SDQA* algorithm, it may be possible to achieve large reductions in the computational complexity of its quality level adaptation process. With this insight, we have developed an approximation algorithm based on *SDQA* called *SDQA-AA*. The primary objective of this algorithm is to obtain a trade-off between running time and optimization accuracy. *SDQA-AA* first executes a preprocessing step. The output of the *preprocessing step* is fed to the *SDQA* procedure (Algorithm 8).

Preprocessing Step: In this step, the *SDQA-AA* strategy rounds down the quality values of each flow (q_{il}) to the nearest multiple of a user specified tunable parameter $m \in \mathbb{Q}^+$. The parameter m is known as *Approximation Quality Index* and its value decides the degree of approximation. Let $\hat{q_{il}}$ be the rounded quality

value of the i^{th} flow at the l^{th} quality level. $\hat{q_{il}}$ is calculated as:

$$\hat{q_{il}} = f\left(\frac{q_{il}}{m}\right) \ \forall i \ \forall l \tag{5.9}$$

where, $f(x)$ is a modified floor function which rounds down any value x to its nearest multiple of m i.e.

$$f(x) = p \times m \quad \textbf{if } p \times m \leq x < (p+1) \times m \ \ \forall p \in \mathbb{Z}^+ \tag{5.10}$$

Once, the $\hat{q_{il}}$ values for all flows are calculated, the algorithm executes *SDQA* with the inputs $\hat{q_{il}}$, r_{il} and R. The asymptotic worst-case time complexity analysis of the *SDQA-AA* algorithm is presented in theorems 5.2.1.

Theorem 5.2.1. *The asymptotic worst-case time complexity of the SDQA-AA strategy is equal to* $\mathcal{O}\big(N \times L_i\big) + \mathcal{O}\big(\boldsymbol{min}\{\Pi_{i=1}^{N}(L_i+1),\ \sum_{i=1}^{N} \boldsymbol{min}\{R, \frac{(\rho SN_{ig_i} \rightarrow q)}{m}\}\}\big)$.

Proof. As discussed above, the *SDQA-AA* algorithm works in two steps. *SDQA-AA* first executes a preprocessing step and then, the output of the first step is fed as input to the *SDQA* procedure (i.e. step 2). Therefore, the computational complexity of the *SDQA-AA* strategy is equal to the total worst-case time complexity of both the steps, which is given in lemmas 5.2.2 and 5.2.3, respectively. □

Lemma 5.2.2. *The complexity of the preprocessing step of SDQA-AA is* $\mathcal{O}\big(N \times L_i\big)$*, where* N *is the total number of flows and* L_i *is the total number of quality levels available for the* i^{th} *flow.*

Proof. In the preprocessing step, *SDQA-AA* rounds down the quality values of all the quality levels of each flow to the nearest multiple of m, the *Approximation Quality Index*. Each round down operation takes constant time. Therefore, the complexity is $\mathcal{O}\big(N \times L_i\big)$. □

Lemma 5.2.3. *After the preprocessing step, complexity of the SDQA algorithm (i.e. with inputs* $\hat{q_{il}}$*,* r_{il} *and* R*) is bounded by* $\mathcal{O}\big(\boldsymbol{min}\{\Pi_{i=1}^{N}(L_i+1),\ \sum_{i=1}^{N} \boldsymbol{min}\{R, \frac{(\rho SN_{ig_i} \rightarrow q)}{m}\}\}\big)$*, where* Π *denotes the multiplication operation and* g_i *is the total number of nodes available in the partial solution* ρS_i*.*

Proof. It may be noted that the computational complexity of the *SDQA* algorithm ultimately depends on the total number of intermediate nodes processed by it to obtain an optimal solution. This is equal to the total number of times the inner loop lines 5 to 10 of Algorithm 8 get executed. As discussed above, the preprocessing step rounds down the quality value of each flow to the nearest multiple of *Approximation Quality Index* (m). Therefore, for any particular value of i (flow index) and p (node index), the attribute value for quality $\rho SN_{ip} \rightarrow q$ is also a multiple of m. Now, if $\rho SN_{ig_i} \rightarrow q$ represents the value of the quality attribute corresponding to the last node of the partial solution ρS_i, then the total number of distinct quality values possible in ρS_i is equal to $(\rho SN_{ig_i} \rightarrow q)/m$. Additionally, the RB demand for all the nodes in ρS_i must always be less than the total number of available RBs i.e. $\rho SN_{ip} \rightarrow \alpha < R \; \forall i \; \forall p$. Also, as discussed before, all nodes in ρS_i contain distinct q and α values. Hence, the total number of nodes in ρS_i is upper bounded by $\mathbf{min}\{R, \frac{(\rho SN_{ig_i} \rightarrow q)}{m}\}$. Therefore, total number of nodes processed by the *SDQA* algorithm (in *SDQA-AA*) over all its iterations is upper bounded by $\sum_{i=1}^{N} \mathbf{min}\{R, \frac{(\rho SN_{ig_i} \rightarrow q)}{m}\}$. It is obvious that the complexity of *SDQA* is always less than or equal to the run time of the exhaustive search $\Pi_{i=1}^{N}(L_i + 1)$, where L_i is the total number of quality levels available for the i^{th} flow. Hence, the complexity of the *SDQA* strategy in *SDQA-AA* is bounded by $\mathcal{O}\big(\mathbf{min}\{\Pi_{i=1}^{N}(L_i + 1), \; \sum_{i=1}^{N} \mathbf{min}\{R, \frac{(\rho SN_{ig_i} \rightarrow q)}{m}\}\}\big)$. □

5.3 Experiments and Results

The performance of the proposed *Adaptive Video Streaming Framework (AVSA)* has been experimentally evaluated against various performance parameters and compared with a popular adaptive video streaming framework called *AVIS* [17]. Performance evaluation is based on simulation studies carried out using *LTE-Sim* [34], an open source simulator for *LTE* networks. *LTE-Sim* has allowed us to create a realistic single-cell with interference scenario working in *Frequency Division Duplex (FDD)* mode with a total available bandwidth of 20 MHz per cell.

QL	QP	Resolution	Avg. Bit-rate (KBps)	$\overline{\text{PSNR}}$
9	10	352×288	218.3	51.8
8	16	352×288	92.5	48.5
7	22	352×288	43.7	45.2
6	24	352×288	33.9	44.0
5	28	352×288	21.0	41.7
4	34	352×288	10.4	38.1
3	38	352×288	6.8	35.5
2	42	352×288	4.5	32.9
1	48	352×288	2.8	31.9
0	48	176×144	1.5	28.3

Table 5.2: *Star Wars video trace [1]*

A cell (of radius 1 km) may contain a variable number of mobile UEs (10 to 50 UEs have been considered) which travel following the random direction mobility model [77] with a speed of 3kph. Each *UE* receives one video flow. Network topology for the generated scenario is depicted in Figure 5.4. We have considered video flows of duration 300 secs (generated from video traces Star Wars, Silence of the Lambs and Tokyo [1]) encoded at 25 frames/sec with 10 distinct quality levels and segmented at 1 sec playback intervals. Hence, the server transmits $\sim$ 7500 frames per video flow during the entire simulation duration. Table 5.2 presents an example of a typical video flow (namely, Star Wars) encoded at different Quantization Parameter (QP) values and resolutions. In the experiments, values of the constants $|AI|$ (adaptation interval duration) and β (refer equation 5.3) have been taken to be 1 sec and 0.75, respectively. We have conducted two sets of experiments in order to analyze the effect of switching stability moderation (refer subsection 5.2.2). In the first set of experiments (discussed in section 5.3.1) switching stability moderation has been ignored. A second set of experiments has then been carried out with stability moderation at two distinct switching bounds ($Th\_sw_i = 0.25$ and $Th\_sw_i = 0.5$) and their effect on overall QoE were analyzed.

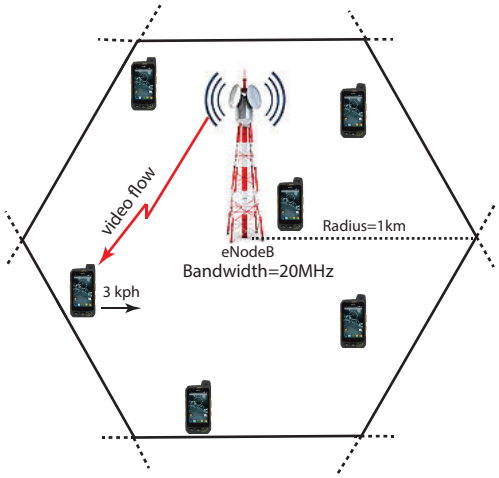


Figure 5.4: *Network Topology for single cell with interference*

5.3.1 Simulation Results

A series of experiments have been conducted in order to measure the performance of *AVSA* in terms of different aspects of the *QoE* achieved by the system under varying scenarios. The performance achieved by the *AVSA* framework using the three proposed adaptation strategies, namely, (i) the *DP-based Quality-level Allocator (DPQA)*, (ii) the *Streamlined DP-based Quality-level Allocator (SDQA)* and (iii) the *Approximation Algorithm (SDQA-AA)* at distinct values of approximation quality index (m), have been measured. Additionally, *AVSA* is compared with a popular adaptive streaming framework called *AVIS* [17] using its two adaptation strategies, namely, (i) *Dynamic programming approach for AVIS* (*AVIS-DP*) and *Greedy approach for AVIS* (*AVIS-GREEDY*).

Figure 5.5 depicts plots for $\overline{Buffering}$ %. It may be observed that $\overline{Buffering}$% is approximately same for all the quality level allocation schemes (DPQA,SDQA and *SDQA-AA* at distinct values of m) of the *AVSA* framework. It is also approximately constant with increasing total number of video flows (system load). Maintenance of stable buffer sizes is achieved mainly by a combination of two principal mechanisms: (i) Throttle rate adjustment for each flow based on instantaneous buffer size by *TRC* and (ii) Cell capacity cum traffic load aware quality level

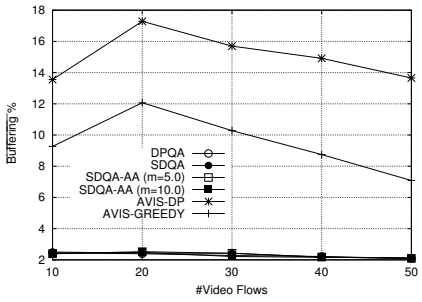


Figure 5.5: $\overline{Buffering}$ *% Vs. #Video Flows*

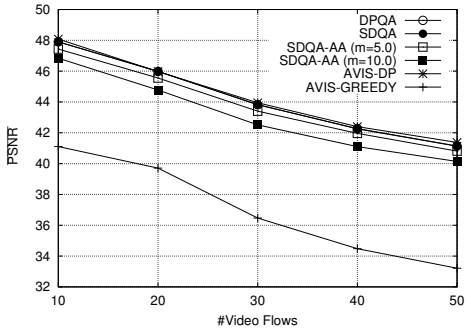


Figure 5.6: $\overline{PSNR}$ *Vs. #Video Flows*

allocation by *AVRC*. On the other hand, both the adaptation strategies of AVIS (namely, *AVIS-DP* and *AVIS-GREEDY*) encounter higher $\overline{Buffering}$%. This is because, their adaptation strategies are ignorant of client-side buffer status.

Fig 5.6 depicts the plots for the average transmitted video qualities ($\overline{PNSR}$) by the different quality level adaptation strategies, as the number of video flows vary from 10 to 50. It may be noted that as a consequence of being adaptive, the trends for $\overline{PNSR}$ for all the adaptation methodologies are decreasing as the number of flows (system load) increases. This is expected because the average number of RBs that may be allotted for a video flow reduces as the total number of video flows increases under a fixed resource block budget within an adaptation interval. With a lower number of available RBs per flow, buffer outage events may only be controlled by degrading *quality* levels of the flows. Although the trends for all the adaptation algorithms in Figure 5.6 are similar, $\overline{PNSR}$ achieved by *AVIS-GREEDY* is lowest among all the adaptation strategies due to greedy decisions in its quality level/bit-rate selection process. On the other hand, the *AVIS-DP*, *DPQA*, *SDQA* strategies provide highest and approximately same $\overline{PNSR}$ due to optimal decisions in their adaptation process. However, a closer look reveals that *AVIS-DP* slightly outperforms *DPQA* and *SDQA*. This is expected because *DPQA* and *SDQA* operates within *AVSA* which also attempts to maintain a stable playout buffer sizes for all clients by appropriately adjusting their throttling

Table 5.3: *Comparative results for average run time (in millisecs)*

Video Flow	*DPQA*	*SDQA*	*SDQA-AA*				*AVIS-DP*	*AVIS-GREEDY*
			$m = 2.5$	$m = 5.0$	$m = 10.0$	$m = 15$		
10	1282.0	15.6	3.0	1.0	1.0	0.055	1511.0	0.012
20	3648.1	131.5	12.5	6.6	3.9	3.000	4227.7	0.018
30	6669.0	387.6	28.1	15.8	9.2	7.727	8498.1	0.026
40	10247.1	836.2	44.7	24.3	14.7	12.018	11697.6	0.030
50	14468.8	1474.6	66.3	36.4	22.1	18.618	16459.3	0.036
Considered system with one core and 2.5 GHz processing capacity								

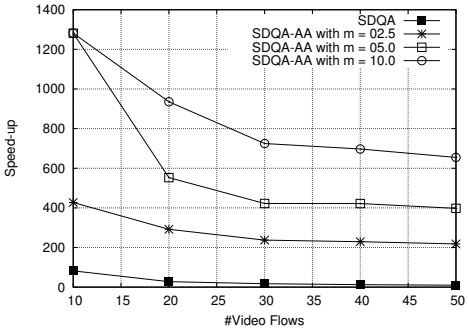


Figure 5.7: *Speed-up Vs. #Video Flows*

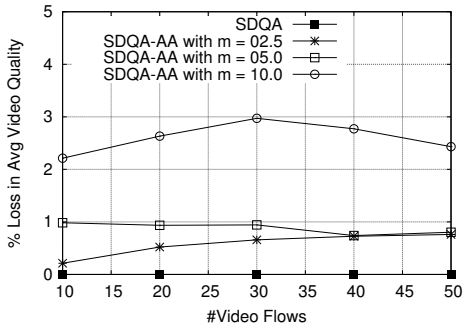


Figure 5.8: *% loss in video Quality Vs. #Video Flows*

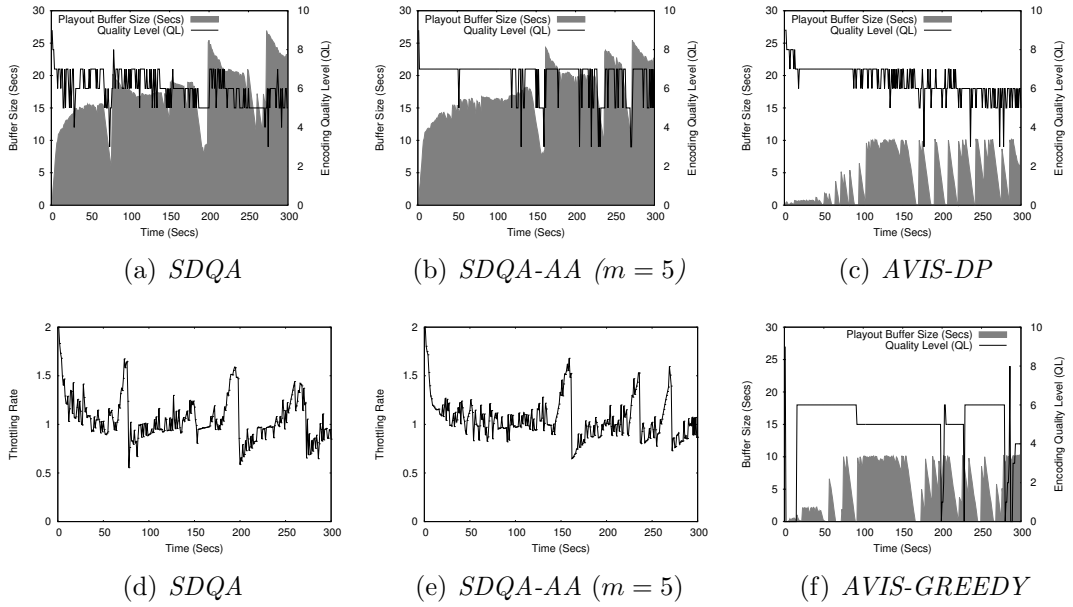
rates. Additionally, it may be observed from Figure 5.6 that the *SDQA-AA* adaptation scheme provides slightly reduced $\overline{PNSR}$ with respect to the *DPQA* and *SDQA* algorithms due to approximation in the memoization process. Moreover, the performance of *SDQA-AA* degrades with increasing values of approximation quality index (m). Such degradation occurs because, with larger values of m, *SDQA-AA* applies a higher degree of approximation in the memoization process.

Table 5.3 shows the average run times (measured in millisecs) taken by the *DPQA*, *SDQA*, *SDQA-AA*, *AVIS-DP* and *AVIS-GREEDY* strategies as the number of video flows vary from 10 to 50. It may be observed from the table that the average execution time of *DPQA* and *AVIS-DP* are comparatively much higher than the other strategies. This is because both these strategies are based on conventional dynamic programming which calculates and memoizes partial so-

lutions for all possible bounds on number of flows ($\forall i \in [0, N]$), quality levels ($\forall l \in [x_i, y_i]$) and RBs ($\forall \alpha \in [0, R]$). On the other hand, the *SDQA* scheme memoizes only those partial optimal solutions which provide distinct enhancements in quality with increment in the bound on the number of RBs α, for each value of i. As a result, *SDQA* provides significant reduction in computational overhead. In comparison, *SDQA-AA* achieves further significant reductions in execution time due to the approximation applied. As the value of the approximation quality index (m) increases, the *SDQA-AA* strategy is able to obtain higher reductions in the number of memoized partial solutions, which reflects as lower required run time. The average run time taken by the *AVIS-GREEDY* strategy is lowest among all the adaptation strategies. This is because, the computational complexity of *AVIS-GREEDY* only depends on the available number of flows and the number of available quality levels. Fig 5.7 portrays the execution speed-ups achieved by *SDQA* and *SDQA-AA* (at distinct values of m) over the *DPQA* algorithm, as the number of video flows vary from 10 to 50.

On the other hand, Figure 5.8 depicts the plots for percentage loss in average video quality suffered by *SDQA-AA* (at distinct values of m) with respect to *DPQA*, as the number of video flows vary from 10 to 50. It may be observed that although as expected, the average performance of *SDQA-AA* degrades with increasing values of m, the degradation is not drastic. For $m = 5$, the loss in average video quality for *SDQA-AA* is less than 1% even for 50 users while the speed-ups obtained are more than ~ 400 times.

Figures 5.9(a), 5.9(b), 5.9(c) and 5.9(f) show instantaneous buffer sizes along with the corresponding quality level values achieved by *SDQA*, *SDQA-AA* ($m = 5$), *AVIS-DP* and *AVIS-GREEDY* strategies respectively, for a single flow (namely, Star Wars) over the entire simulation duration. The scenario considers a cell with 50 active flows. It may be observed from the figures that the *AVSA* based adaptation strategies (i.e. *SDQA* and *SDQA-AA*) are able to maintain stable playout buffer sizes throughout their transmission duration. However, the *AVIS* based adaptation strategies (i.e. *AVIS-DP* and *AVIS-GREEDY*) encounter frequent

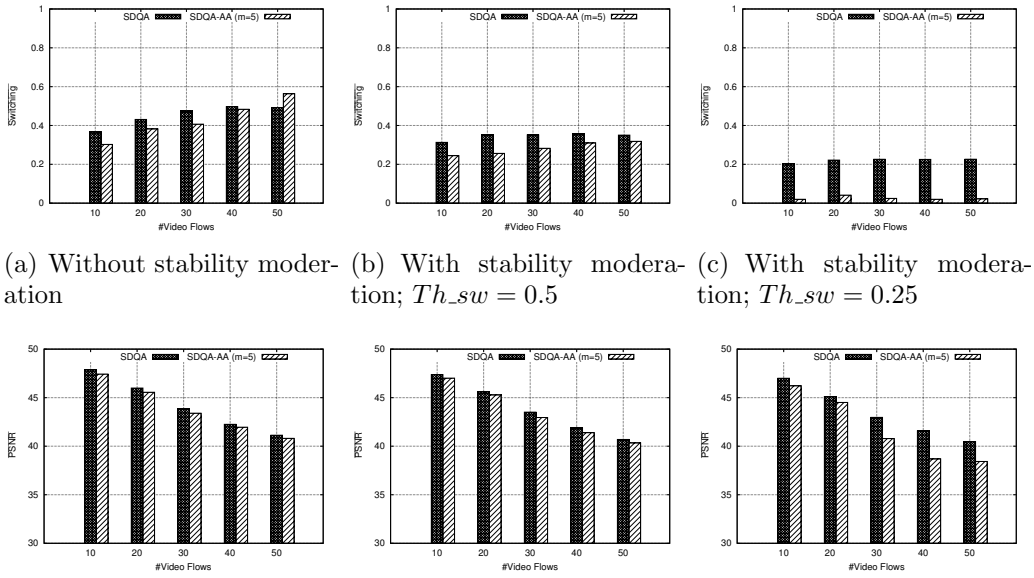


(a) *SDQA* (b) *SDQA-AA* ($m = 5$) (c) *AVIS-DP*

(d) *SDQA* (e) *SDQA-AA* ($m = 5$) (f) *AVIS-GREEDY*

Figure 5.9: *Comparative results of performance metrics Vs. Time for SDQA, SDQA-AA* ($m = 5$) *and AVIS*

re-buffering events because they are oblivious of instantaneous playout buffer sizes. From Figures 5.9(a) and 5.9(b) we see that initially (i.e. when the flow has just started), buffer size grows at a very fast rate. Such fast buffer growth rates during startup allows the system to achieve low startup delays, thus contributing to better overall *QoE* for all the adaptation schemes. Quick buffer ramp-ups for startup flows are mainly achieved by higher throttling rates (refer Figures 5.9(d) and 5.9(e)) during initial transmission phase of a flow. It may be seen that such a throttling rate control mechanism has been effective in maintaining stable buffer sizes for the flow over the entire simulation length. On the other hand, the *AVIS* based adaptation strategies are unaware of client-side buffer status and therefore, do not adjust throttling rates (the throttling rate remains constant at 1) during transmission. Figure 5.9(f) shows that being purely greedy in nature, *AVIS-GREEDY* delivers poorer average video quality levels ~ 5.5 (which corresponds to average bit-rate about 27.5 KBps) compared to *SDQA*, *SDQA-AA* and *AVIS-DP* which deliver video qualities of about ~ 6.5 (bit-rate ~ 37.8 KBps).



(a) Without stability moderation (b) With stability moderation; $Th\_sw = 0.5$ (c) With stability moderation; $Th\_sw = 0.25$

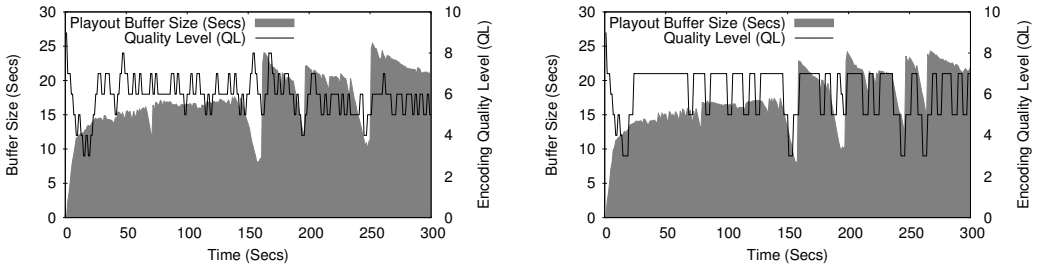
(d) Without stability moderation (e) With stability moderation; $Th\_sw = 0.5$ (f) With stability moderation; $Th\_sw = 0.25$

Figure 5.10: *Results for $\overline{Switching}$ and $\overline{PSNR}$ without stability moderation and with stability moderation (at $Th\_sw = 0.5, 0.25$)*

Additionally, it may be observed that *AVIS-GREEDY* may be subject to harsher quality level fluctuations compared to the other schemes.

5.3.2 Switching Stability Moderation by Varying Th_sw

Controlling the frequency of quality level switches is essential to avoid annoyance among users due to flickering video outputs. However as discussed, quality level switches are essential to maintain stable buffer sizes along with high overall video encoding quality within a limited radio bandwidth. For example, let us consider the result for the instantaneous video qualities over time shown in Figs 5.9(a) - 5.9(b). It may be observed from the figure that without switching stability moderation, a flow may suffer from frequent quality level switches in the effort to enhance video quality and avoid buffer outage. Hence, the *SSC* controller is employed in *AVSA* (refer Figure 5.2) to provide an upper bound on the average allowable number of quality level switches per adaptation interval



(a) *SDQA* (b) *SDQA-AA* ($m = 5$)

Figure 5.11: *Quality levels and buffer status using SDQA and SDQA-AA ($m = 5$) after stability moderation with $Th\_Sw = 0.5$*

(Th_sw) for each flow. Figs 5.10(a) - 5.10(c) depict the results for average quality level switches per adaptation interval ($\overline{Switching}$) encountered by each flow without stability moderation and with stability moderation (at $Th\_sw = 0.5$ and $Th\_sw = 0.25$). The figure shows that as switching threshold (Th_sw) decreases, switching rate also decreases in general, irrespective of the adaptation strategy. From Figs 5.10(d) - 5.10(f), it may be observed that as a natural consequence of controlling quality level switching within a given bound, the average transmitted video quality over the flows gradually decreases with decreasing Th_sw values. However, the degradation in transmitted video quality is not drastic compared to the obtained reduction in switching rate. For example, for the scenarios with 50 active video flows, $\overline{PSNR}$ decreases by 1.09% and 1.15% for *SDQA* and *SDQA-AA* respectively, when the experiments have been carried out with stability moderation at $Th\_sw = 0.5$. Correspondingly, the obtained reduction in switching rates are 28.7% and 43.5%, respectively. Comparing the instantaneous quality level plots in Figs 5.11(a)-5.11(b) with those in Figures 5.9(a)-5.9(b), it may be clearly seen that the stability moderation process has been able to control switching frequencies over time and reduce spikes in the plot to a large extent. Also, it may be noted that for both the strategies, the system is able to maintain stable and satisfactory buffer sizes throughout even when stability moderation is applied.

5.4 Summary

In this chapter, a new adaptive video streaming framework for *LTE* systems has been introduced. The framework is aimed at enabling a service provider to deliver high quality video viewing experience to all end-users by achieving a judicious balance between various *QoE* parameters including: (i) Stutter-free video playout, (ii) Encoding quality and (iii) Stability against bit-rate switches. The problem of scheduling a set of adaptive video streams over *LTE* has been formulated as an *ILP* and a conventional dynamic programming solution (DPQA) has been shown to impose substantial overheads. A new algorithm *SDQA* which memoizes only non-dominating intermediate partial solutions has been designed to accelerate *DPQA*. Further, an approximation algorithm for *SDQA* (called as *SDQA-AA*) has been developed. Simulation results reveal that *SDQA-AA* achieves drastic speed-ups over optimal strategies *DPQA* and *SDQA* with only slight degradations in *QoE*.

5. A RESOURCE ALLOCATION FRAMEWORK FOR ADAPTIVE VIDEO STREAMING OVER LTE

Chapter 6

A QoE Aware SVC Based Client-side Video Adaptation Framework

6.1 Introduction

The previous contributory chapters focused towards the design of efficient resource allocation strategies in the network, primarily at eNodeB. In-network resource allocation schemes tend to be superior towards effectively utilizing the available bandwidth while multiplexing radio resources among clients. However, in order to make good scheduling decisions, these resource allocators require periodic client status feedback including information about channel conditions, buffer size etc. A diametrically opposite approach to resource allocation are client-side schemes where each client attempts to adapt the streaming strategy and/or bitrates corresponding to the video flow it is receiving, with the typical objective of maximizing *QoE*. In this chapter, we have presented two client-side *Quality of Experience (QoE)* aware *SVC-DASH* based adaptation strategies which attempt to deliver satisfactory quality of video viewing experience to the end user even during fluctuating network conditions.

The first adaptation strategy (we called *Video Quality Adaptation Unit (VQAU)*) is designed as a *DES* supported approach which allows us to accurately model the discrete event dynamics corresponding to the *DASH* based video streaming

control framework considered in this work. In particular, *DES* has helped in precisely modeling relevant system components and their interactions under the influence of a set of events that are active at a given time. The principal objective of *DES* supported video quality adaptation unit is to maximize the quality of video viewing experience of an end user by dynamically taking one of the following two actions at any given state: (i) download a new segment at a selected enhancement level or, (ii) upgrade (smooth-out) the enhancement level of an already downloaded segment in the playout buffer by a stipulated value. In order to avoid playback interruptions even under fluctuating bandwidth conditions, *VQAU* performs upgradation/smoothing over downloaded video segments only when the playout buffer size attains an upper safety threshold (q_s^+ in seconds). Once q_s^+ is reached, *VQAU* continues smoothing for a fixed time interval T without downloading any new segment until the buffer size reduces to a lower threshold q_s^- (i.e., $T = q_s^+ - q_s^-$). This smoothing action has a two-fold objective: (i) improve overall playback encoding qualities through enhancement level upgradation and (ii) smooth-out bit-rate switching (difference in enhancement levels) between consecutive segments to reduce flickering. In this work, we have posed the problem of smoothing a set of eligible segments in the buffer (given, time constraint T) as an optimization problem. This problem has been solved through a novel low-overhead variant of the conventional Dynamic Programming (*DP*) approach which we have named as the *Streamlined Enhancement-level Smoother (SES)* strategy.

In the second scheduling strategy, we have formulated the rate adaptation of *SVC-DASH* as a *Markov Decision Process (MDP)* due to its ability to take optimal decisions under uncertainty [27]. The states of the proposed *MDP* are defined based on the playout buffer sizes. At any given state, the adaptation agent takes one of the following two actions: (i) download a new segment at a selected enhancement level or, (ii) upgrade (smooth-out) the enhancement level of an already downloaded segment in the playout buffer by a stipulated value. The action selected at a given state depends on the maximum combined reward

(defined in terms of obtainable *QoE*) derived from: (i) the rewards achieved by probabilistically reaching a set of destination states, and (ii) the maximum rewards derived through the best actions from those destination states. On the other hand, the probability of reaching a state depends on three factors: (i) constant depletion rate of the buffer due to the ongoing playback, (ii) estimated value of the currently received bandwidth and (iii) approximate data demand corresponding to chosen action. Among these, the second and third factors determine the estimated time that will be required to complete the current action. The combination of the three factors therefore, provides the estimated effective buffer size (destination state) that will be reached at the completion of the current action. The reward for a particular action is governed by the overall *QoE* that may be obtained by reaching a certain destination state by taking the action. Because each action is associated with an estimated time budget, the action at a given state partially affects the choice of actions that may be taken at subsequent future states. Therefore, the best possible action at a given state should not only depend on a myopic view of the maximum immediate reward, but also on the effect it has on obtainable future rewards.

The next section discusses the working principle of the Video Quality Adaptation Unit (VQAU) in detail.

6.2 Video Quality Adaptation Unit (VQAU)

Video Quality Adaptation Unit (VQAU) is a *SVC-DASH* based flexible bit-rate and throttle rate selection framework which attempts to deliver satisfactory *QoE* to the end user even during significantly varying network conditions. The objective of any *QoE* aware adaptation agent should be to maximize encoding qualities of the video segments while maintaining playout buffer sizes above a safe threshold and restricting the degree of switching as far as possible. In order to meet this objective, the proposed *VQAU* framework attempts to achieve a judicious balance among all the above mentioned parameters.

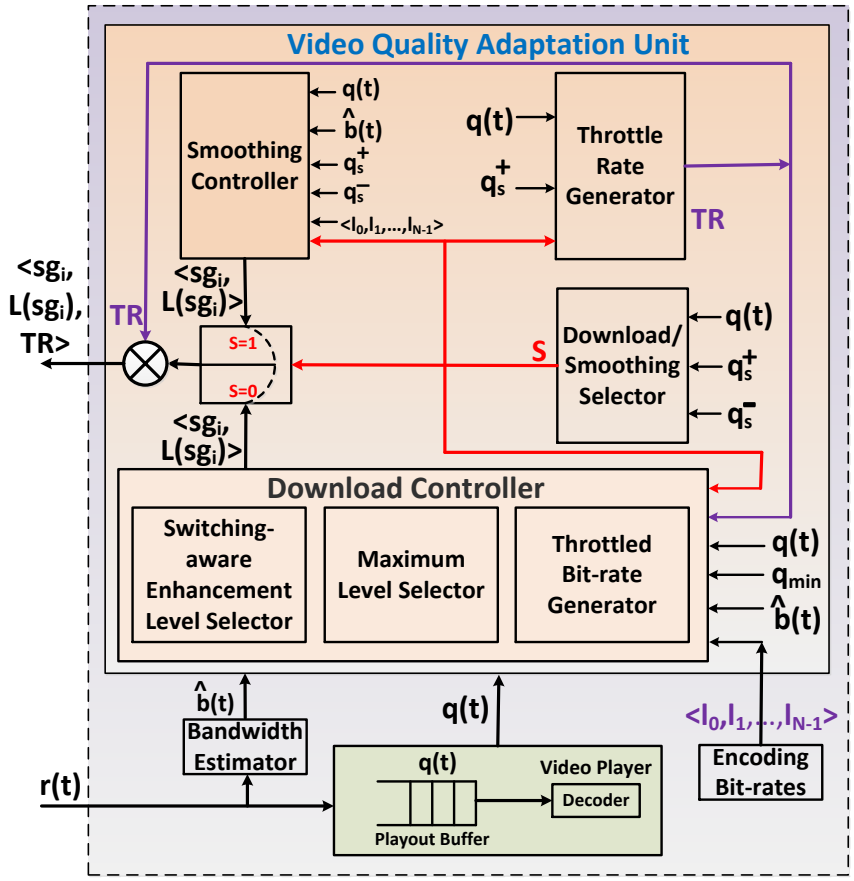


Figure 6.1: *Architecture of Video Quality Adaptation Unit*

The architecture of the overall $VQAU$ framework is shown in Figure 6.1. $VQAU$ selects the enhancement level ($L(sg_i)$) at which an appropriate video segment (sg_i) should be fetched (downloaded/smoothed) with a designated throttle rate (TR), so that the overall QoE of the end user is maximized. The exact values of these variables in the next GET request is determined based on the following information as shown in Figure 6.1:

- *Current playout buffer status* ($q(t)$): Instantaneous buffer status information consists of the following important components: (i) number of segments in the buffer, (ii) enhancement levels of the segments and (iii) enhancement level switches between consecutive segments.

- *Estimated bandwidth* ($\hat{b}(t)$): This is computed by the *Bandwidth Estimator* using the following equation:

$$\hat{b}(t) = \gamma \times \hat{b}(t-1) + (1-\gamma) \times r(t-1)$$

where, $\hat{b}(t-1)$ and $r(t-1)$ are the estimated and actual throughput received by the client for the last segment fetched and γ is a positive constant which controls bandwidth estimation smoothness.

- *Encoding bit-rates* ($\langle l_0, l_1, ...l_{N-1} \rangle$): Encoding bit-rates corresponding to the enhancement levels of the video segments under consideration (obtained by parsing the *MPD* file).

As the block diagram (Figure 6.1) shows, *VQAU* is composed of the following sub-modules: (i) Download/Smoothing Selector, (ii) Throttle Rate Generator, (iii) Download Controller and (iv) Smoothing Controller. Based on the current buffer size (provided by $q(t)$), the *Download/Smoothing Selector* decides whether to *download* or *smooth* at the next adaptation step. If the current buffer size is below a pre-specified threshold, the *Throttle Rate Generator* in *VQAU* adjusts TR to quickly ramp-up data transmission rate from the server. If the *download action* has been chosen, the *Download Controller* selects an appropriate enhancement level $L(sg_i)$ for the next segment sg_i. On the other hand, if the *smoothing action* is chosen, the *Smoothing Controller* selects a particular segment sg_i in the buffer and upgrades its encoding quality to a desired enhancement level $L(sg_i)$ with the objective of maximizing *QoE* (by lowering flickers and/or improving mean encoding quality of the played video).

Finally, the *GET* request $\langle sg_i, L(sg_i), TR \rangle$ for the current adaptation step is formed by the output of *VQAU*. This output is generated by merging TR with $\langle sg_i, L(sg_i) \rangle$, produced either by the *Download Controller* or *Smoothing Controller* depending on whether *VQAU* chooses to perform download or smoothing, as shown in Figure 6.1. It may be noted that the *DES* supported modeling of *VQAU* makes its design modular and flexible towards easy adapta-

tion/reconfiguration in case of modifications in design policy. Next, we present the details of each sub-module in *VQAU*.

6.2.1 Download/Smoothing Selector (DSS)

As discussed previously, playback interruptions is critical factor which adversely affects the *QoE* of an end user. *VQAU* attempts to avoid such interruptions by always maintaining playout buffer sizes above a lower safety threshold (q_s^-). The download/smoothing selector in *VQAU* controls the buffer size by choosing either to *download* or *smooth* in a mutually exclusive fashion at each adaptation step. We describe the behavior of this module through a *DES* model which is formally represented by a six tuple $(X, X_0, I, \delta, O, H)$ where:

- a set of states, $X = \{\text{DOWNLOAD, SMOOTHING}\}$;
- an initial state, $X_0(\in X) = \text{DOWNLOAD}$;
- a set of inputs, $I = \{q(t),\ q_s^+,\ q_s^-\}$;
- a transition relation, $\delta : X \times I \mapsto X$;
- a set of outputs, O: $\{\text{S} = 0,\ \text{S} = 1\}$;
- an output map $H : X \mapsto O$. That is, $H(\text{DOWNLOAD})$: $S = 0$ and $H(\text{SMOOTHING})$: $S = 1$.

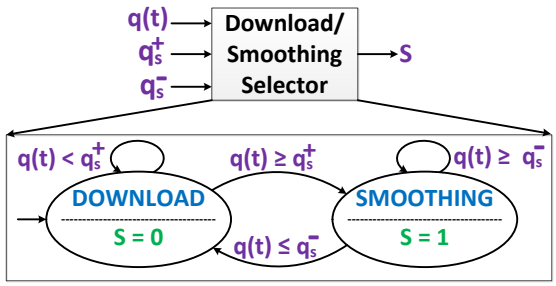


Figure 6.2: *Download/Smoothing Selector (DSS).*

Figure 6.2 presents the *DES* model *DSS* of the *Download/Smoothing Selector (DSS)*. Here, states are represented by circles and transitions are denoted by arrows between states. The initial state is distinguished by representing it as the target of a sourceless arrow. Each circle is labeled by the state (top half) and the corresponding output (bottom half). Each arrow is labeled by the constraint which must be satisfied to effect the corresponding transition.

Initially, at the DOWNLOAD state when $S = 0$, only downloads are allowed while smoothing is disallowed. When size of the playout buffer reaches the upper threshold limit q_s^+, DSS transits to SMOOTHING where $S = 1$. At this state, *VQAU* performs smoothing while disallowing downloads until the playout buffer size reduces below the lower threshold limit q_s^-. It may be noted that the smoothing is performed only when $q_s^+ \leq q(t) < q_s^-$. Thus, values of the thresholds q_s^+ and q_s^- must be carefully chosen to avoid frequent switching between the two states of *DSS*.

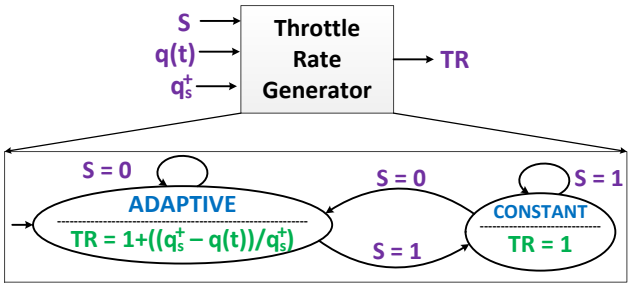


Figure 6.3: *Throttle Rate Generator (TRG).*

6.2.2 Throttle Rate Generator (TRG)

Throttling [86] is a well-known technique in which data transmission rate for a video flow from the sever is boosted (with respect to their encoding bit-rates) in order to quickly ramp-up playout buffer size at the client whenever necessary.

The *DES* model *TRG* of the *throttle rate generator* is shown in Figure 6.3. This model operates based on the inputs S, $q(t)$ and produces output TR. Specifically, when video segment download/smoothing selector selects *downloading* op-

eration ($S = 0$), TRG sets the throttling rate as, $TR = 1 + \frac{q_s^+ - q(t)}{q_s^+}$. Otherwise, TR is set to 1 when $S = 1$ (*smoothing*).

6.2.3 Download Controller (DC)

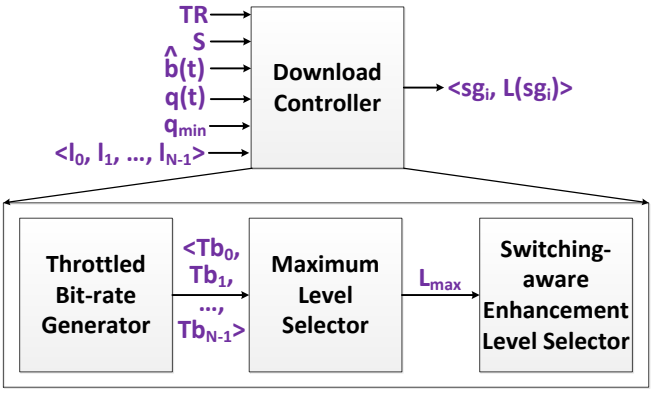


Figure 6.4: *Download Controller (DC).*

If the *DSS* module triggers *download* action ($S = 0$), then $VQAU$ enables the *Download Controller (DC)* to select an enhancement level $L(sg_i)$ corresponding to the next segment to be downloaded (sg_i). Figure 6.4 shows the overall block diagram of the *Download Controller (DC)*. *DC* is composed of three principal components: (i) the *Throttled Bit-rate Generator (TBG)* which calculates the download transmission rates corresponding to each enhancement level of sg_i based on the *throttle rate (TR)* determined by *TRG*, (ii) the *Maximum Level Selector (MLS)* which conducts a combined bandwidth and buffer aware estimation of the maximum enhancement level (L_{max}) that may be selected for sg_i and (iii) the *Switching-aware Enhancement Level Selector (SELS)* which determines the exact enhancement level $L(sg_i)$ by applying an appropriate penalization component on the maximum level that may possibly be selected based on the degree of switching.

Throttled Bit-rate Generator (TBG): Based on the selected throttle rate TR, throttled bit rates Tb_j for each distinct enhancement level (with encoding rate l_j) is calculated ($Tb_j = TR \times l_j$), as shown in Figure 6.5. *TBG* receives

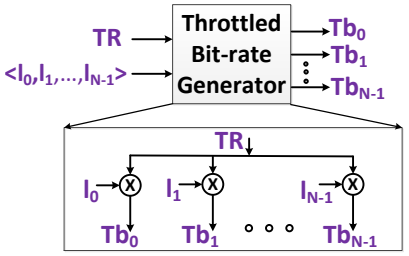


Figure 6.5: *Throttled Bit-rate Generator (TBG).*

information about different encoding rates of a video flow from its corresponding MPD (Media Presentation Description). The set of throttled bit rates generated by this unit is fed as input to *MLS*.

Maximum Level Selector (MLS): Figure 6.6 shows the *DES* model of *MLS*. When enabled ($S = 0$), *MLS* starts from the initial state (*Level 0*) either at the commencement of the video streaming process or whenever the value of S flips from 1 to 0 (*DSS* moves from *SMOOTHING* to *DOWNLOAD* state). *MLS* selects $L_{max} = l_0$ (*Level 0*) whenever the buffer size is less than a minimum threshold value q_{min}. This is indicated by the incoming transitions to the state *Level 0* on the condition $q(t) < q_{min}$. This action enables fast segment downloads at the base enhancement level to allow quick buffer replenishment during transient intervals of critically low buffer sizes. Otherwise, *MLS* sets L_{max} to enhancement level l_j if throttled bit-rate $Tb_j \geq \hat{b}(t) > Tb_{j+1}$. From *DES* models *TRG* and *TBR*, it may be observed that throttled bit-rates Tb_j corresponding to the available enhancement levels l_j are inversely proportional to the instantaneous buffer size. This makes *MLS* a combined bandwidth cum buffer size aware enhancement level estimation mechanism. Thus, the probability of selecting higher enhancement levels increase as buffer sizes and/or bandwidth conditions improve.

Switching-aware Enhancement Level Selector (SELS): It may be noted that enhancement level L_{max} selected by *MLS* is ignorant of the degree of switching and such unrestricted switching may lead to flickering video playbacks. Fre-

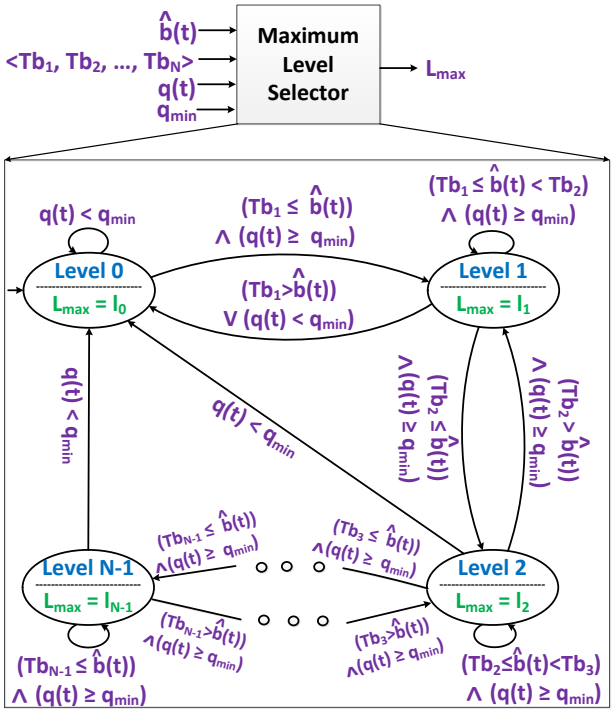


Figure 6.6: *Maximum Level Selector.*

quent flickers cause annoyance among end-user which ultimately pulls down the overall quality of viewing experience. *SELS* attempts to model this effect by defining *QoE* (QoE_{ij}^{d}) for a given enhancement level (say, level j of sg_i) as a linear combination of two components [89, 90]: (i) *PSNR* corresponding to the encoding quality at level l_j (Q_{ij}) and (ii) an appropriate penalization factor to incorporate the effect of switching. Thus,

$$QoE_{ij}^{d} = Q(sg_i, j) - \alpha \times |j - k| \times |Q(sg_i, j) - Q(sg_{i-1}, k)| \tag{6.1}$$

In the above equation, $Q(sg_{i-1}, k)$ denotes the encoding quality of the last downloaded segment sg_{i-1} at the k^{th} enhancement level and α denotes a *weighting parameter* (impact of varying α on the overall obtained *QoE* has been studied in the experiments section).

As shown in Figure 6.4, *MLS* feeds *SELS* the maximum enhancement level L_{max} at which segment sg_i may possibly be downloaded given the instantaneous buffer status and prevailing channel conditions. *SELS* then selects the exact

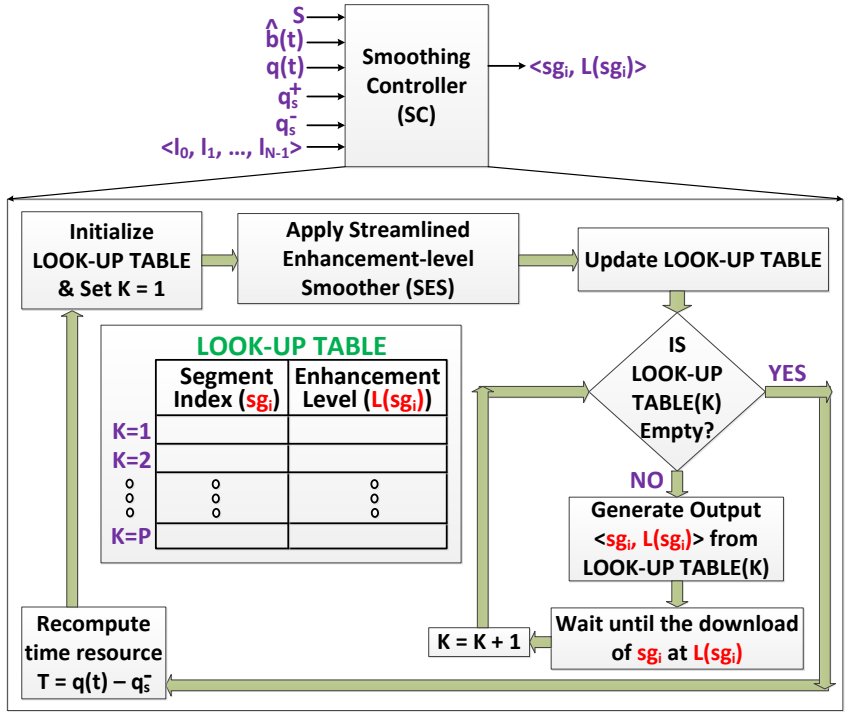


Figure 6.7: *Smoothing Controller.*

enhancement level ($L(sg_i) \leq L_{max}$) for which the obtained QoE is highest. That is,

$$L(sg_i) = \{l_{ij} | \max_j (QoE_{ij})\} \tag{6.2}$$

Thus, DC produces the final output as $\langle sg_i, L(sg_i) \rangle$.

6.2.4 Smoothing Controller (SC)

$VQAU$ enables the *Smoothing Controller (SC)* whenever buffer sizes are determined to be sufficient to ensure un-interrupted stutter-free playback ($S = 1$). Figure 6.7 depicts the block digram of SC. The objective of SC is to maximize the aggregate QoE corresponding to the already downloaded (but yet to be played) segments in the buffer by appropriately upgrading the enhancement levels of a selected subset of the segments.

The SVC video is logically divided into fixed duration segments and each such

segment is encoded and stored at various enhancement levels in the server. In *SVC-DASH*, a segment at the k^{th} enhancement level consists of the data corresponding to level $k-1$ along with additional data which enables the enhancement in encoding quality to the higher level k. Thus, for an enhancement level upgradation from level k to $k+l$ (say), *SVC-DASH* necessitates download of only the incremental data for the desired enhancement by l levels. This is unlike *MPEG-DASH* where such reuse of previously downloaded data is not possible during encoding quality upgradation. This results in bandwidth wastage not only because previously downloaded data for the segment becomes invalid, but also because the upgradation requires far heavier downloads compared to *SVC-DASH*.

Obviously, smoothing is conducted on a window W of P consecutive segments in the buffer (which are yet to be played) such that the minimum segment ID (say, x) in W is greater than η_s. Here, η_s is a threshold which is necessary to guarantee a minimum safety interval to ensure that the x^{th} segment will never reach its playout instant while it is still in the process of being enhanced. It may be observed from Figure 6.2 that smoothing is enabled whenever size of the buffer $q(t)$ becomes q_s^+ and is continued until buffer size reduces below q_s^-. Therefore, $T = q_s^+ - q_s^-$ is the total time for which *SC* can conduct smoothing over the P segments in window W. When the smoothing phase begins, each segment sg_i in W is at its initially downloaded enhancement level c_i.

Let t_{ij} be the time required (at the currently estimated available bandwidth) to enhance sg_i to level j from its initial level c_i. The time $t_{ij} = 0$, in case, $j = c_i$. A binary variable x_{ij} is equal to 1 ($x_{ij} = 1$) if the i^{th} segment is selected to be smoothed to level j. The objective of smoothing is to maximize the overall *QoE* by appropriately upgrading the segments in W such that the expression: $\sum_{\forall i \in W} \sum_{\forall j \geq c_i} QoE_{ij}^s \; x_{ij}$, is maximized, where, QoE_{ij}^s represents the *QoE* component corresponding to the smoothing of sg_i to level j. Here, QoE_{ij}^s is defined as a linear combination of two components: (i) the encoding quality (measured in terms of *PSNR*) corresponding to the playback of sg_i at the j^{th} enhancement level (Q_{ij}) and (ii) resulting switch in encoding quality with respect to the immediate

predecessor and successor of sg_i ($|q-p|+|p-r|$; p, q and r denote the smoothed enhancement levels for sg_{i-1}, sg_i and sg_{i+1}). Thus,

$$QoE_{ij}^{s} = Q_{ij} - \beta(|q-p|+|p-r|) \tag{6.3}$$

where, β is a switching penalty factor.

We then formulate the segment smoothing problem as:

$$\begin{aligned}
\text{Maximize} \sum_{i=1}^{P}\sum_{j=c_i}^{N-1} \Bigg(& Q_{ij}\ x_{ij} - \\
& \beta\Big(\Big|\sum_{j_1=c_i}^{N-1} Q_{ij_1}\ x_{ij_1} - \sum_{j_1=c_{i-1}}^{N-1} Q_{(i-1)j_1}\ x_{(i-1)j_1}\Big| \qquad (6.4a)\\
& + \Big|\sum_{j_1=c_i}^{N-1} Q_{ij_1}\ x_{ij_1} - \sum_{j_1=c_{i+1}}^{N-1} Q_{(i+1)j_1}\ x_{(i+1)j_1}\Big|\Big)\Bigg)
\end{aligned}$$

Subject to

$$\sum_{i=1}^{P}\sum_{j=c_i}^{N-1} t_{ij}\ x_{ij} \le T, \tag{6.4b}$$

$$\sum_{j=c_i}^{N-1} x_{ij} = 1,\ x_{ij} \in \{0,1\}, \forall i \tag{6.4c}$$

The first constraint given in Equation 6.4b guarantees that the total smoothing time over all segments in W is at most the available time T. The second constraint given in Equation 6.4c ensures that exactly one enhancement level is selected for any segment.

SC dynamically smooths the segments in W by essentially solving the above *Constraint Satisfaction Problem (CSP)* in equation 6.4 whenever the value of S flips from 0 to 1. A closer look reveals that an optimal solution to the smoothing problem of selecting appropriate upgradation levels of the segments may be obtained as a composition of the optimal solutions to a set of its sub-problems and therefore, *DP* provides a natural solution mechanism. A step-by-step algorithm along with discussion of *DP* is given in section 4.2.2, Algorithm 6, page no. 96 in previous chapter. The *DP* based optimization procedure corresponding

to the definition in equation 6.4 can be represented by the following recursive formulation:

$$\overline{QoE}(i,\tau) = \begin{cases} 0, & \text{if } i = 0 \text{ or } \tau = 0 \\ \mathbf{max}_j\{\overline{QoE}(i-1,\tau), & \\ QoE^s_{ij} + \overline{QoE}(i-1,\tau - t_{ij})\}, & \forall j \in \{c_i, N-1\} | \tau \geq t_{il} \end{cases} \tag{6.5}$$

It may be noted that, in the above recursive formulation, τ may take any value within the continuous range 0 to T, i.e., $0 \leq \tau \leq T$. Given such continuous parameters (for example, time interval (0, T) in our case), *DP* typically proceeds by dividing such continuous intervals by a constant (say, *tick* of given resolution) and thus, effectively partitions the interval on a discrete scale. Thus, the number of distinct values that τ can assume is given by: $T' = T/tick$. It may be noted that bigger the value of the constant *tick*, lower becomes the partitioning resolution and this results in a degradation of the solution quality.

The computational complexity of *DP* is $\mathcal{O}(P \times N \times T')$ where, P is the number of segments, N is the total the number of enhancement levels available and T' is the total number of time steps. This complexity indicates that run-time of *DP* based solution quickly increase as the value of *tick* is reduced. In fact, overhead of the *DP* strategy proves to be quite expensive even for moderate values of *tick*. For example, in a system with number of segments in the smoothing window $P = 10$, smoothing time $T = 10$ secs and $tick = 0.001$ secs, the value of T' becomes 10000. Our experimental analysis shows that, given segment sizes of 2 secs duration and $N = 10$ enhancement levels, *DP* takes ~ 167 ms to generate a solution on a 2.5 GHz computing core. It is easy to understand that this overhead is significantly high as it wastes time which could have been utilize to conduct further smoothing.

An important observation in the optimization problem under consideration is that it is inherently discrete in nature. Thus, we do not obtain continuous improvements in aggregate *QoE* for each additional *tick* considered in the partial solution. Rather, for any given value of i in equation 6.5, the improvement in aggregate *QoE* has a step-wise nature as τ is increased from 0 to T'. Moreover,

the number of such steps (reflecting *QoE* improvements) is $<< T'$ and this is essentially because $\sum_i N - c_i << T'$. Consequently, a majority of the partial optimal solutions $QoE(i, \tau)$ do not return distinct values. Hence generic *DP*, which memoizes all partial solutions for each distinct value of i and τ (in equation 6.5) may suffer from high and unnecessary computational overheads. This overhead may be reduced (often drastically) through a more efficient implementation which memoizes only those partial optimal solutions which provide distinct enhancements in *QoE* with increment in the bound on time τ, for each value of i. Further, it may be observed that this reduced memoization do not cause solution optimality to be compromised. Here, we propose an efficient and optimal solution strategy called *Streamlined Enhancement-level Smoother (SES)*. A step-by-step description of streamlined dynamic programming along with its description is presented in section 5.2.3.3, algorithms 8 and 9, page no. 122 of the previous chapter.

6.3 MDP Based Video Adaptation: Problem Formulation

Many recent solutions formulated the rate adaptation algorithm for *DASH* using *MDP* [27]. In *MDP*, the agent learns and determines one of the available actions based on iterative interactions with its environment. In order to measure the effectiveness of an action, the environment provides a numeric reward value to the agent for each action. Based on the received reward, the agent gradually learns the optimal action to be taken at a specific system state.

A *MDP* is a 5-tuple $\{\mathcal{S}, \mathcal{A}, \mathcal{P}_a(s, s'), \mathcal{R}_a(s, s'), \gamma\}$ where, $\mathcal{S}$ is a finite set of system states, $\mathcal{A}$ is a finite set of actions, $\mathcal{P}_a(s, s')$ is a transition probability that action a in state s will lead to state s', $\mathcal{R}_a(s, s')$ is the immediate reward obtained after transition to state s' by taking action a in state s and γ is a discounting factor for the reward collected from future actions and states.

6.3.1 System State

In this work, state of the system is observed whenever the adaptation agent completes an action. A state $s_t(q) \in \mathcal{S}$ denotes the state reached after completion of the t^{th} adaptation step. Here, q denotes the instantaneous playout buffer size (in secs) and can take any value within the continuous range 0 (represents an empty buffer) to q_{max} (represents filled buffer) i.e. $0 \leq q \leq q_{max}$. Thus, the number of states in the system is potentially infinite. Therefore, q has been discretized by partitioning the range $(0, q_{max})$ into an integral number of intervals of a specific resolution.

6.3.2 Actions

The objective of this work is to maximize the *quality of video viewing experience* delivered to the end user. Here, *QoE* of a video flow has been considered to be composed of three important components. The first component is based on the encoding qualities/enhancement levels at which the video segments are played out. Higher the enhancement level, better becomes the perceived picture quality. The second parameter is a measure of the degree of variation in enhancement levels of consecutive segments, over a certain video playout duration. This variation which is typically referred to *switching*, causes flickers in the video output. Frequent switching between segments creates annoyance among users as the video image is perceived to continually vary intermittently in quick successions. The third and most important parameter is the buffer size, q. Too short a buffer size increases the possibilities of buffering events during which the client experiences video playback interruptions due to buffer outages. Thus, the objective of any *QoE* aware adaptation agent should be to maximize encoding qualities of the video segments while maintaining playout buffer sizes above a safe threshold (q^{th}) and restricting the degree of switching as far as possible.

Based on the current state say, $s_t \in S$ of the system buffer, the adaptation agent can choose to perform two types of alternative actions: (i) (α_d, n): denotes

the download of the next segment at the n^{th} enhancement level and (ii) (α_s, n): denotes the upgradation of the n^{th} already downloaded but yet to be played segment, by one. The value of n must be greater than a minimum threshold η_s. η_s is necessary to guaranty a minimum safety interval that can ensure that the n^{th} segment will never reach its playout instant while it is still in the process of being enhanced. We also define a boolean function $\delta(s_t)$ on each state. $\delta(s_t)$ is set to 1 only when the following two conditions simultaneously hold: (i) the playout buffer size q is greater than a system level global threshold q^{th} and (ii) atleast one segment in the buffer beyond the η_s^{th} segment is not currently at its highest enhancement level. The adaptation agent selects an action $a(\alpha_i, n) \in \mathcal{A}(s_t)$, where, $\mathcal{A}(s_t)$ is the set of available actions at state s_t and (α_i, n) can either assume values (α_d, n) or (α_s, n) as follows:

$$(\alpha_i, n) = \begin{cases} (\alpha_d, n), & \delta(s_t) = 0 \\ (\alpha_s, n), & \delta(s_t) = 1 \end{cases} \tag{6.6}$$

6.3.3 Transition Probability

This is defined as the probability of reaching state $s'(q')$ from the present state $s(q)$ in one step, by performing action a i.e.

$$\mathcal{P}_a(s, s') = \mathbf{Pr}(s_{t+1} = s' | s_t = s, A(s_t) = a) \tag{6.7}$$

For a given constant depletion rate of the playout buffer, a particular segment length/duration (say, τ) and expected instantaneous bandwidth corresponding to state $s(q)$, there is a distinct transition probability $\mathcal{P}_a(s, s')$ for each possible next state $s'(q')$ on action $a(\alpha_i, n)$. For example, if the action is $a(\alpha_s, 3)$ (which represents upgradation of the 3^{rd} segment in the buffer to its immediately next higher enhancement level), then the transition probabilities only corresponding to those next states whose buffer sizes (q') are lower than the current buffer size q, will be greater than zero. This is because, $a(\alpha_s, 3)$ denotes a smoothing action which cannot add any additional video segment into the playout buffer. The actual value of the transition probability will depend on whether the mentioned

enhancement level corresponding to the third segment can be downloaded within the duration $q - q'$. Assuming SZ_3 to be the amount of data that must be downloaded to effect this upgradation in enhancement level, the transition probability in turn can be expressed as the probability $\mathbf{Pr}(bw')$ that the received instantaneous bandwidth will be $bw' = SZ_3/(q - q')$. Thus, $\mathcal{P}_a(s, s') = \mathbf{Pr}(bw')$. The calculation of transition probability can be divided into two cases:

Case 1: When $\delta(s) = 1$, the transition probability for smoothing the n^{th} segment is given by:

$$\mathcal{P}_{a(\alpha_s,n)}\big(s(q), s'(q')\big) = \begin{cases} 0, & \text{if } q' \geq q \\ \mathbf{Pr}(\frac{SZ_n}{q-q'}), & \text{Otherwise} \end{cases} \tag{6.8}$$

where, SZ_n denotes the amount of data that is required to be downloaded to effect the smoothing action. Hence, $bw' = SZ_n/(q - q')$ represents the corresponding bandwidth required.

Case 2: When $\delta(s) = 0$, the transition probability for downloading the next segment at the n^{th} enhancement level is given by:

$$\mathcal{P}_{a(\alpha_d,n)}\big(s(q), s'(q')\big) = \begin{cases} 0, & \text{if } q' \geq (q + \tau) \\ \mathbf{Pr}(\frac{SZ_n}{q-q'+\tau}), & \text{Otherwise} \end{cases} \tag{6.9}$$

where, SZ_n denotes the amount of data required to download the next segment at the n^{th} enhancement level. $bw' = SZ_n/(q-q'+\tau)$ represents the corresponding bandwidth.

Bandwidth Estimation: From equations 6.8 and 6.9, it is clear that optimal rate adaptation through the correct assignment of transition probabilities is only possible by accurately estimating received bandwidths. However, accurate prediction of fast changing and short-term outages are difficult to predict and therefore, the resultant received network bandwidth for a video session becomes a time-varying random process. In this work, we have conducted bandwidth estimation using *Markov channel model* as it has been widely acknowledged as a useful tool to describe variations in cellular links under uncertainty [65]. According to the Markov property, the system bandwidth bw' at the $t + 1^{th}$ step

depends only on its bandwidth bw at the immediately previous step t. Hence, using the Markov channel model, $\mathbf{Pr}(bw')$ in equations 6.8 and 6.9 gets modified to $\mathbf{Pr}(bw'|bw)$.

The bandwidth is divided into several various distinct regions and each such region represents a state of the Markov channel model. Let N be the total number of states and P with cardinality $N \times N$ be transition matrix used for the Markov channel model. Each element $p_{ij} \in P$ denotes the transition probability from state i to j. In order to obtain the transition probability, another matrix C (with cardinality $N \times N$) is used to count the number of transitions for each state. Once a video segment is successfully downloaded, the received/transmission bandwidth can be calculated by dividing the total size of the video segment over the total download time. Then, count for the observed bandwidth region c_{ij} is incremented by one. p_{ij} is updated by the following equation

$$p_{ij} = \frac{c_{ij} + 1}{\sum_{j=1}^{N} + N} \tag{6.10}$$

Initially, if there is no history data available, $c_{ij} = 0$, and p_{ij} is set to $1/N$. The transition matrix will be updated after each segment has been successfully downloaded, so the transition matrix can better predict the future bandwidth variations with the recent measurements.

6.3.4 Reward Function

In *MDP*, the effectiveness of an action a which leads the system from state s to s' is measured through a parameter known as *reward* $\mathcal{R}_a(s, s')$. Since the objective of this work is to maximize *QoE*, the reward value associated with any action should reflect the change in overall *QoE* of the system as a result of that action. As discussed in section 6.3.2, this work assumes the overall *QoE* to be composed of three important parameters, namely, *encoding quality*, *switching* and *buffer size*. Correspondingly, the reward function $\mathcal{R}_a(s, s')$ has been designed as a linear

combination of three components, R_Q, R_{SW} and R_B. Thus,

$$\mathcal{R}_a(s, s') = R_Q + R_{SW} + R_B \tag{6.11}$$

Let, the action under consideration either downloads a new segment (the p^{th} segment) at encoding quality $Q_p(l)$ or smooths the q^{th} segment in the buffer from encoding quality $Q_q(l-1)$ to $Q_q(l)$. Then, R_Q denotes the reward component effected by the encoding quality of the newly downloaded or smoothed segment and is given by:

$$R_Q = \begin{cases} Q_p(l), & \delta(s) = 0 \\ Q_q(l) - Q_q(l-1), & \delta(s) = 1 \end{cases} \tag{6.12}$$

The component R_{SW} represents the reward share (penalty) resulting from the degree of encoding quality switching between consecutive segments due to the performed download/smoothing. R_{SW} is given by:

$$R_{SW} = \begin{cases} -\left|Q_p(l) - Q_{p-1}(l)\right|, & \delta(s_t) = 0 \\ -\{\left|Q_q(l) - Q_{q-1}(l)\right| + \\ \left|Q_q(l) - Q_{q+1}(l)\right|\}, & \delta(s_t) = 1 \end{cases} \tag{6.13}$$

Finally, R_B denotes the reward component due to the resultant buffer size q' in state s' after action a is performed. R_B is calculated as:

$$R_B = \begin{cases} -100, & : q' < q^{low} \\ q' - q^{th}, & : q^{th} > q' \geq q^{low} \\ 0, & : q' \geq q^{th} \end{cases} \tag{6.14}$$

Since, stutters in video playout (referred to as buffering events) caused due to buffer outages adversely effect overall *QoE*, a strong penalization (-100) has been applied in the above equation for the case in which the resultant buffer size q' is lower than a minimum allowable threshold q^{low}. If q' is above q^{low} but below a safe threshold q^{th}, R_B is assigned the negative reward component $q' - q^{th}$. $R_B = 0$ when q' is at least or above the safe threshold q^{th}.

6.3.5 MDP Solution Methodology and Algorithm

The solution to *MDP* is a policy π $(S \rightarrow A)$ which provide mappings of the action to be taken at each step. Given the reward function for each transition, the policy π has an expected value (long term discounted reward) for every state $V^{\pi}(s)$ which is computed as:

$$\begin{aligned} V^{\pi}(s) &= E_{\pi}\Big\{ \sum_{t=0}^{N_T} R_t | s_t = s \Big\} \\ &= \sum_{s'} \mathcal{P}_a(s,s')[\mathcal{R}_a(s,s') + \gamma V^{\pi}(s')] \end{aligned} \tag{6.15}$$

where, $\gamma \in [0,1]$ is a discount parameter reflecting the present value of a future reward. A small γ lets future rewards weigh less, and thus makes the adaptation decision more *myopic*.

Objective of a reinforcement learning based adaptation agent is to find the optimal streaming policy $\pi^*(s)$ which maximizes the long-term reward (or viewing experience) during video streaming.

6.3.6 The optimal policy

A policy is called optimal if it choses such actions at all steps starting from state s which maximizes the aggregate reward obtained. Therefore, the optimal policy can be obtained as:

$$\pi^*(s) = \underset{\pi}{\operatorname{argmax}} \sum_{s'} \mathcal{P}_a(s,s')[\mathcal{R}_a(s,s') + \gamma V^*(s')] \tag{6.16}$$

where, $V^*(s')$ is the best long-term reward for state s'.

Value Iteration [27] is a well known method for determining an optimal policy. It is an iterative algorithm that calculates an expected value of each state using rewards, transition probabilities and the values of next states, until the difference between the values calculated at two successive steps reduces below a small constant. The asymptotic worst-case time complexity of this algorithm is $\mathcal{O}(\mathcal{S}^2\mathcal{A})$ per iteration.

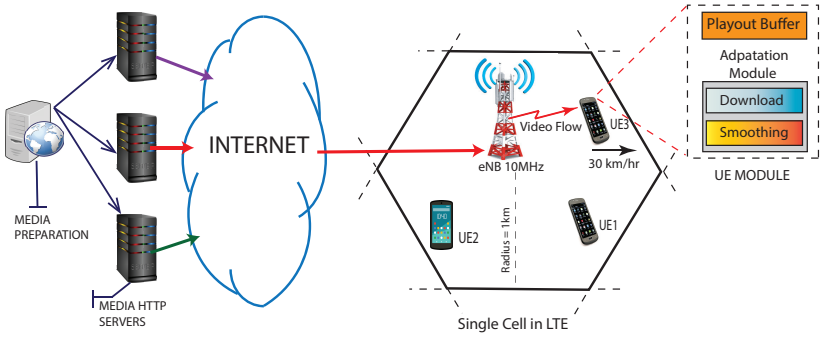


Figure 6.8: *Scenario for Experimental Setup in a Single Cell*

6.4 EXPERIMENTS AND RESULTS

The performance of the proposed strategies namely, *Video Quality Adaptation Unit (VQAU)* and *MDP based Adaptation Agent (MAA)* have been experimentally evaluated using *LTE-Sim* [34], an open source simulator for *LTE* networks. In the next subsection, we describe our detailed simulation setup and the important metrics used for performance evaluation of the proposed frameworks. Detailed experimental results along with a discussion on the same are presented in Section 6.4.2.

6.4.1 Simulation Setup

The overall network topology for the experimental setup has been pictorially depicted in Fig 6.8. As shown in this figure, we have implemented a single-cell environment using *LTE-Sim* where video flows hosted in the *DASH* server are requested by the various UEs/clients via *eNodeB*. *LTE-Sim* has allowed us to create a realistic single cell *4G* cellular network environment working in *Frequency Division Duplex (FDD)*. An UE/Client has been configured to receive a single *H.264/SVC* video flow. All our experiments have been carried out using three standard video sequences (namely, *Big Buck Bunny (BBB)* 720p, *Sony Demo* 720p and *Terminator* 720p) which are downloaded from [91]. The *SVC* trace

Table 6.1: *Simulation Parameters Used*

Number of Cells	1
Radius of the cell	1km
Number of video flows to each equipment	1
Frame Structure used	FDD
Speed of each UE	30km/hr
Video Bit rate of each flow	720p
Simulation time	500 secs
q_s^+ (Upper safe threshold for $VQAU$)	15 secs
q_s^- (Lower safe threshold for $VQAU$)	10 secs
q^{th} safe threshold for MAA	30 secs
Segment Duration	2 secs

files for these raw video sequences are generated using the *JSVM*[1] encoder. All simulations run for 500 secs. A summary of the main simulation parameters are presented in Table 6.1.

6.4.2 Results

We now present the detailed experimental results. In order to measure the effectiveness of the proposed adaptation strategies, namely *Video Quality Adaptation Unit (VQAU)* and *MDP based Adaptation Agent (MAA)*, we have compared it with two recent and important schemes, namely, *QoE-aware client-side Buffer Management Algorithm* [92] (Henceforth referred to as *QBMA*) and *Hybrid Adaptive Video Streaming* [58] (Henceforth referred to as *HAVS*).

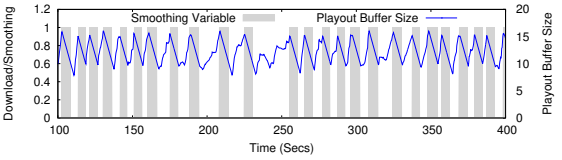
QBMA is a *MPEG-DASH* based streaming strategy which attempts to reduce playback interruptions by appropriately selecting the video quality of the next segment based on estimated bandwidth. Thus, selected video qualities are low when the client experiences bad network conditions. With the expectation that

[1][Online]. Available: https://www.hhi.fraunhofer.de/en/departments/vca/research-groups/image-video-coding/research-topics/svc-extension-of-h264avc/jsvm-reference-software.html

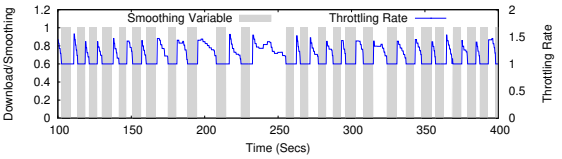
network conditions will improve in the near future (which will allow segments to be downloaded at better video qualities), *QBMA* caps maximum buffer size below a dynamic threshold in times of bad channel conditions. Lower the selected video quality at a given time, lower is the value of this dynamic threshold. *QBMA* pauses segment download when buffer size becomes higher than the dynamic threshold. The motivation behind this mechanism is to increase the chances of keeping the playout buffer filled with higher quality video segments. As an extreme measure to avoid buffer outages, *QBMA* downloads video segments at their lowest qualities in situations when the buffer size becomes critically low.

On the other hand, *HAVS* is a *SVC-DASH* based streaming methodology which essentially works with two distinct operational modes, namely, the progressive download mode for fetching only base layers of the video segments and the adaptive download mode for downloading enhancement layers at appropriate levels. When bandwidth conditions are poor, only the progressive download mode is enabled in the system. At this time, *HAVS* assumes the base layers of all the segments to be merged into a single file and stored on the server prior to streaming. The adaptation strategy therefore, downloads all the segments at base layer by sending a single *HTTP* request and stores them into a secondary storage device. On the other hand when the channel conditions are good, both modes are concurrently enabled. Multiple HTTP requests for desired enhancement layers are sent and subsequently fetched while performing progressive download in parallel. In order to perform adaptive download, *HAVS* defines two parameters: (i) the target segment index which should be upgraded (equal to the current playing segment index + 2) and (ii) the enhancement level of the target segment selected. This level is determined based on the instantaneously available bandwidth.

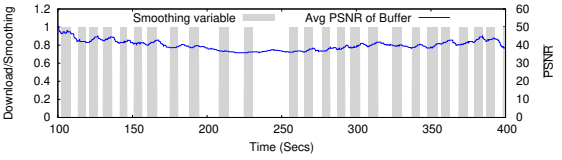
We have conducted two sets of experiments in order to analyze the efficacy of the proposed adaptation strategies.



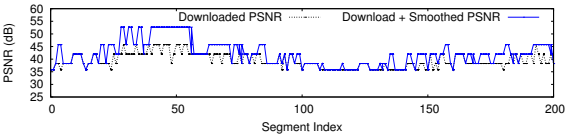
(a) Instantaneous playout buffer sizes Vs. Time



(b) Instantaneous throttling rates Vs. Time

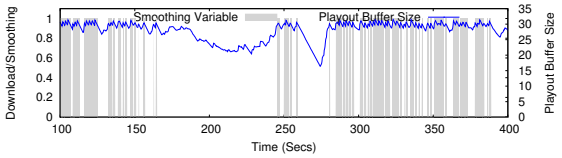


(c) Instantaneous $\overline{PSNR}$ of playout buffer Vs. Time

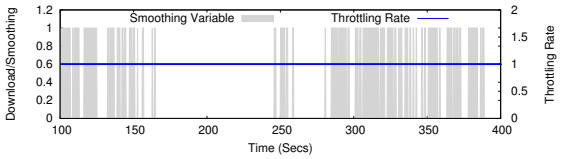


(d) Downloaded Vs. Played-back segments qualities

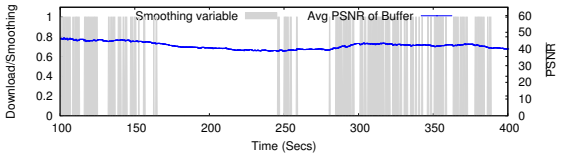
Figure 6.9: *Instantaneous Results for VQAU*



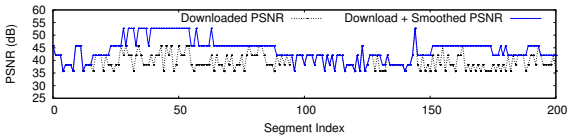
(a) Instantaneous playout buffer sizes Vs. Time



(b) Instantaneous throttling rates Vs. Time



(c) Instantaneous $\overline{PSNR}$ of playout buffer Vs. Time



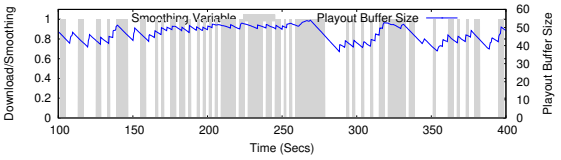
(d) Played-back segments qualities

Figure 6.10: *Instantaneous Results for MAA*

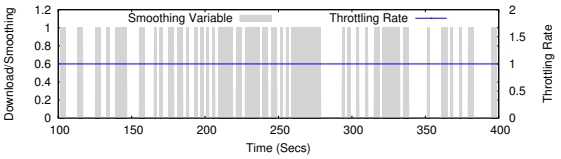
6.4.2.1 Experiment 1

In this experiment, we have measured the instantaneous variations of the three performance metrics (playout buffer size, switching instability, *PSNR*) for four strategies, namely, *VQAU*, *MAA*, *HAVS*, and *QBMA* over the entire simulation duration, corresponding to the video flow *Terminator*. Figures 6.9, 6.10, 6.11 and 6.12 depict the simulation results for *VQAU*, *MAA*, *HAVS*, and *QBMA* strategies, respectively.

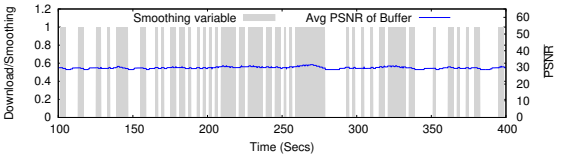
The grey color shade in Figures 6.9, 6.10 and 6.11 has been used to show the



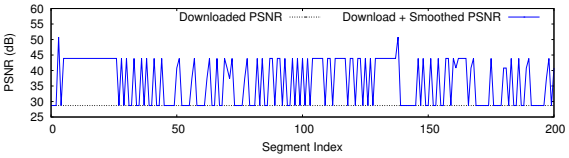
(a) Instantaneous playout buffer sizes Vs. Time



(b) Instantaneous throttling rates Vs. Time

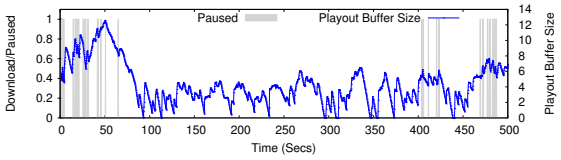


(c) Instantaneous $\overline{PSNR}$ of playout buffer Vs. Time

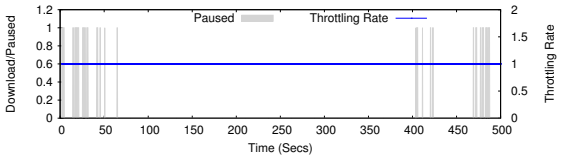


(d) Played-back segments qualities

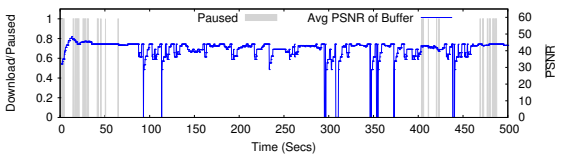
Figure 6.11: *Instantaneous Results for HAVS*



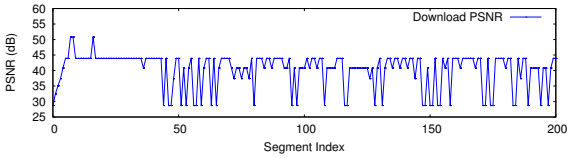
(a) Instantaneous playout buffer sizes Vs. Time



(b) Instantaneous throttling rates Vs. Time



(c) Instantaneous $\overline{PSNR}$ of playout buffer Vs. Time



(d) Played-back segments qualities

Figure 6.12: *Instantaneous Results for QBMA*

intervals in which the smoothing operation is enabled for *VQAU* (i.e. smoothing variable S is set to 1 by *DSS* module), *MAA* (i.e. boolean function $\delta(s_t) = 1$), *HAVS* (for this strategy smoothing is conducted whenever adaptive download mode is enabled). In Fig .6.12, the grey color shade depicts time durations when *QBMA* pauses segment downloads.

Comparing Figures 6.9(a), 6.10(a), 6.11(a) and 6.12(a), it may be observed that the *SVC-DASH* based strategies (i.e. *VQAU*, *MAA*, *HAVS*) are able to maintain stable playout buffer sizes during the entire simulation duration. On

the other hand, for *QBMA* (*MPEG-DASH* based strategy), buffer sizes may be seen to be poor on average with sizes rarely rising above 7 *secs*. Additionally, the buffer size frequently falls to zero resulting in outages leading to playback interruptions. The poor performance of *QBMA* may be attributed to its attempt to bound the maximum buffer size below a dynamic threshold to avoid low quality video downloads especially during long intervals with bad network conditions. Although, this mechanism helps to deliver relatively higher video qualities on average (it can be observed from Figure 6.12(c)), it also increases possibilities of buffer outages (which result in playback interruptions) when network conditions drastically degrade in an unpredictable manner.

As Figure 6.9(a) shows, *VQAU* is able to deliver such stable buffer sizes by initiating new segment downloads (done by setting S to 0, causing a switch from *smoothing* to *download* phase) whenever the buffer size falls below the lower threshold limit $q_s^- = 10$ *secs*. The *download* phase continues until buffer size reaches the upper threshold limit $q_s^+ = 15$ *secs*. In the figure, such a situation occurs at the 59^{th} second of the experiment when *VQAU* switches to *download* from *smoothing* because buffer size drops below q_s^- to 9.8 *secs*. This *download* phase continues up to the 68^{th} second of the experiment when the buffer size reaches 15.5 *secs* crossing q_s^+, and *VQAU* switches back to *smoothing*. During the download phase, *VQAU* attempts to conduct a buffer, bandwidth and switching aware quick buffer ramp-up using a combination of two principal mechanisms: (i) Buffer aware throttle rate adjustment by the *Throttle Rate Generator (TRG)* module and, (ii) Dynamic bandwidth and switching aware enhancement level selection for segment downloads given throttled bit-rate demands, by the *SELS* module (refer Figure 6.1). It may be seen form Figure 6.9(b), how the throttle rate control mechanism effectively hikes throttling rates based on instantaneous buffer sizes, during the download phase, to allow quick buffer replenishment. For a chosen segment transmission rate from the server (as decided by *TRG*), *SELS* determines the appropriate enhancement level for future segments such that download rates over wireless can match transmission rates from server while

satisfying switching bounds, given the expected channel capacity and throttled bit-rates. It is obvious that, to achieve desired throttled download rates over a limited bandwidth, the video qualities/enhancement levels of the segments being fetched must be degraded. This effect is indicated during all the download phases (for example, within the 59^{th} to the 68^{th} second of the experiment) in Figure 6.9(c) where the average video qualities ($\overline{PSNR}$) of segments in the buffer is seen to generally degrade. However, this degradation may not be exactly monotonic in all download phases as, (i) selected video qualities depend upon instantaneous bandwidths (ii) *SELS* gradually upgrades download enhancement levels as buffer size increases above q_S^-.

Similarly, it may be observed form Figure 6.10(a) that the *MDP* based adaptation agent is also able to achieve stable buffer sizes mainly due to the following two reasons. First, the *MAA* scheme is able to take better scheduling decisions by comprehensively considering future bandwidth variations in its choice of action at a given state. Second, the proposed strategy quickly ramps-up the buffer until a safe threshold buffer size (q^{th} =30 secs) is reached. On the other hand, *HAVS* fills the buffer with the base level of the segments using progressive download. Therefore, the number of downloaded segments available in the player is typically high.

During the smoothing phase, new segment downloads are cancelled, all the *SVC-DASH* based strategies (i.e., *VQAU*, *MAA*, and *HAVS*) delve into the process of enhancing a subset of buffered segments such that *QoE* is maximized. In Figures 6.9(c), 6.10(c) and 6.11(c), this facet is exhibited during smoothing phases where $\overline{PSNR}$ increases monotonically subsequent to the fetch of enhancement layers of already downloaded segments. The effectiveness of smoothing may be observed from Figures 6.9(d), 6.10(d) and 6.11(d), which shows the initial enhancement levels (during downloads) and the final played-back enhancement levels (subsequent to smoothing) for all segments of the video session considered in the experiment. Whereas, the average *PSNR* of the segments during initial download by the *VQAU*, *MAA*, and *HAVS* strategies are 39.15 dB, 40.29 dB

and 28.7 dB, respectively, the corresponding average played-back video qualities get enhanced to 41.82 dB, 42.68 dB and 36.49 dB after smoothing. It may be noted that, although, upgradation in the enhancement levels of the downloaded segments increases the quality of the video segment, such upgradation also adds switching instability in the video output. Therefore, the adaptation strategies must be aware of switching instability during the smoothing process in order to deliver smooth viewing experience to the end user. It may be noted that the *HAVS* strategy is oblivious of incurred switching instability. Therefore, it suffers form higher number of enhancement level switching. On the other hand, both the proposed adaptation strategies, namely $VQAU$ and MAA incorporate the effect of switching in the definition of their reward functions and hence, encounters a lower degree of switching.

Figure 6.12(c) shows that by displaying strong antipathy towards low quality video downloads through dynamic buffer threshold management, *QBMA* is able to generally deliver high $\overline{PSNR}$ values which are comparable and many a times slightly better than *SVC-DASH* strategies. However, the flip-side of this strong antipathy is that *QBMA* becomes susceptible to frequent buffer outages specially during fluctuating network conditions (as in case of fading channels in cellular networks) resulting in $\overline{PSNR}$ dropping to zero. It may be observed form Figure 6.12(d) that *QBMA* conducts segment downloads at good qualities on average, although frequently slipping down to the lowest quality during critically low buffer sizes. This not only brings down the average video qualities but also degrades overall QoE due to increased switching.

6.4.2.2 Experiment 2

A second set of experiments have been conducted to further evaluate and compare the performance of the strategies $VQAU$, MAA, *HAVS* and *QBMA* over three video sequences namely, *Big Buck Bunny (BBB)*, *Sony Demo* and *Terminator*. Table 6.2 shows comparative results for the total playback interruption due to *buffering* (in secs), mean instability (in #switching per sec) and mean video qual-

Table 6.2: *Comparative results for QoE*

	$\overline{PSNR}$				*Instability*				*Buffering*			
	VQAU	*MAA*	*HAVS*	*QBMA*	*VQAU*	*MAA*	*HAVS*	*QBMA*	*VQAU*	*MAA*	*HAVS*	*QBMA*
BBB	41.35	43.27	40.38	43.43	0.38	0.32	0.96	0.66	3.87	0.01	0.003	61.29
SD	37.00	37.30	31.53	36.99	0.34	0.26	1.61	1.22	5.24	0.04	0.003	115.7
Ter	41.82	42.68	36.49	40.41	0.12	0.10	1.04	0.43	0.003	0.018	0.003	32.62
Avg	40.05	41.08	36.13	40.27	0.28	0.22	1.20	0.77	3.03	0.02	0.003	69.87
	SD: Sony Demo; Ter: Terminator; Avg: Average											

ity (in dB) corresponding to these three video traces, over the entire simulation duration. The last row of this table presents the consolidated average over the different video traces for each of the parameters.

It may be observed from the last row of the table that the performance of *MAA* is slightly better than the *VQAU* strategy. Such better results are achieved by *MAA* due to its ability to take optimal decisions under uncertainty. On the other hand, as expected, the *HAVS* scheme is able to avoid re-buffering events by maintaining high buffer sizes throughout the simulation duration. However, being oblivious towards instability, the scheme encounters frequent bit-rate switches. The proposed strategies significantly outperforms *QBMA* in terms of buffer state stability and degree of switching. Played-back qualities of *QBMA* may be observed to be significantly higher than *HAVS* and comparable to schemes proposed in this chapter.

6.5 Summary

This chapter presents the design of two efficient *SVC-DASH* based video adaptation strategies called *VQAU* and *MAA*. The proposed adaptation strategies dynamically adjusts video bit-rates/enhancement levels based on perceived network environment with the objective of maximizing *QoE*. Both the strategies consist of two distinct phases, (i) a download phase which fetches new segments at selected quality levels until buffer size reaches a stipulated safety threshold and

(ii) a smooth-out phase which upgrades already downloaded segments with the objective of minimizing switching instability as well as improving overall video quality. The design of *VQAU* is based on a *DES* supported modelling scheme which makes it modular and flexible towards easy modification/reconfiguration in case of changes in design policy. On the other hand, the *MDP* based adaptation agent has the ability to take optimal decisions under uncertainty. Simulation results reveal that the proposed strategies is able to significantly restrict buffering events and bit-rate switching while delivering improved video quality to the end user.

Chapter 7

Conclusions and Future Perspectives

7.1 Summarization

In spite of progressively increasing data transmission capacities over wireless, effective allocation of radio resources to diverse latency sensitive applications is set to remain a daunting problem in *LTE* and other futuristic mobile networks. This is because, these systems impose at least two important functional/architectural challenges which any scheduling technique should effectively handle.

The first challenge relates to the significant temporal variations in both available radio network capacities as well as data rate demands of flows, in most practical scenarios. In *LTE*, such network capacity variations occur due to changes in *SINR* values corresponding to each *RB*-flow pair in every *TTI*. Given a stipulated transmission power, the instantaneous *SINR* (or *CQI*) for a *RB*-flow pair in turn, determines the amount of data (reflecting instantaneous capacity of the *RB*) that may be transmitted through the *RB*. Variations in data rate demands, especially in multimedia flows, typically occur due to structural changes in the underling multimedia sequence, particularly during scenarios like scene change. Although, many efficient strategies which attempt to model such data rate demand variations through stochastic prediction mechanisms are available in literature, accurate characterization of these latency sensitive multimedia flows is difficult to devise. It is perceivable that scheduling techniques which do not incorporate

efficient control mechanisms to provide satisfactory *QoS* in the face of these variations will be susceptible to heavy packet losses (PLR) and re-transmissions.

The second challenge relates to the high perception sensitivity and interactiveness of todays data intensive multimedia flows which consume a major part of the system bandwidth in current mobile networks. Delivering satisfactory *QoE* corresponding to these emerging multimedia applications such as streaming video depends on several key factors like video quality/bit-rate, the degree of buffering (causes playback interruptions) and frequency of bit-rate switches (results in flickering video outputs) etc. In order to provide a high quality of video viewing experience by harnessing limited and temporally varying available bandwidths, state-of-the-art multimedia streaming technologies allow dynamic quality adaptability in various dimensions [93]. However, design of scheduling strategies for multimedia must employ effective mechanisms which can utilize this power of dynamic adaptability to deliver the highest levels of transmission quality while avoiding playback interruptions and switching through careful management of client-side playout buffers at very short time scales.

This dissertation presents a few novel ideas towards the design of scheduling strategies for multimedia services over cellular networks. The strategies can be broadly categorized based on the types of flows being handled. Resource allocation strategies presented in chapters 3, 4 and 5 are in-network scheduling approaches while, the algorithms discussed in chapter 6 are client-side *SVC-DASH* based video streaming adaptation mechanisms. Chapter 3 deals with two scheduling strategies for generic real-time variable bit rate traffic over *LTE*. In chapter 4, we have adapted and enhanced the scheduling mechanisms designed in the previous chapter to specifically support non-adaptive video streaming. The problem and algorithms designed in chapters 3 and 4 were extended to handle adaptive video flows, in the fifth chapter. Then in chapter 6, our last contributory chapter, we proposed client-side adaptation strategies for *SVC-DASH* video streaming which attempt to deliver high quality video viewing experience to the end user even during temporally varying network conditions. We now present

brief summaries of these works in more detail.

In chapter 3, we presented two low overhead *LTE* downlink scheduling frameworks for *RT VBR* traffic. The frameworks are aimed at enabling mobile operators to effectively achieve good *QoS* and cell spectral efficiencies while incurring low overall scheduling overheads. A structured algorithm along with associated theoretical analyses have been presented to show how an efficient *RB* to flow mapping can be conducted for the *LTE* downlink channel taking into account their inherent temporal variability. Experiments conducted reveal that the proposed resource allocation frameworks are able to deliver high *QoE* along with high resource utilization over a variety of simulation scenarios.

Chapter 4, our second contributory chapter, enhanced the basic three level scheduling framework (presented in the third chapter) to support video streaming services. The modified framework is able deliver smooth video viewing experience to the end user by enabling client-side buffer awareness during the mapping of resource blocks to video flows. The resource allocation model has been formulated as an optimization problem and three solution strategies which are optimal, stochastic and heuristic (deterministic) in nature, have been proposed. The experimental results show that the proposed framework is able to minimize re-buffering events significantly as compared to the three level framework. For example, buffer-aware resource allocation strategies are able to reduce re-buffering events by up to 20% compared to a buffer unaware strategy when the total number of active video flows is equal to 50.

Chapter 5 deals with the design of an in-network adaptive video streaming framework for *LTE* systems. The proposed framework is able to achieve a judicious balance among important quality of experience parameters such as degree of re-buffering, encoding quality and stability against bit-rate switches. The chapter posed the problem of scheduling a set of adaptive video streams over *LTE* as an optimization problem and solved through a conventional *DP* strategy. However, theoretical analysis and the conducted experiments revealed that the *DP* based strategy poses significant overheads which makes it unsuitable for online appli-

cation. Then, a new optimal algorithm (called as *SDQA*) which memoizes only non-dominating intermediate partial solutions was designed to accelerate *DPQA*. Our experimental results showed that given a *LTE* bandwidth of 20 MHz in a system with 50 active video flows, conventional *DP* takes 14.4 ms (approximately 29 times the size of resource block) on average to generate a solution even for a moderate adaptation interval size of 1 sec on a 2.5 GHz computing core. In order to allow online employment, we have modified conventional *DP* and devised a new strategy known as *Streamlined DP-based Quality-level Allocator (SDQA)*. *SDQA* intelligently leverages the discrete nature in the data-rate scalability of video flows to retain a far lower number of non-dominating partial *DP*-solutions and allows the ultimate optimal solution to be generated much quicker. With the same experimental scenario, *SDQA* takes 1.4 ms on average to generate a solution. Further, we proposed a tunable approximation scheme called *SDQA-AA* that may be employed to accelerate the speed of generating solutions (or limit necessary computational resources) by various optional degrees with distinct bounds on the degradation in solution quality. *SDQA-AA* is able to generate solutions in about 0.067 ms for the same scenario while effecting less than 1% degradation in the solution quality.

In chapter 6, as our last contributory chapter, we have delved towards the design of efficient client-side schemes which adapt video bit-rates based on expected channel conditions and player status. In this chapter, we have presented two client-side *SVC-DASH* based adaptation strategies. The first strategy is a deterministic adaptation approach whose behavior is formally represented by a *Discrete Event System (DES)*. Then we have presented a *Markov Decision Process* based client-side video adaptation agent which dynamically adjusts bit-rate of the video segments based on perceived network environment. The principal objective of both the strategies is to maximize the quality of video viewing experience of an end user by dynamically taking one of the following two actions at any given state: (i) download a new segment at a selected enhancement level or, (ii) upgrade (smooth-out) the enhancement level of an already downloaded

segment in the playout buffer by a stipulated value. Simulation results revealed that the proposed strategies are able to significantly restrict buffering events and bit-rate switching while delivering improved video quality to the end user.

7.2 Future Works

The work presented in this thesis leaves several open directions and there is ample scope for future research in this area. In this section, we present five such future perspectives.

- **Application of effective heuristic search strategies for video streaming adaptation frameworks:** In chapter 5, we have formulated the resource allocation scheme as an optimization problem and presented two optimal strategies, namely, *DPQA* and *SDQA*. Although, both the strategies are optimal in nature, *SDQA* takes much lower time to obtain a solution compared to *DPQA*. A further reduction in the computational overhead of *SDQA* is possible by implementing effective pruning strategies which dynamically disregards those partial solutions whose further consideration is guaranteed to result failure in the search for the optimal solution.

- **QoE aware video streaming with limited battery and/or limited subscribed data:** User equipments in wireless networks consume large amounts of power while receiving, decoding and finally delivering multimedia flows to end users. However, battery technology has not evolved enough to cope with the exponentially growing power hungry multimedia services. Today, efforts are being made both at the hardware and software levels to make these systems more energy efficient. For example, given adaptive video flows, a client-side adaptation agent may be tuned to: (i) download a video stream such that a certain playback may be completed with a stipulated battery backup, (ii) plan time instants and durations of future video segments download such that the intervals in which the underline radio

frequency circuitry operates in lower power modes, is maximized [94].

A different but related problem arises in the case of a user who purposefully opts for lower bit-rate consumption so as to match his/her subscribed data plan. Even in this case, scheduling strategies have the potential to play a major role in respecting data rate of mobile devices/users.

- **Adaptation strategies for multi-view video streaming**: Multi-view video services over the cellular system are expected to be the next step in the evolution of digital video streaming technology. The immersive viewing experience produced by a multi-view video is obtained through the mechanism of constructive composition of multiple video streams, captured by a number of cameras at different positions, creating a number of views. The existence of such multiple views, however, makes the data content in video frames very high and this increases proportionally as the number of view-points increases. Therefore, although multi-view video is gaining popularity, its effective transmission over cellular networks poses an huge challenge on the radio resource allocation and management frameworks. In their endeavors to maintain minimum acceptable QoE to all end users even during transient network overloads, adaptive 3D multi-view video streaming is being widely viewed as a promising enabling technology. Again, effective scheduling technique must be design for high quality uninterrupted and smooth playback of multi-view videos.

- **Multimedia multicast/broadcast services:** Multicast technology allows each multimedia stream to be transmitted to a distinct group of users with possibly flexible QoS provisioning among users within each group. The Third Generation Partnership Project (3GPP) defined Multimedia Broadcast/Multicast Service (MBMS) in 2005 to optimize the distribution of multimedia traffic. This MBMS standard has evolved into enhanced MBMS (eMBMS) in 3GPP REl - 11 (June 2013) and builds on top of the 3GPP

LTE standard. We intend to propose multicast scheduling and resource allocation strategies for multimedia transmission over *LTE* in the near future.

- **Efficient resource allocation in 5G systems:** It is expected that 5G will bring unique network and service capabilities in order to ensure satisfactory user experience even in challenging situations such as high mobility (e.g. in trains), very dense or sparsely populated areas, and journeys covered by heterogeneous technologies. Such diverse requirements of next generation wireless networks can be satisfied through intelligent adaptive learning and decision making strategies.

Disseminations out of this Work

Book Chapter

1. **S. Kumar**, D. P. Goswami, A. Sarkar and A.Sur. "*A Buffer Aware Resource Allocation Framework for Video Streaming over LTE*". Communication Systems and Networks, pp. 223-242. Springer, Cham, 2017.

Journal Papers

1. **S. Kumar**, A. Sarkar, S. Sriram, A. Sur. "*A Three Level LTE Downlink Scheduling Framework for RT VBR Traffic*". Computer Networks, Elsevier, vol 91, pp. 654-674, 2015.

2. **S. Kumar**, R. Devaraj, A. Sarkar. "*A Hybrid Offline-Online Approach to Adaptive Downlink Resource Allocation Over LTE*". IEEE/CAA Journal of Automatica Sinica, June, 2017 (Accepted).

3. **S. Kumar**, A. Sarkar, A. Sur. "*A Resource Allocation Framework for Adaptive Video Streaming Over LTE*". Accepted in: Journal of Network and Computer Applications, Elsevier, Vol. 97, pp. pp. 126-139, 2017.

4. **S. Kumar**, R. Devaraj, A. Sarkar, A. Sur, S. Biswas. "*Client-side QoE Control for SVC-DASH Video Streaming: A DES Supported Design Approach*". IEEE Transactions on Network and Service Management. (Under review).

5. **S. Kumar**, A. Sarkar, A. Sur. "*An Efficient QoE Aware MDP Based Client-side Agent for Video Adaptation in Cellular Networks*". IEEE Transactions on Consumer Electronics. (Under review).

Conference Paper

1. **S. Kumar**, S. Sriram, A. Sarkar, A. Sur. "*A Three Level Adaptive Video Streaming Framework Over LTE*". IEEE International Conference on Systems, Man, and Cybernetics, Budapest Hungary, Oct, 2016.

2. **S. Kumar**, D. Goswami, A. Sarkar, A. Sur. "*Buffer Aware Three Level Scheduler for Video Streaming over LTE*". IEEE International Conference on COMmunication Systems & NETworkS, Jan, 2017.

3. A. Lekharu, **S. Kumar**, A. Sur and A. Sarkar. "*A QoE Aware SVC Based Client-side Video Adaptation Algorithm for Cellular Networks*". In Proceedings of the 19th International Conference on Distributed Computing and Networking (p. 27). ACM 2018.

4. A. Lekharu, **S. Kumar**, A. Sur, A. Sarkar. "*A QoE Aware LSTM based Bit-Rate Prediction Model for DASH Video*". International Conference on COMmunication Systems & NETworkS (COMSNETS), Jan, 2018. (Accepted).

References

[1] G. Van der Auwera, P. David, and M. Reisslein, "Traffic and quality characterization of single-layer video streams encoded with the H. 264/MPEG-4 advanced video coding standard and scalable video coding extension," *IEEE Transactions on Broadcasting*, vol. 54, no. 3, pp. 698–718, 2008. [Pg.xxiii], [Pg.55], [Pg.61], [Pg.129]

[2] C. V. N. Index, "Global Mobile Data Traffic Forecast Update, 2016–2021," White Paper, March 28, 2017. [Pg.1], [Pg.32]

[3] D. Astély, E. Dahlman, A. Furuskär, Y. Jading, M. Lindström, and S. Parkvall, "LTE: the evolution of mobile broadband," *IEEE Communications magazine*, vol. 47, no. 4, 2009. [Pg.1]

[4] H. Jiang and W. Zhuang, "Realtime service provisioning in CDMA wireless cellular networks," in *Global Telecommunications Conference, 2005. GLOBECOM '05. IEEE*, vol. 5, 2005, pp. 5 pp.–2636. [Pg.4]

[5] G. Piro, L. Grieco, G. Boggia, and P. Camarda, "A two-level scheduling algorithm for QoS support in the downlink of LTE cellular networks," in *Wireless Conference (EW), 2010 European*, 2010, pp. 246–253. [Pg.4]

[6] P. Kela, J. Puttonen, N. Kolehmainen, T. Ristaniemi, T. Henttonen, and M. Moisio, "Dynamic packet scheduling performance in UTRA Long Term Evolution downlink," in *Wireless Pervasive Computing, 2008. ISWPC 2008. 3rd International Symposium on*, 2008, pp. 308–313. [Pg.4]

[7] V. Vukadinovic and G. Karlsson, "Video streaming performance under proportional fair scheduling," *Selected Areas in Communications, IEEE Journal on*, vol. 28, no. 3, pp. 399–408, 2010. [Pg.4]

[8] Y. Fan, P. Lunden, M. Kuusela, and M. Valkama, "Efficient Semi-Persistent Scheduling for VoIP on EUTRA Downlink," in *Vehicular Technology Conference, 2008. VTC 2008-Fall. IEEE 68th*, 2008, pp. 1–5. [Pg.4]

[9] V. Vukadinovic and G. Karlsson, "Video streaming performance under proportional fair scheduling," *Selected Areas in Communications, IEEE Journal on*, vol. 28, no. 3, pp. 399–408, 2010. [Pg.4], [Pg.30], [Pg.47], [Pg.48]

[10] F. Capozzi, G. Piro, L. A. Grieco, G. Boggia, and P. Camarda, "Downlink packet scheduling in lte cellular networks: Key design issues and a survey," *Communications Surveys & Tutorials, IEEE*, vol. 15, no. 2, pp. 678–700, 2013. [Pg.4], [Pg.25], [Pg.29], [Pg.30], [Pg.65]

[11] B. Sadiq, S. J. Baek, and G. De Veciana, "Delay-optimal opportunistic scheduling and approximations: The LOG rule," *IEEE/ACM Transactions on Networking (TON)*, vol. 19, no. 2, pp. 405–418, 2011. [Pg.4], [Pg.10], [Pg.30], [Pg.61]

[12] S. Shakkottai and A. L. Stolyar, "Scheduling for Multiple Flows Sharing a Time-Varying Channel: The Exponential Rule," *American Mathematical Society Translations, Series*, vol. 2, p. 2002, 2000. [Pg.4], [Pg.10], [Pg.30], [Pg.61]

[13] M. Iturralde, A. Wei, T. Ali Yahiya, and A.-L. Beylot, "Resource allocation for real time services using cooperative game theory and a virtual token mechanism in lte networks," in *Consumer Communications and Networking Conference (CCNC), 2012 IEEE*. IEEE, 2012, pp. 879–883. [Pg.4], [Pg.10], [Pg.30], [Pg.31], [Pg.61]

[14] G. Piro, L. A. Grieco, G. Boggia, R. Fortuna, and P. Camarda, "Two-level downlink scheduling for real-time multimedia services in LTE networks," *Multimedia, IEEE Transactions on*, vol. 13, no. 5, pp. 1052–1065, 2011. [Pg.4], [Pg.10], [Pg.31], [Pg.61]

[15] S. Petrangeli, J. Famaey, M. Claeys, S. Latré, and F. D. Turck, "QoE-driven Rate Adaptation Heuristic for Fair Adaptive Video Streaming," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 12, no. 2, p. 28, 2015. [Pg.4]

[16] K. Miller, D. Bethanabhotla, G. Caire, and A. Wolisz, "A control-theoretic approach to adaptive video streaming in dense wireless networks," *IEEE Transactions on Multimedia*, vol. 17, no. 8, pp. 1309–1322, 2015. [Pg.4]

[17] J. Chen, R. Mahindra, M. A. Khojastepour, S. Rangarajan, and M. Chiang, "A scheduling framework for adaptive video delivery over cellular networks," in *Proceedings of the 19th annual international conference on Mobile computing & networking*. ACM, 2013, pp. 389–400. [Pg.4], [Pg.34], [Pg.128], [Pg.130]

[18] 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. [Pg.5]

[19] M. H. Hajiesmaili, A. Khonsari, A. Sehati, and M. S. Talebi, "Content-aware rate allocation for efficient video streaming via dynamic network utility maximization," *Journal of Network and Computer Applications*, vol. 35, no. 6, pp. 2016–2027, 2012. [Pg.5]

[20] K. Ivesic, L. Skorin-Kapov, and M. Matijasevic, "Cross-layer QoE-driven admission control and resource allocation for adaptive multimedia services in LTE," *Journal of Network and Computer Applications*, vol. 46, pp. 336–351, 2014. [Pg.5]

[21] H. Riiser, H. S. Bergsaker, P. Vigmostad, P. Halvorsen, and C. Griwodz, "A comparison of quality scheduling in commercial adaptive HTTP streaming solutions on a 3G network," in *Proceedings of the 4th Workshop on Mobile Video*. ACM, 2012, pp. 25–30. [Pg.5], [Pg.36]

[22] S. Avocanh, J. Thierry, M. Abdennebi, and J. Ben-Othman, "A new two-level scheduling algorithm for the downlink of LTE networks," in *Globecom*

Workshops (GC Wkshps), 2013 IEEE. IEEE, 2013, pp. 4519–4523. [Pg.10], [Pg.29], [Pg.61]

[23] C. Müller, S. Lederer, and C. Timmerer, "An evaluation of dynamic adaptive streaming over HTTP in vehicular environments," in *Proceedings of the 4th Workshop on Mobile Video*. ACM, 2012, pp. 37–42. [Pg.12], [Pg.36], [Pg.92]

[24] D. E. Golberg, "Genetic algorithms in search, optimization, and machine learning," *Addion wesley*, vol. 1989, p. 102, 1989. [Pg.13], [Pg.98], [Pg.100]

[25] W. U. Rahman, D. Yun, and K. Chung, "A client side buffer management algorithm to improve QoE," *IEEE Transactions on Consumer Electronics*, vol. 62, no. 4, pp. 371–379, 2016. [Pg.16], [Pg.36]

[26] S. Kim, D. Yun, and K. Chung, "Video quality adaptation scheme for improving QoE in HTTP adaptive streaming," in *Information Networking (ICOIN), 2016 International Conference on*. IEEE, 2016, pp. 201–205. [Pg.16]

[27] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014. [Pg.16], [Pg.140], [Pg.153], [Pg.159]

[28] R. Nossenson, "Long-term evolution network architecture," in *Microwaves, Communications, Antennas and Electronics Systems, 2009. COMCAS 2009. IEEE International Conference on*. IEEE, 2009, pp. 1–4. [Pg.22]

[29] J. Zyren and W. McCoy, "Overview of the 3GPP long term evolution physical layer," *Freescale Semiconductor, Inc., white paper*, 2007. [Pg.22]

[30] H. Holma and A. Toskala, *LTE for UMTS-OFDMA and SC-FDMA based radio access*. John Wiley & Sons, 2009. [Pg.23]

[31] A. M. Maia, D. Vieira, M. F. de Castro, and Y. Ghamri-Doudane, "A mechanism for uplink packet scheduler in LTE network in the context of machine-to-machine communication," in *Global Communications Conference (GLOBECOM), 2014 IEEE*. IEEE, 2014, pp. 2776–2782. [Pg.23]

[32] L. Á. M. R. De Temiño, G. Berardinelli, S. Frattasi, and P. Mogensen, "Channel-aware scheduling algorithms for SC-FDMA in LTE uplink," in *Personal, Indoor and Mobile Radio Communications, 2008. PIMRC 2008. IEEE 19th International Symposium on.* IEEE, 2008, pp. 1–6. [Pg.23]

[33] J. Zyren and W. McCoy, "Overview of the 3GPP long term evolution physical layer," *Freescale Semiconductor, Inc., white paper*, 2007. [Pg.24], [Pg.25]

[34] G. Piro, L. A. Grieco, G. Boggia, F. Capozzi, and P. Camarda, "Simulating LTE cellular systems: an open-source framework," *Vehicular Technology, IEEE Transactions on*, vol. 60, no. 2, pp. 498–513, 2011. [Pg.24], [Pg.61], [Pg.104], [Pg.128], [Pg.160]

[35] C. Cox, *An introduction to LTE: LTE, LTE-advanced, SAE and 4G mobile communications.* John Wiley & Sons, 2012. [Pg.25]

[36] C. Mehlführer, M. Wrulich, J. C. Ikuno, D. Bosanska, and M. Rupp, "Simulating the long term evolution physical layer," in *Proc. of the 17th European Signal Processing Conference (EUSIPCO 2009), Glasgow, Scotland*, vol. 27. Citeseer, 2009, p. 124. [Pg.25]

[37] 3GPP, "Tech. Specif. Group Radio Access Network; Conveying MCS and TB Size via PDCCH," *3GPP TSG-RAN WG1 R1-081483.* [Pg.25]

[38] V. Vukadinovic and G. Karlsson, "Video streaming performance under proportional fair scheduling," *IEEE Journal on Selected Areas in Communications*, vol. 28, no. 3, 2010. [Pg.29]

[39] P. Svedman, S. K. Wilson, and B. Ottersten, "A QoS-aware proportional fair scheduler for opportunistic OFDM," in *Vehicular Technology Conference, 2004. VTC2004-Fall. 2004 IEEE 60th*, vol. 1. IEEE, 2004, pp. 558–562. [Pg.29]

[40] A. Pokhariyal, K. I. Pedersen, G. Monghal, I. Z. Kovacs, C. Rosa, T. E. Kolding, and P. E. Mogensen, "HARQ aware frequency domain packet scheduler

with different degrees of fairness for the UTRAN long term evolution," in *Vehicular Technology Conference, 2007. VTC2007-Spring. IEEE 65th.* IEEE, 2007, pp. 2761–2765. [Pg.29]

[41] J. Park, S. Hwang, and H.-S. Cho, "A packet scheduling scheme to support real-time traffic in OFDMA systems," in *Vehicular Technology Conference, 2007. VTC2007-Spring. IEEE 65th.* IEEE, 2007, pp. 2766–2770. [Pg.29]

[42] X. Yang and G. Wei, "Adaptive resource allocation techniques in OFDMA using cross-layer architecture," *Journal of Electronics (China)*, vol. 22, no. 6, pp. 586–593, 2005. [Pg.29], [Pg.30]

[43] H. A. M. Ramli, R. Basukala, K. Sandrasegaran, and R. Patachaianand, "Performance of well known packet scheduling algorithms in the downlink 3GPP LTE system," in *Communications (MICC), 2009 IEEE 9th Malaysia International Conference on.* IEEE, 2009, pp. 815–820. [Pg.29]

[44] Q. Wang, H.-L. Liu, Z.-h. Li, Y.-m. Cheung, and J. Zhang, "An Evolutionary Algorithm for TD-LTE Resource Allocation Based on Adaptive Fairness Threshold," in *Simulated Evolution and Learning.* Springer, 2014, pp. 713–722. [Pg.30]

[45] S. Shakkottai and A. L. Stolyar, "Scheduling algorithms for a mixture of real-time and non-real-time data in HDR," *Teletraffic Science and Engineering*, vol. 4, pp. 793–804, 2001. [Pg.30]

[46] G. Monghal, K. I. Pedersen, I. Z. Kovacs, and P. E. Mogensen, "QoS oriented time and frequency domain packet schedulers for the UTRAN long term evolution," in *Vehicular Technology Conference, 2008. VTC Spring 2008. IEEE.* IEEE, 2008, pp. 2532–2536. [Pg.31]

[47] "Multi-QoS-aware fair scheduling for LTE, author=Zaki, Yasir and Weerawardane, Thushara and Gorg, C and Timm-Giel, Andreas," in *Vehicular technology conference (VTC spring), 2011 IEEE 73rd.* IEEE, 2011, pp. 1–5. [Pg.31]

[48] T. Stockhammer, "Dynamic adaptive streaming over HTTP–: standards and design principles," in *Proceedings of the second annual ACM conference on Multimedia systems*. ACM, 2011, pp. 133–144. [Pg.33]

[49] A. El Essaili, D. Schroeder, E. Steinbach, D. Staehle, and M. Shehada, "QoE-based traffic and resource management for adaptive HTTP video delivery in LTE," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, no. 6, pp. 988–1001, 2015. [Pg.34], [Pg.35], [Pg.114]

[50] R. Radhakrishnan and A. Nayak, "Cross layer design for efficient video streaming over LTE using scalable video coding," in *Communications (ICC), 2012 IEEE International Conference on*. IEEE, 2012, pp. 6509–6513. [Pg.34]

[51] J. Yang, Y. Ran, S. Chen, W. Li, and L. Hanzo, "Online source rate control for adaptive video streaming over SHPA and LTE-style variable bit rate downlink channels," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 2, pp. 643–657, 2016. [Pg.34]

[52] A. Ahmedin, K. Pandit, D. Ghosal, and A. Ghosh, "Content and buffer aware scheduling for video delivery over LTE," in *Proceedings of the 2013 workshop on Student workhop*. ACM, 2013, pp. 43–46. [Pg.34]

[53] S. Cicalo, N. Changuel, R. Miller, B. Sayadi, and V. Tralli, "Quality-fair HTTP adaptive streaming over LTE network," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 714–718. [Pg.34]

[54] V. Ramamurthi and O. Oyman, "Video-QoE aware radio resource allocation for HTTP adaptive streaming," in *Communications (ICC), 2014 IEEE International Conference on*. IEEE, 2014, pp. 1076–1081. [Pg.34], [Pg.35]

[55] Z. Li, X. Zhu, J. Gahm, R. Pan, H. Hu, A. C. Begen, and D. Oran, "Probe and adapt: Rate adaptation for HTTP video streaming at scale," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 4, pp. 719–733, 2014. [Pg.36]

[56] M. Seufert, S. Egger, M. Slanina, T. Zinner, T. Hobfeld, and P. Tran-Gia, "A survey on quality of experience of HTTP adaptive streaming," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 1, pp. 469–492, 2015. [Pg.36]

[57] T. Andelin, V. Chetty, D. Harbaugh, S. Warnick, and D. Zappala, "Quality selection for dynamic adaptive streaming over HTTP with scalable video coding," in *Proceedings of the 3rd Multimedia Systems Conference*. ACM, 2012, pp. 149–154. [Pg.36]

[58] J. Hwang, J. Lee, N. Choi, and C. Yoo, "HAVS: Hybrid adaptive video streaming for mobile devices," *IEEE Transactions on Consumer Electronics*, vol. 60, no. 2, pp. 210–216, 2014. [Pg.36], [Pg.161]

[59] 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*. ACM, 2011, pp. 145–156. [Pg.36]

[60] L. De Cicco and S. Mascolo, "An adaptive video streaming control system: Modeling, validation, and performance evaluation," *IEEE/ACM Transactions on Networking (TON)*, vol. 22, no. 2, pp. 526–539, 2014. [Pg.36]

[61] L. De Cicco, G. Cofano, and S. Mascolo, "A hybrid model of the Akamai adaptive streaming control system," *Nonlinear Analysis: Hybrid Systems*, vol. 21, pp. 139–154, 2016. [Pg.36]

[62] M. S. Ito, D. Bezerra, S. Fernandes, D. Sadok, and G. Szabo, "A fine-tuned control-theoretic approach for dynamic adaptive streaming over http," in *Computers and Communication (ISCC), 2015 IEEE Symposium on*. IEEE, 2015, pp. 301–308. [Pg.37]

[63] D. Bezerra, M. Ito, W. Melo, D. Sadok, and J. Kelner, "DBuffer: A state machine oriented control system for DASH," in *Computers and Communication (ISCC), 2016 IEEE Symposium on*. IEEE, 2016, pp. 861–867. [Pg.37]

[64] T.-Y. Huang, R. Johari, and N. McKeown, "Downton abbey without the hiccups: Buffer-based rate adaptation for http video streaming," in *Proceedings*

of the 2013 ACM SIGCOMM workshop on Future human-centric multimedia networking. ACM, 2013, pp. 9–14. [Pg.37]

[65] M. Xing, S. Xiang, and L. Cai, "A real-time adaptive algorithm for video streaming over multiple wireless access networks," *IEEE Journal on Selected Areas in communications*, vol. 32, no. 4, pp. 795–805, 2014. [Pg.37], [Pg.156]

[66] C. Zhou, C.-W. Lin, and Z. Guo, "mDASH: A markov decision-based rate adaptation approach for dynamic HTTP streaming," *IEEE Transactions on Multimedia*, vol. 18, no. 4, pp. 738–751, 2016. [Pg.37]

[67] C. Zhou and C.-W. Lin, "A Markov decision based rate adaption approach for dynamic HTTP streaming," in *Visual Communications and Image Processing (VCIP), 2015*. IEEE, 2015, pp. 1–4. [Pg.37]

[68] A. Bokani, M. Hassan, and S. Kanhere, "HTTP-based adaptive streaming for mobile clients using markov decision process," in *Packet Video Workshop (PV), 2013 20th International*. IEEE, 2013, pp. 1–8. [Pg.37]

[69] P. De Cuetos and K. W. Ross, "Optimal streaming of layered video: joint scheduling and error concealment," in *Proceedings of the eleventh ACM international conference on Multimedia*. ACM, 2003, pp. 55–64. [Pg.37]

[70] M. Koppen, K. Ohnishi, and M. Tsuru, "Multi-jain fairness index of per-entity allocation features for fair and efficient allocation of network resources," in *Intelligent Networking and Collaborative Systems (INCoS), 2013 5th International Conference on*. IEEE, 2013, pp. 841–846. [Pg.38]

[71] G.-M. Su, Z. Han, M. Wu, and K. R. Liu, "Joint uplink and downlink optimization for real-time multiuser video streaming over wlans," *IEEE Journal of Selected Topics in Signal Processing*, vol. 1, no. 2, pp. 280–294, 2007. [Pg.41], [Pg.46]

[72] L. Guo, E. Tan, S. Chen, Z. Xiao, O. Spatscheck, and X. Zhang, "Delving into internet streaming media delivery: a quality and resource utilization perspective," in *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*. ACM, 2006, pp. 217–230. [Pg.52]

[73] S. Kang, S. Lee, Y. Won, and B. Seong, "On-line prediction of nonstationary variable-bit-rate video traffic," *Signal Processing, IEEE Transactions on*, vol. 58, no. 3, pp. 1219–1237, 2010. [Pg.52]

[74] H. Liu and G. Mao, "Prediction algorithms for real-time variable-bit-rate video," in *Communications, 2005 Asia-Pacific Conference on.* IEEE, 2005, pp. 664–668. [Pg.52]

[75] C. C. Holt, "Forecasting seasonals and trends by exponentially weighted moving averages," *International Journal of Forecasting*, vol. 20, no. 1, pp. 5–10, 2004. [Pg.53]

[76] D. Trigg and A. Leach, "Exponential smoothing with an adaptive response rate," *OR*, pp. 53–59, 1967. [Pg.56]

[77] C. Bettstetter, "Mobility modeling in wireless networks: categorization, smooth movement, and border effects," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 5, no. 3, pp. 55–66, 2001. [Pg.61], [Pg.129]

[78] J. Rice, *Mathematical statistics and data analysis.* Cengage Learning, 2006. [Pg.63]

[79] M. M. S. Pasand and M. Montazeri, "Structural properties, LQG control and scheduling of a networked control system with bandwidth limitations and transmission delays," *IEEE/CAA Journal of Automatica Sinica*, 2017. [Pg.76]

[80] M. Danancher, J.-J. Lesage, L. Litz, and G. Faraut, "Online location tracking of a single inhabitant based on a state estimator," in *Systems, Man, and Cybernetics (SMC), 2013 IEEE International Conference on.* IEEE, 2013, pp. 391–396. [Pg.84]

[81] ——, "A discrete event model for multiple inhabitants location tracking," in *Automation Science and Engineering (CASE), 2013 IEEE International Conference on.* IEEE, 2013, pp. 910–915. [Pg.84]

[82] "Conviva: Think Streaming Will Replace Cable TV." [Online]. Available: http://www.conviva.com/think-streaming-will-replace-cable-tv-this-data-on-streaming-quality-proves-otherwise/ [Pg.92]

[83] P. Sinha and A. A. Zoltners, "The multiple-choice knapsack problem," *Operations Research*, vol. 27, no. 3, pp. 503–515, 1979. [Pg.96]

[84] X. Cheng and P. Mohapatra, "Quality-optimized downlink scheduling for video streaming applications in lte networks," in *Global Communications Conference (GLOBECOM), 2012 IEEE*. IEEE, 2012, pp. 1914–1919. [Pg.98]

[85] Y. Kaya, M. Uyar *et al.*, "A novel crossover operator for genetic algorithms: ring crossover," *arXiv preprint arXiv:1105.0355*, 2011. [Pg.100]

[86] M. A. Hoque, M. Siekkinen, J. K. Nurminen, M. Aalto, and S. Tarkoma, "Mobile Multimedia Streaming Techniques: QoE and Energy Consumption Perspective," *arXiv preprint arXiv:1311.4317*, 2013. [Pg.115], [Pg.145]

[87] V. Q. E. Group *et al.*, "Final Report on the validation of objective models of video quality assessment," 2003. [Pg.119]

[88] "ITU-R (2002) Methodology for the subjective assessment of the quality of television pictures," *Int Telecommun Union*, BT-500, 11. [Pg.126]

[89] V. Joseph and G. de Veciana, "NOVA: QoE-driven optimization of DASH-based video delivery in networks," in *INFOCOM, 2014 Proceedings IEEE*. IEEE, 2014, pp. 82–90. [Pg.148]

[90] M. Zhao, X. Gong, J. Liang, W. Wang, X. Que, Y. Guo, and S. Cheng, "QoE-driven optimization for cloud-assisted DASH-based scalable interactive multiview video streaming over wireless network," *Signal Processing: Image Communication*, vol. 57, pp. 157–172, 2017. [Pg.148]

[91] A. S. University, "Yuv video sequences," [Online]. Available: http://trace.eas.asu.edu/yuv/. Last Accessed in May, 2017. [Pg.160]

[92] W. U. Rahman, D. Yun, and K. Chung, "A client side buffer management algorithm to improve qoe," *IEEE Transactions on Consumer Electronics*, vol. 62, no. 4, pp. 371–379, 2016. [Pg.161]

[93] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the scalable video coding extension of the H.264/AVC standard," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 17, no. 9, pp. 1103–1120, 2007. [Pg.172]

[94] M. S. Mushtaq, A. Mellouk, B. Augustin, and S. Fowler, "QoE power-efficient multimedia delivery method for LTE-A," *IEEE Systems Journal*, vol. 10, no. 2, pp. 749–760, 2016. [Pg.176]