ADAPTIVE WIRELESS VIDEO STREAMING AND TELECONFERENCING

by

Wei Chen

A dissertation submitted to the Faculty of the University of Delaware in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computer Science

Winter 2017

ADAPTIVE WIRELESS VIDEO STREAMING AND TELECONFERENCING

by

Wei Chen

Approved: _____
Kathleen F. McCoy, Ph.D.
Chair of the Department of Computer and Information Sciences

Approved: _____
Babatunde Ogunnaike, Ph.D.
Dean of the College of Engineering

Approved: _____
Ann L. Ardis, Ph.D.
Senior Vice Provost for Graduate and Professional Education

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: _____

Chien-Chung Shen, Ph.D.
Professor in charge of dissertation

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: _____

Adarshpal S. Sethi, Ph.D.
Member of dissertation committee

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: _____

Christopher Rasmussen, Ph.D.
Member of dissertation committee

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: _____

Liangping Ma, Ph.D.
Member of dissertation committee

To my parents who brought me to this world.

To my wife and our son who are my love and my inspiration.

ACKNOWLEDGEMENTS

Foremost, I would like to express my sincere appreciation to thank my advisor, Dr. Chien-Chung Shen, for his continuous support, encouragement and inspirations. Over the years throughout my Ph.D. studies, Dr. Shen has been providing so many resources and help to accommodate my research and study. Dr. Shen keeps stimulating me to explore new research ideas, brainstorming and discussing research issues with me while patiently showing me the methodology to do research. His enthusiasm for research and his dedication to work have deeply influenced me since my first day at the University of Delaware.

Next, I would like to thank my external committee member Dr. Liangping Ma, for his invaluable inspiration, guidance and continuous encouragement. During the projects working with Dr. Ma, his deep knowledge and insight to the research helped me tremendously in developing precious research skills in approaching and analyzing complicated problems.

I am also grateful to my other two defense committee members, Dr. Adarsh Sethi, Dr. Christopher Rasmussen. Dr. Adarsh Sethi helped me with proposal and thesis writing problems and provided precious suggestions. Dr. Christopher Rasmussen made valuable comments on my proposal and dissertation.

I feel very fortunate to work with a group of talented research colleagues, Dr. Ke Li, Dr. Yang Guan, Dr. Zequn Huang, and Hao Luan. We have been sharing knowledge together and helping each other with research projects. It was my privilege of working with all of them. I also greatly appreciate the assistance of our department staff, Teresa Louise, Krystal Proctor, Samantha Fowle and T. Gregory Lynch, throughout my study at the University of Delaware.

I want to thank my parents for making many sacrifices that allowed me to get to this point. They have respected every decision I made, been happy for every little thing

I achieved and supported me in every way they could. I could not have gone through the process without the support from my parents.

Finally, I want to thank my wife Sha and my son Ivan. I thank you for being here with me, for sharing my happiness and sadness, for being proud of every achievement I have made and for all the sacrifices you made for our family. Your love will always be the motivation of my work and life.

TABLE OF CONTENTS

LIST OF TABLES . . . . . **xii**
LIST OF FIGURES . . . . . **xiii**
ABSTRACT . . . . . **xvii**

Chapter

1 INTRODUCTION **1**

1.1 Background and Overall Motivations 1

1.1.1 Video Streaming vs. Video Teleconferencing 2
1.1.2 QoS vs. QoE 2
1.1.3 Effectiveness vs. Efficiency 4

1.2 Problem Statement 6

1.2.1 User Adaptive Wireless Video Streaming 6
1.2.2 Protocol Adaptive Wireless Video Teleconferencing 6
1.2.3 User Adaptive Wireless Video Teleconferencing 8
1.2.4 MAC Layer Optimization to Improve TCP Performance over Wi-Fi 8
1.2.5 Delay Constrained MAC Layer Adaptation to Improve Wireless Video Teleconferencing 9

1.3 Organization of The Dissertation 10

2 REVIEW OF THE LIMIT OF HUMAN VISION SYSTEM **12**

2.1 Factors Affecting Human Perception of Visual Information 12
2.2 Limits of Human Vision 12

2.2.1 Visual Acuity 12
2.2.2 Viewing Distance 14
2.2.3 Spatial Frequency 15

2.2.4 Contrast and Contrast Sensitivity . . . 17
2.2.5 Contrast Sensitivity Function . . . 18

2.3 User Adaptive Video Coding . . . 19

3 USER ADAPTIVE WIRELESS VIDEO STREAMING . . . 21

3.1 Introduction and Related Work . . . 21
3.2 System Design . . . 22

3.2.1 UDASH Client Architecture . . . 22
3.2.2 Example Scenarios Showing The Benefits of UDASH . . . 23

3.2.2.1 UAV Helps DASH Traffic . . . 24
3.2.2.2 UAV Helps Non-DASH TCP Traffic . . . 25

3.2.3 Benefit Analysis of UDASH . . . 25
3.2.4 Conditions When UAV Helps . . . 27

3.3 System Implementation . . . 27

3.3.1 DASH Protocol . . . 27
3.3.2 User-adaptive Video (UAV) . . . 28
3.3.3 MAC Layer Protocol . . . 29

3.4 Evaluation . . . 29

3.4.1 Simulation Setup . . . 29
3.4.2 Performance Metrics . . . 30
3.4.3 Simulation Results . . . 30

3.5 Summary . . . 33

4 PROTOCOL ADAPTIVE WIRELESS VIDEO TELECONFERENCING . . . 34

4.1 Introduction and Related Work . . . 34
4.2 Problem Analysis and Motivation . . . 37

4.2.1 Testbench Setup . . . 37
4.2.2 Background of WebRTC . . . 40
4.2.3 Experimental Setup . . . 41
4.2.4 Lower Received Video Bit Rate due to Packet Losses . . . 41
4.2.5 Video Freezes due to Packet Losses . . . 43

4.2.6 Motivation . . . 44

4.3 Congestion Detection . . . 46

4.3.1 Congestion Metric . . . 46
4.3.2 Evaluation of the congestion metric . . . 49

4.4 Cross Layer Approach . . . 56

4.4.1 Approach Description . . . 57
4.4.2 Analytic Model . . . 59
4.4.3 Performance Evaluation . . . 62

4.4.3.1 Controlled Scenario . . . 62
4.4.3.2 Realistic Scenario . . . 63

4.5 MAC Layer Only Approach . . . 63

4.5.1 Challenges in Adjusting The Retry Limit . . . 63
4.5.2 Algorithm . . . 64
4.5.3 Performance Evaluation . . . 67

4.5.3.1 Improved Video Bit Rate . . . 67
4.5.3.2 Improved Target Sending Rate . . . 67
4.5.3.3 Reduced Video Freezes . . . 68
4.5.3.4 Impact on Competing Traffic . . . 71

4.6 Comparisons between The Two Approaches . . . 72
4.7 Summary . . . 73

5 USER ADAPTIVE WIRELESS VIDEO TELECONFERENCING . . . 74

5.1 Related Work and Motivations . . . 74
5.2 System Design . . . 75

5.2.1 System Architecture . . . 75
5.2.2 Inferring the Viewing Distance . . . 77
5.2.3 Determining The Pixel Density . . . 78
5.2.4 User-adaptive Video Encoding . . . 78

5.3 System Implementation . . . 79
5.4 Evaluation . . . 80
5.5 Conclusion . . . 81

6 MAC LAYER ADAPTATION TO IMPROVE TCP PERFORMANCE OVER WI-FI . . . 83

6.1 Introduction and Related Work . . . 83
6.2 Problem Analysis . . . 86

6.2.1 TCP performance degradation . . . 86
6.2.2 Unnecessarily Reduced and Fluctuated TCP Performance . . . 88

6.3 System Design and Implementation . . . 89

6.3.1 Principles . . . 90
6.3.2 Retry Limit Impact on RTT and TCP Performance . . . 92
6.3.3 Retry Limit Adaptation Algorithm . . . 95

6.4 Evaluation . . . 98

6.4.1 Simulation Testbench . . . 98
6.4.2 Impact on Competing Traffic . . . 102
6.4.3 Improved TCP Performance . . . 102

6.5 Summary . . . 103

7 DELAY CONSTRAINED MAC LAYER ADAPTATION TO IMPROVE WIRELESS VIDEO TELECONFERENCING . . . 105

7.1 Related Work and Motivations . . . 105
7.2 Delay Constrained MAC Layer Adaptation Algorithm . . . 107
7.3 Performance Evaluation . . . 107

7.3.1 Testbench and Experiments Setup . . . 107
7.3.2 Experiments Results . . . 111

7.4 Joint Discussions on The Two Adaptation Algorithms . . . 112
7.5 Summary . . . 114

8 CONCLUSIONS AND FUTURE WORK . . . 115

8.1 Conclusions . . . 115
8.2 Future Work . . . 117

BIBLIOGRAPHY **119**

LIST OF TABLES

3.1 Simulation parameters 23

3.2 Simulation results 25

3.3 Set of available bit rate in MPD 29

4.1 Notations for the Analytic Model 60

LIST OF FIGURES

1.1 Dissertation Structure 11

2.1 Characteristics of viewing setup [78] 13

2.2 Ambient illuminance in different environments [12] 13

2.3 Mobile video viewed with different surrounding light level. (a) indoors, (b) direct sunlight, (c) contrast perceived as in (a), (d) contrast perceived as in (b) [100] 13

2.4 (a) Snellens chart [95] (b) Visual acuity calculation [66] 14

2.5 Probability distribution of smartphone reading distances [78] 15

2.6 (a) Sinusoidal grating (b) Computation of spatial frequency 16

2.7 Finding highest spatial frequency limit for the HVS [66] 16

2.8 Gabor patches with progressively reduced contrast 18

2.9 Illustration of shape of contrast sensitivity [88] 19

3.1 The overall UDASH architecture. 23

3.2 Network path with last mile Wi-Fi network. 30

3.3 Rebuffering probability 31

3.4 Total playing time when the playback is HD quality 32

3.5 Average TCP throughput for cross FTP traffic 33

4.1 A typical real-time video system 35

4.2 WebRTC generated real video traffic passes through an OPNET-based network emulator . . . 38

4.3 Screenshot of the emulated OPNET scenario . . . 38

4.4 Video bit rate received by the video receiver . . . 41

4.5 When FEC is turned On and OFF: (a) Target sending rate A_s at Laptop A (b) Received video bit rate at Laptop B . . . 42

4.6 Video freeze duration on Laptop B: (a) FEC is turned off (b) FEC is turned on . . . 43

4.7 Realtime calculations of Eq. 4.6 in WebRTC-based video teleconferencing experiments . . . 48

4.8 Network resource occupation of hidden terminal traffic . . . 50

4.9 Segmented network resource occupation of hidden terminal traffic . . . 52

4.10 Comparison between theoretical results and experimental results of estimated MAC layer capacity as a function of the arrival rate . . . 54

4.11 Realtime calculations of Eq. 4.5 in WebRTC-based video teleconferencing experiments . . . 54

4.12 In the cross layer approach, the packet flow of the spoofed RTCP packets is indicated by blue solid lines, the packet flow of the normal RTCP packets (generated by the video decoder and received from the network) is indicated by red dashed lines, and the mixed flow is indicated by purple solid lines. . . . 59

4.13 The freeze duration τ as a function of the feedback delay T_{feedback}. . . . 61

4.14 The video freeze duration resulting from controlled packet losses at 1 packet per second (a) when EPLF is off, and (b) when EPLF is on. . . . 62

4.15 The video freeze duration with realistic packet losses for (a) the case where EPLF is off and (b) the case where EPLF is on. . . . 63

4.16 Received video bit rate on Laptop B: (a) FEC is turned off (b) FEC is turned on. R_{ex} stands for retry limit extension in Algorithm 3. $R_{ex} = 0$ means that our adaptation algorithm is not used. This definition applies to all following figures. 67

4.17 Target sending rate of Laptop A: (a) FEC is turned off (b) FEC is turned on. 68

4.18 Video freeze duration on Laptop B (FEC is turned off): (a) $R_{ex} = 0$, (b) $R_{ex} = 3$,(c) $R_{ex} = 5$ and (d) $R_{ex} = 7$. 69

4.19 Aggregate throughput of competing traffic, when FEC in WebRTC is: (a) turned off; (b) turned on. 71

5.1 System architecture with two clients and an MCU. 76

5.2 The calculation of the viewing angle. 78

5.3 The arrival video bit rate (blue solid line) v.s. the departure video bit rate (red dashed line), for the cases of user adaptive transcoding (a) being turned on, and (b) being turned off. 81

5.4 Subjective testing scores for the case where user adaptive transcoding is used (blue stars) and the case where user adaptive transcoding is not used (red circles). The error bars stand for 95% confidence intervals. 82

6.1 TCP performance degradation due to large channel error duration 86

6.2 TCP performance degradation due to small channel error duration 87

6.3 TCP performance degradations: (a) Congestion window at the TCP sender (b) Average TCP traffic received at the TCP receiver. 89

6.4 How the value of Retry Limit affects TCP performance (Internet delay = 30ms): (a) Round trip time calculated at the TCP sender (b) Average TCP traffic received at the TCP receiver. 95

6.5 A typical lossy communication network simulated in OPNET 99

6.6 Aggregate throughput of competing traffic, when Internet delay is: (a) 30 ms; (b) 60 ms. 100

6.7 Round trip time calculated at TCP sender, when Internet delay is: (a) 30 ms; (b) 60 ms. 100

6.8 Congestion window size at TCP sender, when Internet delay is: (a) 30 ms; (b) 60 ms. 101

6.9 Zoom in details (extended figure of the Fig. 6.8) of congestion window size at TCP sender, when Internet delay is: (a) 30 ms; (b) 60 ms. 101

6.10 Average TCP traffic received at the TCP receiver, when Internet delay is: (a) 30 ms; (b) 60 ms. 104

7.1 Difference in the transmit delay with different values of R_{ex}. 111

7.2 (a) target sending rate comparison when using fixed value of retry limit ($R_{ex} = 7$) and transmit delay constrained retry limit (b) Video freeze duration on Laptop B (FEC is turned off) when using transmit delay constrained retry limit. 112

ABSTRACT

With the popularity of and the advances in wireless networking technologies, wireless multimedia traffic has grown dramatically in recent years. Despite having many advantages, wireless multimedia services, particularly video services, still pose a number of challenges due to the time-varying, error-prone and bandwidth-fluctuating channels in the wireless networks. Therefore, provisioning end-to-end Quality of Service and Quality of Experience (QoS and QoE) of video transmission over wireless channels is of great importance.

Video transmission is often described to be bursty as video is basically a sequence of frames transmitted at a particular frame rate. A video frame cannot be decoded or played out at the receiver side until all or most of its transmitted constituent packets are received in time. Depending on the application scenarios, video services may have different emphases in terms of QoE and QoS. While video streaming (e.g., Netflix and YouTube) allows for modest delay (on the order of a few seconds) at the beginning of the playout, video teleconferencing (e.g., FaceTime and WebRTC) is much more delay constrained (less than a few hundred milliseconds). This is because in real-time video systems, each frame must be delivered and decoded by its playback time, and any packet that is retransmitted due to loss in the last transmission or arriving late becomes useless when its stringent decoding and display deadline cannot be met. In this dissertation, we propose several optimization algorithms to improve the QoE and QoS for both video streaming (non real-time) and video teleconferencing (real-time) over wireless networks.

In optimizing wireless video streaming, we focus on MPEG-DASH (ISO/IEC Standard 23009-1), the current standard for video streaming. We optimize video streaming by leveraging a technique called User Adaptive Video (UAV), which exploits the perceptual limits of the human visual system to modulate a video stream's bit rate based on the viewing conditions, such as viewing distance and ambient illuminance, resulting in significant

bandwidth saving without perceived loss of quality to the user. UAV presents an opportunity to significantly improve the efficiency of DASH by not requesting unnecessarily high bit rate videos. We design UAV-enabled DASH (UDASH) and evaluate its performance in Wi-Fi networks. Simulation results show that UDASH in a Wi-Fi network has the benefits of not only significantly improving the video streaming performance such as reducing the rebuffering probability, but also enhancing the performance of cross traffic.

In addition, the MPEG-DASH standard uses TCP as the underlying transport layer protocol, and more importantly, TCP is one type of dominant traffic in the Internet. Therefore, we investigate how to improve TCP performance in wireless networks. We identify two issues of TCP performance degradation due to common channel errors via both analytical study and simulations in a typical Wi-Fi network. Motivated by these issues, a MAC layer optimization technique is proposed, which is based on the adaptation of the Retry Limit parameter after considering TCP traffic characteristics and throughput model. The evaluation results confirm that the proposed technique achieves higher performance gain.

In optimizing video teleconferencing, we consider WebRTC, which is Google's open source real-time communication framework. In wireless networks such as those based on IEEE 802.11, packet losses due to fading and interference are often misinterpreted as indications of congestion, causing unnecessary decrease in the data sending rate due to congestion control by the RTCP protocol working beneath WebRTC and above RTP. For delay-constrained applications such as video teleconferencing, packet losses may result in excessive artifacts or freeze in the decoded video. We propose a simple and yet effective mechanism to detect and reduce channel-caused packet losses by dynamically adjusting the retry limit parameter of the IEEE 802.11 protocol. Since the retry limit is left configurable in the IEEE 802.11 standard, and does not require cross-layer coordination, the proposed scheme can be easily implemented and incrementally deployed. We also propose to use a delay constrained retry limit adaptation algorithm to control transmission delays so that delay constraints required by different application scenarios can be met. Experimental results of applying the proposed scheme to a WebRTC based real-time video communication prototype show significant performance gain compared to the case where retry limit is configured

statically.

In addition to the optimization techniques proposed for the IEEE 802.11 protocol, we also propose a cross-layer approach to optimize video teleconferencing, termed early packet loss feedback (EPLF). In EPLF, if a packet loss is due to channel errors, the MAC layer directly feeds back the loss information to the RTP layer with a spoofed RTCP packet that carries a NACK message so that the RTP layer can retransmit the lost RTP packet. Since the whole feedback process takes place in the same device (the video sender), the latency is negligible in relation to the RTT, and hence the term 'early' in EPLF. Theoretical analysis and prototype-based experimental results show that EPLF almost completely eliminates channel-caused video freezes in the decoded video while improving congestion control.

Furthermore, we also apply the technique of UAV to video teleconferencing to further reduce bandwidth consumption, and build a prototype based on WebRTC and Licode (a video teleconferencing hub platform) to validate the bandwidth savings.

Chapter 1

INTRODUCTION

In this chapter we first describe the background and overall motivations of our research. We then summarize the problems investigated in the dissertation along with the specific motivation for each problem. We end this chapter with the organization of the dissertation.

1.1 Background and Overall Motivations

Videos are undoubtedly the most important and effective carriers of information. With the popularity of and the advances in wireless networking technologies, wireless video traffic has grown dramatically in recent years. Recent statistics show that video traffic accounts for the highest percentage of the traffic mix in the Internet. According to Cisco Visual Networking Index [26], mobile video traffic will increase thirteen-fold and count for 66% of the mobile traffic by the year 2019.

Video service is the most demanding among all multimedia services. It generates a huge amount of data that need to be transmitted and processed in a timely manner, which would be impossible/infeasible without highly efficient compression schemes. Standard video compression technologies (such as H.264 and VP8) exploit the spatial and temporal redundancy in uncompressed video to achieve a high compression ratio, which, however, makes compressed video vulnerable to transmission errors. A compressed/encoded video frame cannot be decoded or played out at the receiver side until all or most of its transmitted constituent packets are received in time. As a result, packet losses due to transmission errors often lead to serious video quality degradation, like artifacts in the decoded video, which affect not only the current frame, but also subsequent frames because of error propagation resulting from the use of motion-compensated prediction from previous frames. Numerous

error control mechanisms (e.g., error resilient video coding, error concealment, channel coding, retransmission) have been proposed to address the error-sensitivity issues, and various schemes (e.g., buffering) have been designed for adapting to the time-varying video content characteristics and wireless channel conditions. However, most of the previous designs suffer from various drawbacks. For example, most of the previous designs focus on video streaming services that allow a relatively large end-to-end delay (i.e., several seconds) and therefore may not be suitable for interactive video services that have stringent delay requirements (e.g., for video teleconferencing, the end-to-end delay is required to be below 100 ms [9]). We elaborate various reasons in the following subsections.

1.1.1 Video Streaming vs. Video Teleconferencing

Depending on application scenarios, two main types of video traffic exist: interactive video and streaming video. Each type of video has different emphases in terms of Quality of Experience (QoE) and Quality of Service (QoS), among which the tolerable delay is the important differential factor. While video streaming (e.g., Netflix and YouTube) allows for modest delay (in the order of a few seconds) at the beginning of the playout, interactive video such as video teleconferencing (e.g., Facetime [8] and WebRTC [92]) is much more delay constrained (less than a few hundred milliseconds) because in real-time video systems, each frame must be delivered and decoded by its playback time, and any packet that is retransmitted due to loss in the last transmission or late arrival becomes useless if its stringent decoding and display deadline cannot be met. In this dissertation, we investigate both types of traffic and propose optimization algorithms accordingly.

1.1.2 QoS vs. QoE

Quality of Experience (QoE) is the perceptual Quality of Service (QoS) from the users' perspective. While monitoring and controlling QoS parameters (e.g., bitrate, delay, jitter) of the video transmission system is important for achieving high video quality, it is more crucial to evaluate video quality from the users' perspective, which is perceived as QoE or user-level QoS. For video service, the relationship between QoE and QoS (such as coding parameters and network statistics) is complicated because users' perceptual video

quality is subjective and diversified in different environments. Traditionally, QoE is obtained from subjective tests, where human viewers evaluate the quality of tested videos under a laboratory environment. To avoid the high cost and offline nature of such tests, objective quality models are developed. The major purpose is to identify the objective QoS parameters that contribute to user perceptual quality, and map these parameters to user QoE [103]. Subjective test results are often used as ground truth to validate the performance of the objective quality models. Most of the objective quality models are based on how the Human Visual System (HVS) receives and processes the information of the video signals [103]. One of the commonly used methods is to quantify the difference between the original video and the distorted video, and then weigh the errors according to the spatial and temporal features of the video. However, the need to access the original videos hinders online QoE monitoring. In order to develop QoE prediction models that do not depend on original videos, network statistics (such as packet loss) and spatiotemporal features extracted or estimated from the distorted videos, are usually leveraged [103].

In this dissertation, different from aforementioned prior efforts, we propose a new real-time QoE predication model. Specifically, we are motivated to design user adaptive video streaming and video teleconferencing by a recently proposed and validated video coding technique called user adaptive video (UAV) [90][89]. UAV is a new technique that exploits the fact that the human visual system (HVS) cannot perceive spatial frequencies in an image that are above a certain limit (or cutoff frequency), which is influenced by the viewing conditions such as the viewing distance, ambient luminance and display characteristics. Frequency components above the limit can be removed to reduce the information in an image before the conventional video compression is applied, resulting in significant increase of image/video compression efficiency and reduction of bandwidth needed during transmission without perceived loss of quality to the user. In other words, UAV presents an opportunity to significantly improve the efficiency of video applications by not requesting unnecessarily high bit rate video.

1.1.3 Effectiveness vs. Efficiency

As mentioned earlier, numerous error control mechanisms (e.g., error resilient video coding, error concealment, channel coding, retransmission) have been proposed to address the error-sensitivity issue of compressed video traffic and various schemes (e.g., buffering) have been designed for adapting to the time-varying video content characteristics and wireless channel conditions. However, effectiveness and efficiency of those designs are still open issues.

Effectiveness means a method is adequate to accomplish a purpose. Back to those aforementioned error control designs, some of them may not be effective all the time. For example, some error concealment techniques can help stop error propagation, such as the "frame copy" used in WebRTC where a frame that cannot be correctly decoded is replaced by the last correctly decoded frame. Such techniques, however, may cause video freezes. Error resilient video coding works well with light packet losses, but may not work with bursty packet losses. In addition, application layer Forward Error Correction (FEC) is a commonly used scheme for data protection. However, it introduces extra complexity, overhead and delay, which are undesirable to applications with stringent delay constraints such as in the case of video telephony and its use should be minimized.

Efficiency means a method performs or functions in the best possible manner with the least waste of time and/or effort. Some of those aforementioned error control designs may work effectively most of the time, but the efficiency may not be satisfactory. Again, take the application layer FEC as an example. FEC can be performed either among the RTP packets of a single frame or among the RTP packets of multiple frames. Considering the case of single frame FEC which is used in WebRTC, the overhead is lower bounded by the number of RTP packets in a frame. For a bit rate of 2 Mbps, an MTU of 1500 bytes, and a frame rate of 15 frames per second (fps), a frame consists of less than 12 packets and the minimum overhead of single-frame FEC is 1/12 = 8.33%. In practice, the overhead could be much higher. For example, [80] shows that for single-frame FEC, the overhead varies from 40% to 20% as the video bit rate increases from 0.5Mbps to 1.2Mbps with a packet loss rate of 5% and burst length of 2. In addition, the overhead (or redundancy) in application-layer

FEC is often adaptive to the packet loss rate: the higher the loss rate, the higher the overhead. According to our prototype-based real experiments (Chapter 4), FEC overhead can be up to 50%.

Lastly, it is worthy reviewing the efficiency requirement from the perspective of mobile devices, which consume a significant portion of the global multimedia traffic. On one hand, many mobile devices are already matching and surpassing HDTV in terms of graphics capabilities. For instance, mobile devices often feature high-density "retina" screens with 1280*720, 1920*1080, or even higher resolutions. They also come equipped with powerful processors, making them possible to receive, decode and play HD-resolution videos. On the other hand, network and battery resources in mobile devices remain limited. Wireless networks, including the latest 4G/LTE networks and Wi-Fi networks, are fundamentally constrained by the capacities of their base stations/APs. Each base station's or AP's capacity is shared among its users, and it can be saturated by only a limited number of users simultaneously watching high-quality videos [3]. High data rates used to transmit video also cause high energy consumption, draining their batteries quickly. All these issues suggest that efficiency in mobile video services is very important.

In this dissertation, We propose several adaptation algorithms that consider both effectiveness and efficiency. For instance, user adaptive DASH and user adaptive video teleconferencing are able to reduce bandwidth usage by up to 40% without perceived loss of quality to the users, according to the experiment results collected from real-world implementations. Such significant amount of reduction in bandwidth usage not only dramatically improves energy efficiency and video smoothness, but also helps other cross traffic (i.e., the saved bandwidth can be used by other cross traffic) to create a win-win situation. We also propose a real-time, light-weight and passive algorithm to effectively detect wireless network congestion, and then propose two different algorithms to almost completely eliminate packet losses in wireless networks. The evaluation results collected from our realistic WebRTC-based testbed confirm that our approach can significantly improve the QoE of the received video by dramatically reducing video freezes or increasing the video bit rate. Compared to the application-layer FEC approach (the default error recovery mechanism in WebRTC), our

approach can achieve the same goal with minimum cost. More importantly, our approach could be used in conjunction with application-layer FEC to further improve QoE.

1.2 Problem Statement

1.2.1 User Adaptive Wireless Video Streaming

HTTP streaming has been widely adopted for multimedia delivery. Dynamic Adaptive Streaming over HTTP (DASH) is an emerging standard ([68]) for adaptive HTTP streaming to enable interoperability in the industry. User Adaptive Video (UAV) is a new technique that exploits the perceptual limits of the human visual system to modulate a video stream's bit rate based on the viewing conditions, such as viewing distance and ambient illuminance, resulting in significant bandwidth reduction without perceived loss of quality to the user. UAV presents an opportunity to significantly improve the efficiency of DASH by not requesting unnecessarily high bit rate video. Due to the random access nature of the Wi-Fi MAC protocol and the intricate interaction among DASH traffic flows, it is not clear whether UAV will manifest its benefits in Wi-Fi networks.

In this research, We design UAV-enabled DASH (UDASH) and evaluate its performance in Wi-Fi networks. The performance of UDASH is demonstrated through simulations and comparisons with other DASH clients not using UDASH.

Chapter 3 discusses this research in details, which has been published in [19].

1.2.2 Protocol Adaptive Wireless Video Teleconferencing

In wireless networks such as those based on IEEE 802.11, packet losses due to fading and interference are often misinterpreted as indications of congestion by the congestion control protocol at the higher layers, causing unnecessary decrease in the data sending rate. For delay-constrained applications such as video teleconferencing, packet losses may result in excessive artifacts or freezes in the decoded video.

WebRTC utilizes both proactive and reactive packet loss mitigation methods [80]. The proactive method used by WebRTC is packet-level FEC. However, FEC adds redundancy to mitigate packet losses at the cost of high overhead. Note that the overhead (or

redundancy) in application-layer FEC is often adaptive to the packet loss rate: the higher the loss rate, the higher the overhead. However, if not using FEC, the reactive packet loss mitigation method alone is not sufficient to mitigate packet losses. This is because the reactive approach is based on end-to-end packet-loss feedback. In one reactive method, the video encoder encodes a future video frame as an I-frame upon receiving an I-frame request or a frame-loss feedback. However, this may be undesirable in practice because generating an I-frame for each lost frame may either suddenly and substantially increase the video bit rate, causing network congestion, or lead to poor video quality given a fixed video bit rate, since the coding efficiency of an I-frame is often much lower than that of a P-frame. Also, all the frames after the lost frame and before the new I-frame will suffer from artifacts or video freezes, which last about one round trip time (RTT). Alternatively, the RTP layer at the sender can retransmit a lost RTP packet upon receiving a packet-loss feedback from the receiver [49], and again the value of RTT is critical to the video quality.

In this research, we propose two solutions:

1. (Cross layer approach) The MAC layer of the video sender detects a packet loss by not receiving a positive acknowledgement within the maximumly allowed number of retransmissions. On behalf of the video receiver, inside the video sender the MAC layer sends a spoofed negative acknowledgement (NACK) to the RTP layer to trigger an RTP layer retransmission in time, which effectively reduces the RTT and makes the receiver probably to be unaffected by the packet loss.
2. (MAC layer only approach) The MAC layer adaptation grants the packet which will experience an imminent loss (i.e., the packet is about to be discarded and considered as lost by a higher-layer protocol after reaching the retry limit) higher priority or more opportunities to be transmitted to prevent the loss.

Details of this research are presented in Chapter 4, which have been published in [20, 64].

1.2.3 User Adaptive Wireless Video Teleconferencing

The human visual system (HVS) cannot perceive spatial frequencies in an image that are above a certain limit (termed cutoff frequency), which is influenced by viewing conditions such as the viewing distance, ambient luminance and display characteristics. Frequency components above the limit can be removed (or filtered) to reduce the information in an image before the conventional video compression is applied, thereby improving the efficiency of image/video compression. The filtering is referred to as *perceptual pre-filtering*, and the whole process (filtering together with the conventional video encoding) is referred to as *user adaptive video coding*.

Perceptual pre-filtering has been used to improve the video coding efficiency of streaming video [90, 89, 88]. However, it has not been applied to video teleconferencing, which is the focus of our research. In this research, we propose a network-based transcoding scheme for video conferencing to improve the video coding efficiency using perceptual pre-filtering. Specifically, we propose to implement perceptual pre-filtering in a network entity such as a Multipoint Control Unit (MCU). By analyzing the video sent from a client, the MCU infers the client's viewing conditions, which are then used to adapt the encoding of the video destined to the client. The scheme is implemented in a real-world video teleconferencing system called Licode.

The detailed study is presented in Chapter 5, which has been published in [65].

1.2.4 MAC Layer Optimization to Improve TCP Performance over Wi-Fi

The current video streaming standard MPEG-DASH uses TCP as the underlying transport layer protocol. In a DASH system, a video receiver is strictly based on TCP's performance to estimate the available bandwidth of a network and select suitable video quality dynamically. In addition, not only video streaming, about 90% of the DATA traffic in the Internet today is carried by TCP [55], and a majority of that traffic may be preferably transferred via a path over Wi-Fi which is significantly faster and cheaper than a cellular connection. Therefore, a satisfactory performance of TCP over Wi-Fi networks is thus essential to effectively design, deploy and manage a large number of applications in the Internet.

In this research, we study the issues leading to TCP performance degradation in a typical Wi-Fi network. Based on an analytical study and a set of simulations, we identify two types of TCP performance degradation due to common channel errors (e.g., collisions or interferences). Motivated by these analyses and simulation studies, we propose a MAC layer technique which optimizes the Retry Limit parameter in Wi-Fi networks to avoid triggering the costly TCP loss recovery process.

Details of the study are presented in Chapter 6.

1.2.5 Delay Constrained MAC Layer Adaptation to Improve Wireless Video Teleconferencing

In IEEE 802.11 standard wireless networks, transmit delay in the MAC layer refers to the aggregate time spent by a MPDU from the moment the MPDU reaches the head of the sending queue to the moment the MPDU is removed from the sending queue due to either its successful delivery (receiving an ACK from the destination MAC) or being discarded after reaching the retry limit. When network conditions (e.g., aggregate traffic load, available bandwidth, etc) fluctuate, transmit delay of each MPDU may vary significantly under the influence of several non-deterministic factors: number of retransmissions, values of random backoff time, backoff time freeze (busy channel), and so on. Furthermore, due to the following two reasons, transmit delay of a real-time video MPDU needs to be upper bounded:

- The video receiver imposes a video decoding deadline. The arrival of video packets at the receiver that take a large transmit delay and after the deadline is a waste of network resources.
- Due to unexpected backoff timer freeze after carrier sensing, random backoff duration in a noisy channel or a heavily loaded channel can be quite different. A sudden increase of transmit delay means the corresponding MPDU takes more time to be transmitted and less time for other MPDUs to be transmitted within the same time period. This leads to a sudden decrease in the received bit rate at the receiver which in turn induces a reduction in the sending rate at the sender. This is because the receiver usually uses the

average received bit rate to estimate available bandwidth and then report this receiver estimated bandwidth to the sender.

For these reasons, we propose to use a delay-constrained retry limit to control the transmit delays. The delay constraint should be assigned by the application layer according to the applications' specific requirements for the transmitted media data, as different applications will place different requirements on the performance even in the future 5G system [7]. Based on the delay-contrained retry limit optimization, we propose a MAC layer adaptation algorithm and implement it on a prototype testbed. Our real traffic based emulations confirm that the proposed algorithm is able to bring significant performance improvement.

Details of this work are presented in Chapter 7, which have been published in [21].

1.3 Organization of The Dissertation

The remainder of this dissertation is organized as follows. We provide an overview of this thesis work in Chapter 1. We review the limits of human vision system and introduce the user adaptive video (UAV) encoding in Chapter 2. We propose an UAV-enabled DASH client for wireless video streaming and analyze its benefits on improving QoE performance in Chapter 3. In Chapter 4, based on WebRTC, we propose two different approaches to optimize wireless video teleconferencing, and present extensive and realistic evaluation results for each approach. In Chapter 5, we improve wireless video teleconferencing by proposing an online UAV encoding method. In Chapter 6, we explore and analyze the potentials how those optimization approaches proposed in Chapter 4 could help general TCP traffic in wireless networks. In Chapter 7, we propose an delay-constrained MAC layer adaptation algorithm to further optimize wireless video teleconferencing. Chapter 8 concludes the thesis with future research directions. The overall structure of the dissertation is shown in Fig. 1.1.

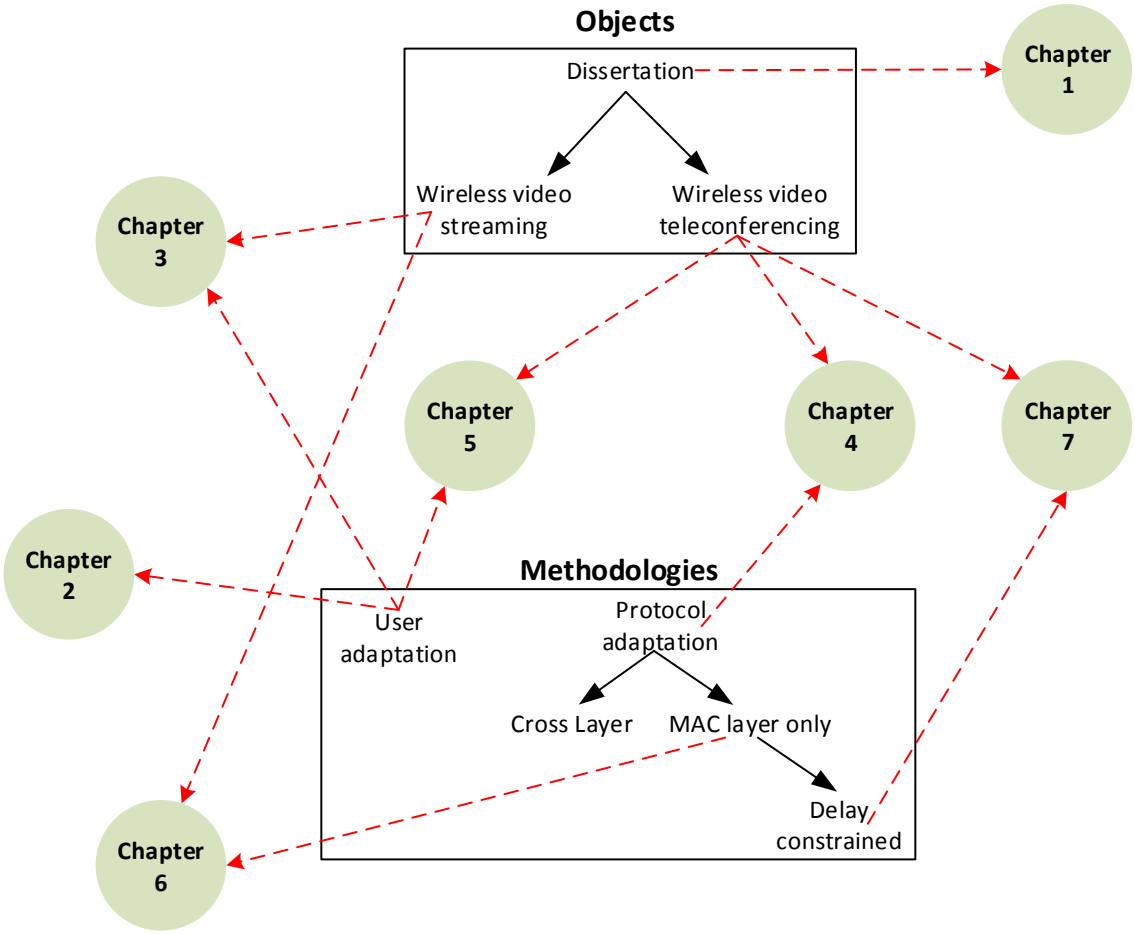


Figure 1.1: Dissertation Structure

Chapter 2

REVIEW OF THE LIMIT OF HUMAN VISION SYSTEM

In this chapter, background information of human visual limits [50] will be reviewed. After the review, the concept of user adaptive video coding is introduced.

2.1 Factors Affecting Human Perception of Visual Information

There are several factors that can affect a user's ability to discern the visual information rendered on an electronic screen. In Fig. 2.1, several important parameters of a viewing setup is shown. These include viewing distance, display size, viewing angle, and ambient light. The variation of ambient illuminance across several possible types of environments is captured in Fig. 2.2. This shows a very broad (5 orders of magnitude) range of this characteristic.

Fig. 2.3 illustrates some effects of ambient light on the visibility of information presented on a mobile screen when indoors and in direct sunlight conditions. The examples in Fig. 2.3 show that the impact of the viewing setup and the environmental factors can be very significant. A high-quality video becomes completely washed away under sunlight. Viewing distance may also significantly impact the amount of information that will be delivered. The further the viewer is from the screen, the more content in the video falls beyond the resolution capability of human vision, making it invisible to the viewer.

2.2 Limits of Human Vision

2.2.1 Visual Acuity

Visual acuity (VA) is a quantitative measure of the ability to identify black symbols on a white background at a standardized distance as the size of the symbols is varied. It is the most common clinical measurement of visual function which represents the smallest size

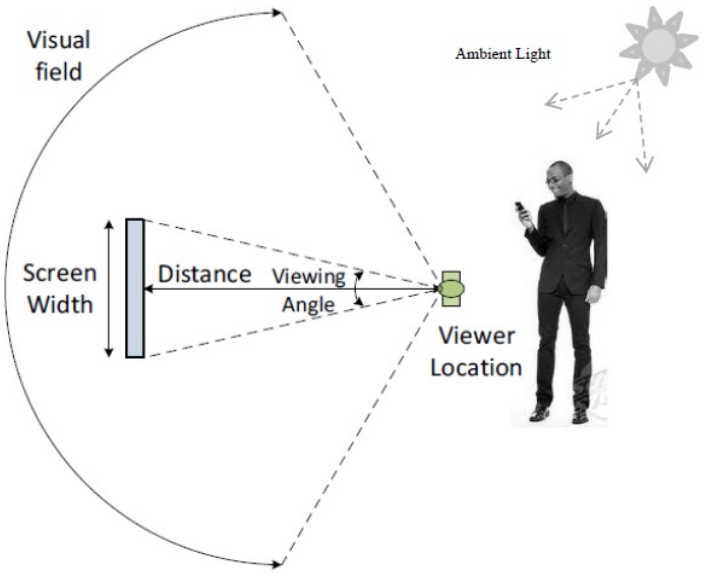


Figure 2.1: Characteristics of viewing setup [78]

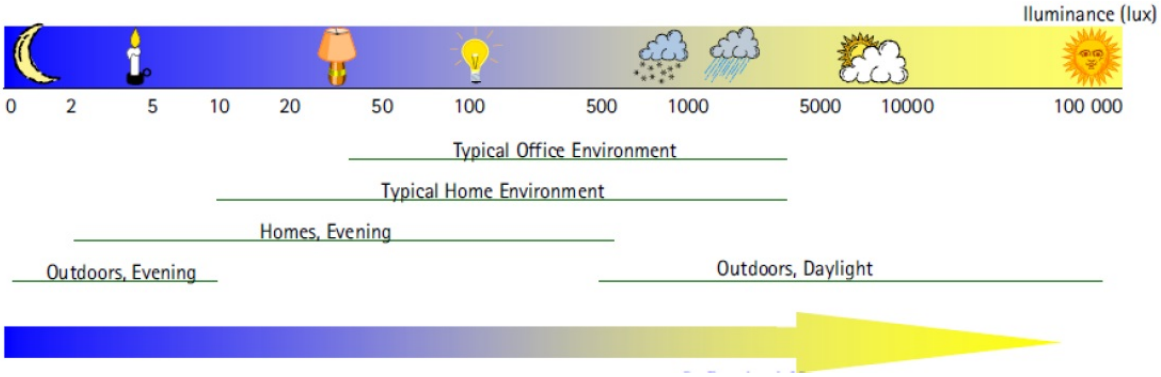


Figure 2.2: Ambient illuminance in different environments [12]

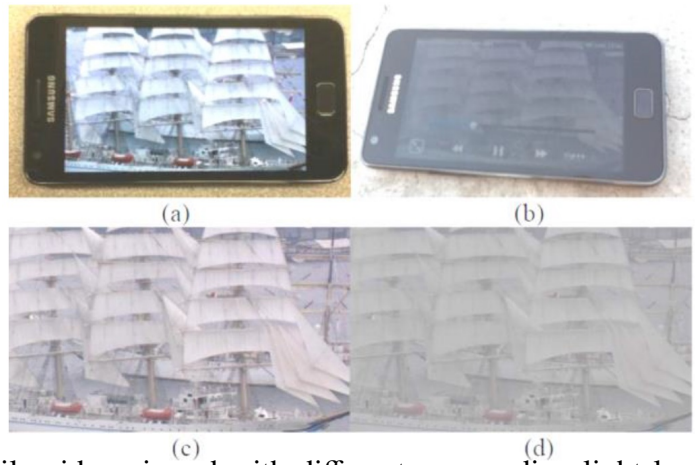
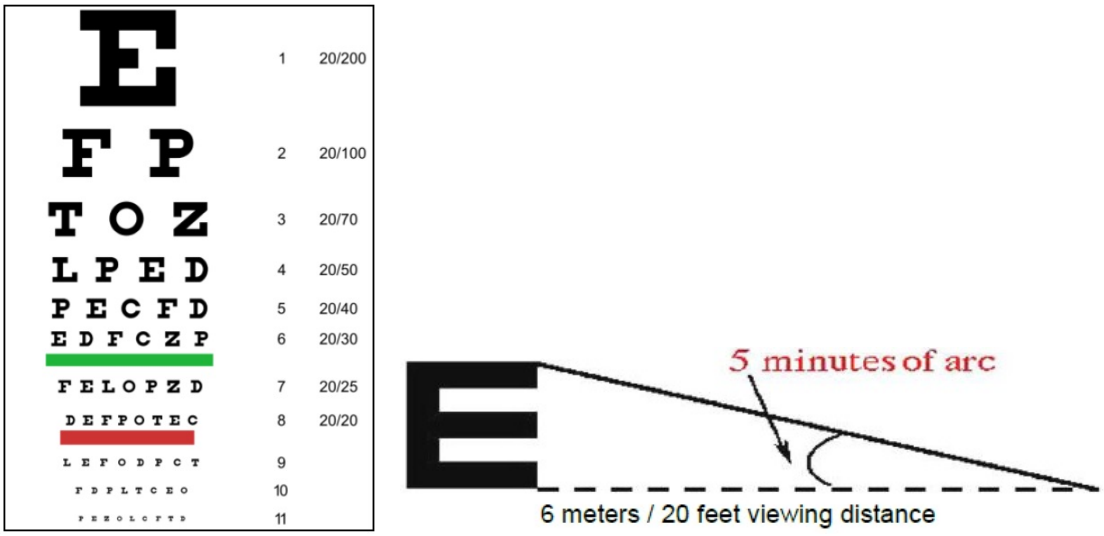


Figure 2.3: Mobile video viewed with different surrounding light level. (a) indoors, (b) direct sunlight, (c) contrast perceived as in (a), (d) contrast perceived as in (b) [100]



(a) (b)

Figure 2.4: (a) Snellens chart [95] (b) Visual acuity calculation [66]

that can be reliably identified. A visual acuity of 20/20 is described as meaning that a person can see detail from 20 feet away the same as a person with normal eyesight would see from 20 feet.

Visual acuity often is referred to as "Snellen" acuity [86]. The chart and the letters are named for a 19th-century Dutch ophthalmologist Hermann Snellen (1834-1908) who created them as a test of visual acuity. Snellen letters are constructed so that the size of the critical detail (stroke width and gap width) subtends 1/5th of the overall height. To specify a person's visual acuity in terms of Snellen notation, a determination is made of the smallest line of letters of the chart that he/she can correctly identify. Visual acuity (VA) in Snellen notation is given by the relation:

$$VA = D'/D \tag{2.1}$$

where D' is the standard viewing distance (usually 6 metres or 20 feet) and D is the distance at which each letter of this line subtends 5 minutes of arc.

2.2.2 Viewing Distance

The human visual system (HVS) uses two mechanisms to focus on objects: convergence and accommodation. Convergence denotes the eyes moving inward when focusing on nearby objects, and accommodation describes the focusing of objects of different distance

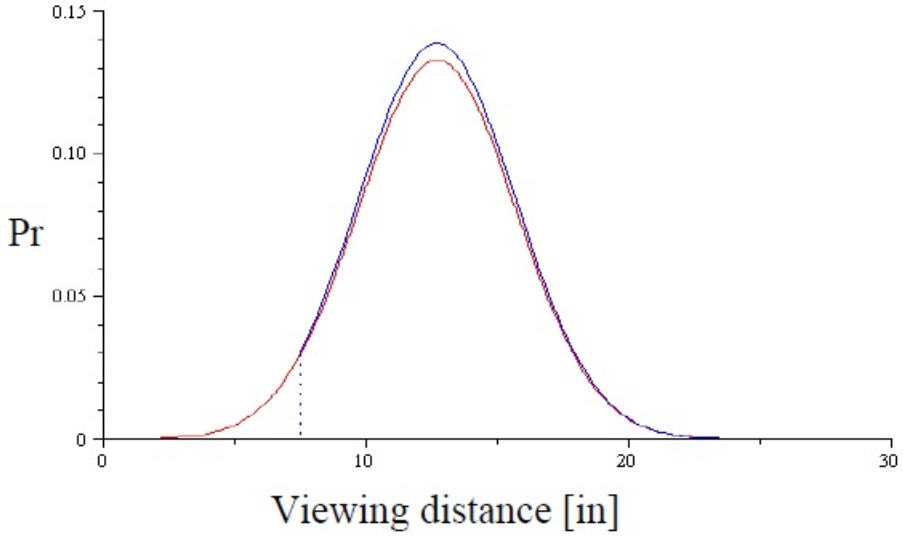


Figure 2.5: Probability distribution of smartphone reading distances [78]

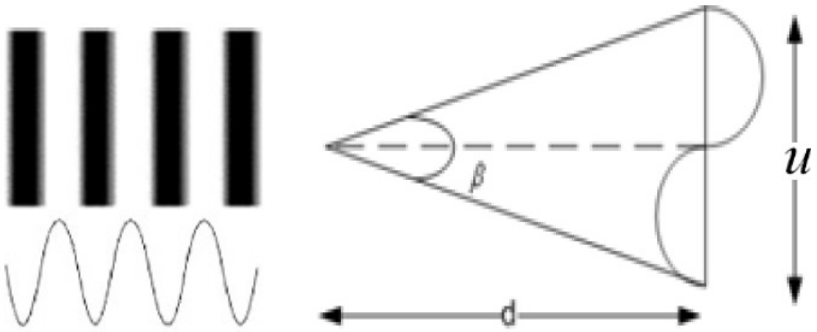
by means of physically deforming the lens of the eye. The default distance at which objects appear sharp is called the resting point of accommodation (RPA). RPA is around 75 cm for younger people and increases in distance with age [74]. The distance at which the eyes are set to converge when there is no object to converge on is called the resting point of vergence (RPV) [52]. RPV is 114 cm when looking straight ahead and drops to 89 cm when looking 30 degrees down [74].

Recently a study [101] has been conducted pertaining to the distances with which a person with normal 20/20 vision can be comfortable in reading text on smart phones. The result shows that viewing distances for a smartphone range from 7.5” to 23.6” with a mean distance 12.7” and standard deviation of 3”. The approximate shape of such a distribution, obtained by fitting Gaussian model, is shown in Fig. 2.5.

2.2.3 Spatial Frequency

The spatial frequency is a measure of how often sinusoidal components of a structure repeat per unit of spatial distance. It is also often described as the frequency of change per angular unit, capturing the relative position of a viewer to the image as being projected to the screen (see Fig. 2.6(a)).

As shown in Fig. 2.6(b), spatial frequency, u, of a sinusoidal grating (see Fig. 2.6(a))



(a) (b)

Figure 2.6: (a) Sinusoidal grating (b) Computation of spatial frequency

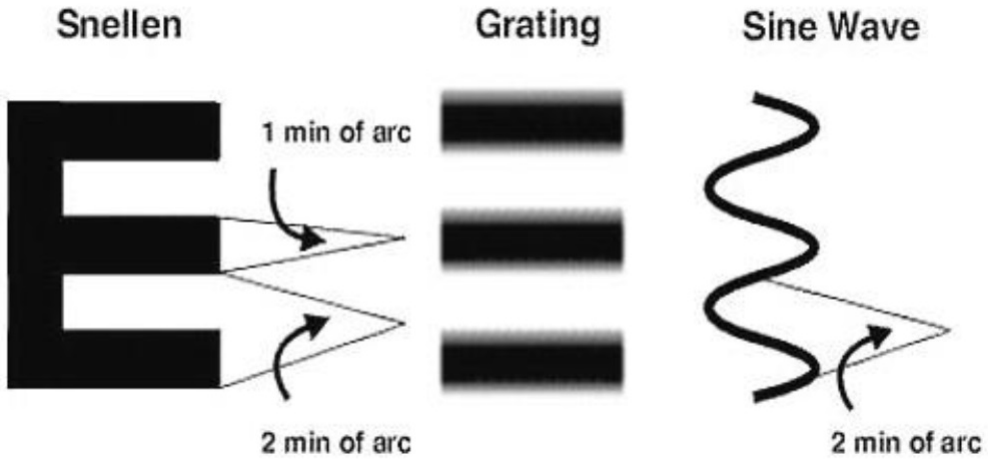


Figure 2.7: Finding highest spatial frequency limit for the HVS [66]

with a cycle length of n pixels can be computed (in cycles per degree) as

$$u = \frac{1}{\beta}, \beta = \frac{\pi}{360} \arctan(\frac{n}{2d\rho}) \tag{2.2}$$

where ρ is the display pixel density (in pixels per inch), d is the distance between viewer and the screen (in inches) and β is the angular span of one cycle of grating (in degrees).

The concept of visual acuity or 20/20 vision can also be understood as a limit in spatial frequency space. To illustrate this, consider Snellens E grating conversion presented in Fig. 2.7. It can be observed that for the 20/20 Snellen's letter E, there are 2 minutes of arc in one cycle. It is known that 60 minutes makes one degree and hence in a 20/20 letter, there are 30 cycles per degree (cpd). This means that 20/20 acuity implies ability to resolve frequencies of at least 30 cpd.

2.2.4 Contrast and Contrast Sensitivity

Contrast is a fundamental characteristic of displays or other visual sources capturing the dynamic range of luminance that they can reproduce. There are several alternative definitions of contrast used in the literature. The important ones for the work in this thesis will be the Michelson contrast, the Contrast Sensitivity, and the Contrast Ratio.

Michelson Contrast (C): The Michelson definition of contrast is used very commonly in vision research. Michelson contrast C is defined in [44]:

$$C = \frac{L_{max} - L_{min}}{L_{max} + L_{min}} \tag{2.3}$$

where L_{max} and L_{min} are luminances of darkest and brightest colors in an image or video projected to a screen. If follows from definition, Michelson contrast C ranges from 0 to 1.

Contrast Sensitivity (S): Contrast sensitivity S is most commonly defined as an inverse of the Michelson contrast:

$$S = \frac{1}{C} = \frac{L_{max} + L_{min}}{L_{max} - L_{min}} \tag{2.4}$$

The range of contrast sensitivities is from 1 to infinity. In other words, contrast sensitivity cannot be lower than 1.

Display contrast ratio (CR): Contrast ratio is the ratio between the luminances of the brightest (typically white) L_{white} and the darkest (typically black) L_{black} colors that a display device can reproduce:

$$CR = \frac{L_{white}}{L_{black}} \tag{2.5}$$

This Contrast ratio (CR) is commonly used by the display industry to characterize contrasts of TVs and monitors being produced. Such manufacturer-reported contrast ratios are typically measured in a dark room, and they can be very high (contrast ratios of 1000:1 or even 100000:1 are very common for modern displays). However, in the presence of ambient light contrast ratios can be several orders of magnitude lower [12].

Figure 2.8: Gabor patches with progressively reduced contrast

2.2.5 Contrast Sensitivity Function

Visual acuity is measured using high contrast letters (black symbols on white background). The contrast Sensitivity Function (CSF) is a more complete characteristic of human vision, obtained by considering images of different contrasts.

Some example images, so-called Gabor patches as used in CSF measurements, are shown in Fig. 2.8. Such patches are viewed from a distance limiting their angular span to a certain angle χ (usually between 2° and 12°). The maximum and minimum luminances, L_{max} and L_{min}, of such patches are also controlled so as to achieve different levels of their contrast. During each test, the contrast of the patch is progressively reduced until the point when a viewer can no longer detect it. This test is repeated for patches with different frequencies. It is also performed involving a fairly large (20 viewers +) panel of viewers.

The Michaels contrast level C_τ at which 50% of viewers say that they can see oscillations and the other 50% of viewers cannot is called the contrast visibility threshold C_τ. The inverse of it, $S = \frac{1}{C_\tau}$, is called the sensitivity threshold for Gabor patches with a certain spatial frequency u.

The Contrast Sensitivity Function is the collection of sensitivity thresholds measured across different spatial frequencies. An approximate shape of the CSF curve is illustrated in Fig. 2.9.

The two important points on the CSF curve are: maximum point (corresponding to about 3-5 [cpd]), and a point at which it approaches contrast sensitivity of 1. This farthest right point coincides with the visual acuity limit.

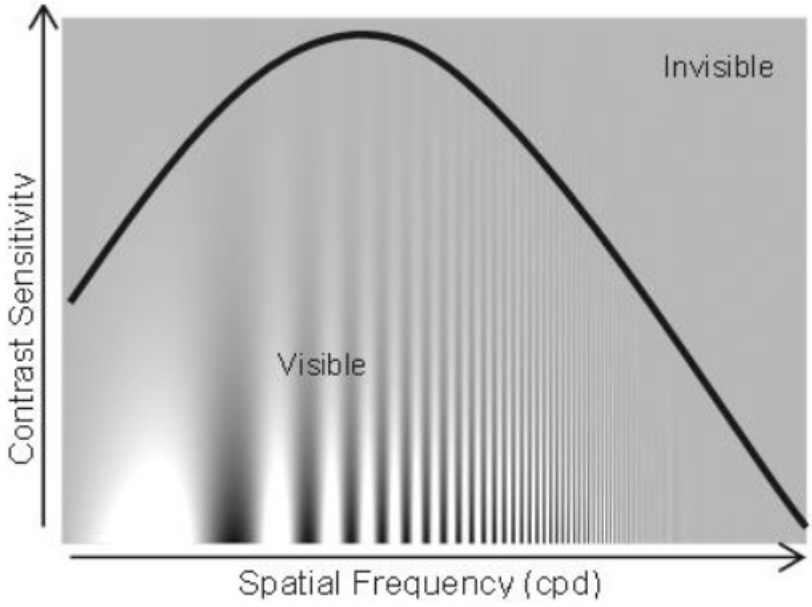


Figure 2.9: Illustration of shape of contrast sensitivity [88]

2.3 User Adaptive Video Coding

The human visual system (HVS) cannot perceive frequency components in an image that are above a certain threshold (or cutoff frequency) under a given viewing condition, which includes factors such as viewing distance, ambient luminance and display characteristics. The frequency components above the cutoff frequency can be removed to reduce the information in an image before the conventional video compression is applied, thereby further improving the efficiency of image/video compression.

In this thesis, we use the same contrast sensitivity function (CSF) model used in [18][67][10] to characterize this phenomenon. This model establishes a relationship between the *spatial frequency* (in cycles per degree or cpd) and the *contrast sensitivity*. The spatial frequency characterizes the oscillation of a sinusoid in an image with respect to the angular span of the sinusoid to the eyes. For a fixed sinusoid, as the viewing distance increases, the angular span decreases and hence the spatial frequency increases. The contrast sensitivity is the inverse of the Michelson contrast, and the higher it is, the lower the contrast is. The contrast sensitivity is determined by the contrast of the image itself, and the viewing conditions such as the ambient luminance and the reflection of the display. Therefore, as the viewing condition changes, the frequency components in an image that are visible to the

HVS also changes. Additionally, since the contrast sensitivity may vary significantly from region to region in an image, applying different cutoff frequencies instead of the same one to different regions of an image to filter the image prior to conventional video encoding may dramatically improve the video compression efficiency [90][89]. The filtering is referred to as *perceptual pre-filtering*, and the whole process (filtering together with the conventional video encoding) is referred to as *user adaptive video coding*.

Chapter 3

USER ADAPTIVE WIRELESS VIDEO STREAMING

3.1 Introduction and Related Work

HTTP streaming has been widely adopted for multimedia delivery. Dynamic Adaptive Streaming over HTTP (DASH) is an emerging standard for adaptive HTTP streaming to enable interoperability in the industry. In a DASH system, the content is offered at the server in different representations, which may provide different qualities (e.g., resolutions, quantization parameters) usually with different bit rates. Each version of the content is divided into small segments corresponding to relatively small time intervals, typically between two and ten seconds. This allows clients to select the most suitable video version dynamically, i.e., the version that matches the capabilities of their equipments and the currently available bandwidth.

We are motivated to design UAV-enabled DASH (UDASH) by a recently proposed and verified video coding technique called user adaptive video (UAV) [90][89], which exploits the perceptual limits of the human visual system to reduce the video bit rate without affecting the perceived video quality. The perceptual limit is the highest spatial frequency that can be perceived under a given viewing condition, which can be obtained by using the front-facing camera of a device to estimate, for example, user distance, number of viewers, and ambient illuminance. Any frequency component higher than the perceptual limit can be removed from an image without affecting the perceived quality of the image. As demonstrated in [90]and [89], UAV can result in bandwidth reduction of up to 70-80% without perceived loss of quality to the user. The perceptual limit is a function of viewing condition, which may be different from one user to another or vary over time for the same user. Thus, by making the DASH client aware of a user's viewing condition, a DASH client can

avoid requesting unnecessarily high bit rates, resulting in potentially significant bandwidth savings.

We are particularly interested in the benefits of using UDASH in Wi-Fi networks for two reasons. First, Wi-Fi has been widely used and there are many scenarios where network congestion can happen, for example, on campus, at airport, and in densely populated residential areas. Most Wi-Fi networks operate only in random access modes (e.g., DCF, EDCA[1]), without a central controller that allocates network resource for fairness and minimum bandwidth guarantee. This is in drastic contrast to cellular networks where resource allocation is centrally controlled so that the network resource saved due to UDASH may be shifted to disadvantageous users to reduce their rebuffering probabilities [99]. It is not clear whether UAV can provide a similar benefit in a Wi-Fi network, especially given the intricate interaction among DASH traffic flows.

Second, most existing DASH bit rate selection algorithms, e.g., [32] [39] [37] are designed for wired networks. However, due to interferences, path losses, competing wireless traffic and user mobility, the network condition in a wireless network is dramatically more dynamic than a wired network, making these algorithms ineffective for wireless networks. Few algorithms, [34] [40] [33], are developed for wireless networks. The use of UAV with DASH is a different approach that can potentially mitigate the deficiencies of existing rate selection algorithms.

Our proposed UDASH can incorporate any rate selection algorithm. Additionally, in [90] and [89], UAV is based on feedback and requires online cooperations between the client and server. Our work is a client-side-only approach without requiring online feedback. In [78], a-client-side-only approach is also proposed, but without any performance evaluation.

3.2 System Design

3.2.1 UDASH Client Architecture

A UDASH client has an additional UAV block compared to a DASH client. As shown in Fig. 3.1, the UAV block uses a device's sensor (e.g. front-facing camera) to estimate the user's distance and environmental factors, such as ambient light, to determine a human user's

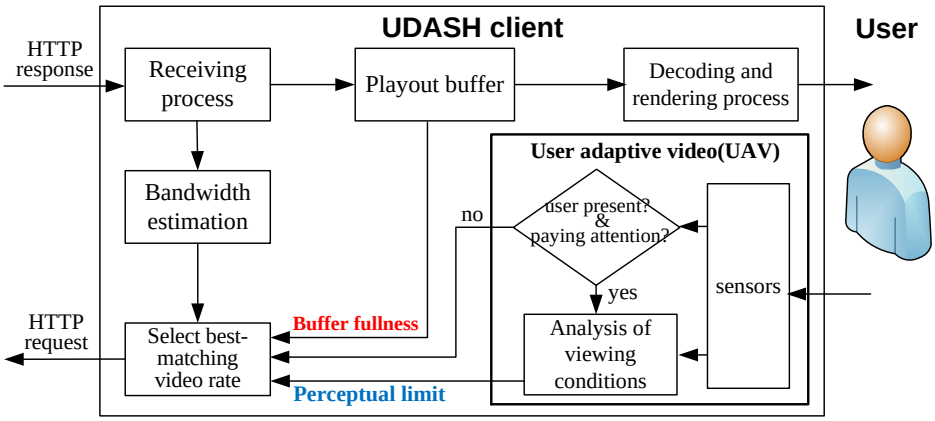


Figure 3.1: The overall UDASH architecture.

Table 3.1: Simulation parameters

Parameters	**value**
Number of seeds	200
Simulation time	100 seconds
Traffic start time	uniform distribution from [10, 30] seconds
Physical Characteristics	802.11n (2.4GHz,TxOP enabled)
TCP receive buffer	65535 bytes

perceptual limit. The perceptual limit describes the highest spatial frequency that can be perceived in an image. The video is converted to a reduced bit rate compared to the highest bit rate representation of the content and results in a bit rate savings without perceivable loss in quality which depends on a given video codec. For example, if the highest quality representation of content is 3.0 Mb/s, a perceptually equivalent representation may be produced at a lower bit rate such as 2.3 Mb/s under specific viewing conditions. Thus UAV can be used to produce visually identical representations at reduced bit rates. This property is explored in the following analysis.

3.2.2 Example Scenarios Showing The Benefits of UDASH

We start exploring the benefits of UDASH by using a simplified scenario in which the network is lightly loaded, and the encoded video only includes two bit rate representations.

3.2.2.1 UAV Helps DASH Traffic

Simulated network. In OPNET 17.1A [71], we create a network consisting of two IEEE 802.11n stations each running a DASH client and an access point connected to a DASH server via the Internet which models a 30 ms one-way delay. The maximum achievable MAC layer throughput for an active client is 5.5 Mb/s, and the value becomes 2.7 Mb/s when two active clients equally share the network with the access point. We call the two identical clients A and B, respectively. The key simulation parameters are in Table 3.1.

Video content, rate selection and perceptual limit. We assume that the set of available bit rates in MPD (Media Presentation Description) are 2.3 Mb/s, 3.0 Mb/s, and the perceptual limit is 2.3 Mb/s. All video segments are 1 second long. The initial buffer size is set to 2 seconds, and the full buffer size is 4 seconds. Each client immediately sends the request for the next segment right after fully receiving the last segment unless the buffer is full (attains 4s). For simplicity, we also assume that each client always requests the highest bit rate with UAV disabled. In other words, a DASH client always requests the segment with bit rate of 3.0 Mb/s, and a UDASH client always requests the segment with bit rate at 2.3 Mb/s. While in practice the viewing conditions will vary, this assumption will provide a reference based on somewhat typical bandwidth savings due to UAV.

Performance metrics. The appearance of interruptions caused by rebuffering events has a significant impact on the QoE in video streaming. It is widely accepted that video interruptions are really annoying for users and the quality of experience can be significantly improved by the lack of interruptions [38]. The presence of a rebuffering event in a simulation trial follows a Bernoulli distribution. The Bernoulli variable takes a value 1 (rebuffering occurred) with probability p and value 0 (no rebuffering occurred) with probability $1 - p$. In this work, we calculate the rebuffering probability p by counting the fraction of clients who experience buffer starvation in each of the 200 simulation runs. We also calculate the 95% confidence interval according to [31, p 393]. In addition to the rebuffering probability, we also use the average total buffering time to characterize the overall time duration during which a video playback has to suspend. This average total buffering time includes both initial buffering time and rebuffering time. The initial buffering time is the one-time waiting

Table 3.2: Simulation results

	No UDASH	UDASH A	UDASH B	Both UDASH
95% Confidence Interval for Rebuffering Probability				
Client A	[0.982,1.0]	[0,0.018]	[0.112,0.218]	[0,0.018]
Client B	[0.982, 1.0]	[0.116,0.224]	[0.0,0.018]	[0,0.018]
Average Total Buffering Time (s)				
Client A	8.89	1.97	2.36	1.87
Client B	8.83	2.31	1.92	1.82

time at the beginning of a video playback. The rebuffering time is the pause time during the middle of a video playback due to buffer starvation.

Simulation results. As shown in Table 3.2, when either client A or B runs UDASH, both of them see improved performance. When both clients use UDASH, the gain is even more significant. Refer to Section 3.2.3 for detailed analysis.

3.2.2.2 UAV Helps Non-DASH TCP Traffic

Simulated network. Based on Section 3.2.2.1, we add one more client which runs an FTP application. The DASH server also serves as the corresponding FTP server. Only FTP download traffic is simulated. To make sure that the bandwidth saved by UAV is fully utilized by the FTP client, we adjust the MAC layer data rate to achieve 10.0 Mb/s MAC layer throughput when only one client is active, or 3.3 Mb/s per-client throughput in the case of 3 active clients.

Video and perceptual limit. Same as Section 3.2.2.1.

Simulation results. The average TCP throughput of the FTP client over 200 random seeds is increased by 36%. Note that there are only two UDASH clients. Higher gain is expected if more UDASH clients use a network with enough capacity to trigger UAV.

3.2.3 Benefit Analysis of UDASH

Traditional DASH clients select the best-matching video bit rate based on the estimated bandwidth and/or the buffer fullness. With UAV, as depicted in Fig. 3.1, UDASH clients consider the user's perceptual limit and may choose a lower bit rate to save bandwidth. The best bit rate at time t_i is given by

$$S(t_i) = \min(F_R(\chi(t_i), \nu(t_i)), \psi_R(t_i)) \tag{3.1}$$

where R denotes the vector of available video bit rates; $F_R(.)$ is the function finding the best affordable video rate in R based on average throughput χ and buffer fullness ν at time t_i; ψ_R is the lowest video rate in R that is above or equal to the perceptual limit at time t_i.

In an overly congested Wi-Fi network, at any time t, for any client, if $F_R(\chi(t), \nu(t)) \leqslant \psi_R(t)$, UAV will not be triggered. In a lightly loaded Wi-Fi network, every DASH client can receive the highest video bit rate. In this case, UAV will not benefit the DASH traffic due to the absence of network induced rebuffering and because DASH users will already be receiving the highest quality representations. However, UAV will still benefit the non-DASH traffic, as shown in Section 3.2.2.2.

However, in a network which is between the above two extremes, the situation is more complicated. As shown in Fig. 3.1, the information whether a user is present and paying attention is conveyed to the DASH rate selection logic. For example, if the user is not paying attention, the buffer size may be extended such that the entire content may be stored in the client. If sufficient time passes before the user continues to pay attention, the client may stop downloading the video and save bandwidth. Some other alternative actions of the client can be proposed, but this is not the focus of this work.

On the other hand, if an active user is detected, UAV starts to work and can offer three direct benefits. The first benefit is to reduce the rebuffering probability. UDASH client can build its own buffer faster, which in turn helps avoid buffer starvation in the future when the network capacity or aggregated traffic load changes. Also, if conventional DASH clients face buffer starvation because of either an aggressive decision (the request of a bit rate much higher than the available bandwidth) or due to the available short-term bandwidth being below the lowest available video bit rate, the saved bandwidth from UAV may compensate for the resource shortfall. The second benefit is to improve the video quality. The saved bandwidth from UAV may help other UDASH or DASH clients get video segments with higher bit rate and hence generally better video quality. The third benefit is to help non-DASH traffic that shares the network, such as the example in Section 3.2.2.2. Section 3.4.3 will confirm that the three benefits can be realized at the same time in a realistic scenario even

though they compete with each other. However, as the network conditions and overall traffic change from time to time, it is hard to analytically characterize how the saved bandwidth will contribute to each of these benefits.

As UAV helps reduce the highest consumed video bit rate, some indirect benefits may be obtained. For example, the decisions to increase quality/bitrate of future segments may be less aggressive. Less quality oscillation and fewer abrupt quality transitions will happen because the gap between the highest and lowest bit rate is smaller.

3.2.4 Conditions When UAV Helps

First of all, an active user must be detected to determine the perceptual limit. Secondly, a network needs to have clients whose available bandwidth may sometimes be higher than the bit rate that corresponds to the perceptual limit so as to enable UAV. In a network where UDASH clients see different and/or dynamic available bandwidth, some clients are able to apply UAV and thereby save bandwidth, while others may use the saved bandwidth to mitigate buffer starvation or enjoy higher bit rate video. If a network is stable and equally shared by all users, UAV either is not triggered or only helps non-DASH TCP traffic (if it exists). Lastly, the full buffer size of a UDASH client may affect how the saved bandwidth will be utilized. A DASH client usually continues requesting the next segment until its buffer is full. The amount of UAV savings used to build the buffer likely increase with the full buffer size.

The above conditions can be easily satisfied in a Wi-Fi network. Next, we describe the implementation of UDASH and evaluate the benefits for a more realistic scenario.

3.3 System Implementation

Our implementation is done in OPNET 17.1A [71].

3.3.1 DASH Protocol

The video bit rate adaptation algorithm is the core component of DASH. Inside the rate adaptation algorithm, an exponentially weighted moving average filter [37] [36] [35] is

widely used for estimating the available bandwidth. The equation is given by

$$T_e(i) = \begin{cases} (1-\delta)T_e(i-1) + \delta T_s(i-1) & i > 1 \\ T_s(i-1) & i = 1 \end{cases} \tag{3.2}$$

where $T_s(i-1)$ is the measured bandwidth in downloading the last segment, $T_e(i-1)$ is the estimated bandwidth for the last iteration, $T_e(i)$ is the estimated bandwidth for the current iteration and δ is the weight given to the last measurement.

Note that UDASH can essentially work with any rate adaptation algorithm. We choose two rate adaptation algorithms using Eq. 3.2 for the purpose of illustration: fixed weight and variable weight.

Fixed weight: $\delta = 0.2$ in our simulation.

Variable weight: $\delta = \frac{1}{1+e^{-k(\rho-P_0)}}$, where $\rho = \frac{|T_s(i-1)-T_e(i-1)|}{T_e(i-1)}$.

The variable weight algorithm uses the logistic function to map the normalized instantaneous estimation error, ρ, to the filter weight, δ, where k and P_0 are parameters of the logistic function. We set $k = 21$ and $P_0 = 0.2$ as in [37].

3.3.2 User-adaptive Video (UAV)

In [90] and [89], UAV is done on the server side. Given the viewing conditions of the receiver, the server uses a pre-processing filter to remove spatial frequencies that are invisible under such conditions. By removing such frequencies the filter reduces the amount of information in the video, therefore leading to more efficient encoding (lower bit rate) without causing any visible alterations of the content. In contrast, in our implementation, we do not use pre-processing filter so everything is done at the client side. Specifically, the parameters on viewing conditions collected by a client will not be sent to the server, but are used to calculate the target video bit rate, which is also called perceptual limit. For example, before calculating the perception limit, the client is about to request the next video segment with quality at 7 which is 3 Mb/s in TABLE 3.3. With UAV, the client determines that the perception limit is 2.3 Mb/s so as to request the segment with quality at 5 in TABLE 3.3.

Table 3.3: Set of available bit rate in MPD

Quality level	Bitrate (Kbps)	Resolution (width x height)
1	400	428x240 (240P)
2	900	640x360 (360P)
3	1400	854x480 (480P)
4	2000	1280x720 (720P_A14[1])
5	2300	1280x720 (720P_A16)
6	2700	1280x720 (720P_A28)
7	3000	1280x720 (720P)

In this case, the saving of bandwidth is 0.7 Mb/s. In [78], the author does not mention the implementation details on the server side.

The MPD profile used in the simulation is listed in Table 3.3. Assuming a median viewing distance and typical viewing conditions (viewing angles is 16 ° as defined in Fig. 1 of [78].) and for the purpose of easier evaluation, the perceptual limit is always set to 2.3 Mb/s. Each video segment is 2 seconds. The initial buffering before playback begins is 4 seconds, and the full buffer size is 20 seconds.

3.3.3 MAC Layer Protocol

Currently, most commercial Wi-Fi products have enabled data rate adaptation (or MCS adaption) at the MAC layer. Data rate adaptation algorithms usually use data rate probes to improve network throughput when the channel quality becomes better. This leads to inevitable packet losses because the probed data rate may be higher than what the true current channel conditions allow. To capture this in our simulation, we implement a MAC layer data rate adaptation algorithm [96] in the Wi-Fi model of OPNET 17.1A.

3.4 Evaluation

3.4.1 Simulation Setup

As shown in Fig. 3.2, we simulate a randomly distributed WLAN network within a 500x500 m^2 area, with 2 FTP clients and a variable number of UDASH clients. With the

[1] A14, A16 and A28 stand for 14 °,16 ° and 28 ° viewing angles as defined in Fig. 1 of [78].

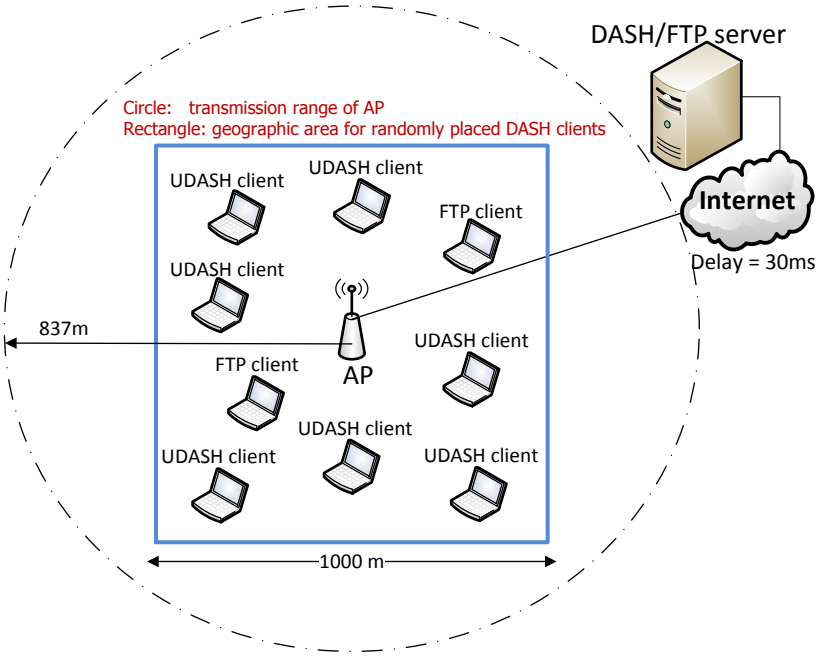


Figure 3.2: Network path with last mile Wi-Fi network.

MAC layer MCS adaptation algorithm implemented, default OPNET 802.11n PHY/MAC layer parameters allow a transmission range of 837 meters. The Internet is simulated between the DASH/FTP server and the Wi-Fi network with a 30ms one-way delay. The other parameters are the same as those in Table 3.1.

3.4.2 Performance Metrics

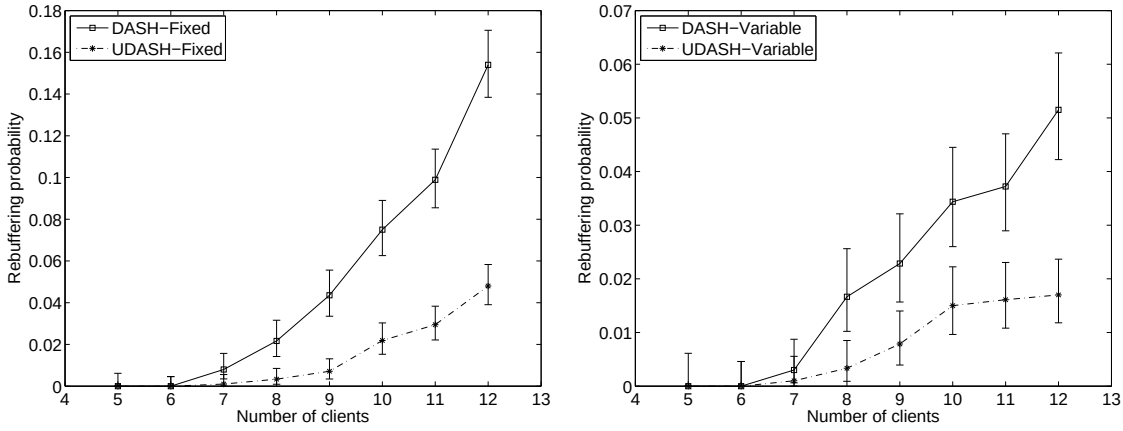
First, we reuse the concept of the rebuffering probability as defined in Section 3.2.2.1.

To show that UDASH may improve user perceived video quality as explained in Section 3.2.3, we also collect the total time during which playback is effectively HD video (720P) or better, that is, the quality level is 4 or above in Table 3.3.

Lastly, we calculate the average TCP throughput for the two FTP clients to show that UAV can also help TCP cross traffic.

3.4.3 Simulation Results

With the setup introduced in Section 3.4.1, we run extensive simulations and present the results here. In general, we see UDASH has significant and consistent improvement over DASH when either rate adaptation algorithm is used. In a lightly loaded Wi-Fi network,



(a) (b)

Figure 3.3: Rebuffering probability

with 5 or 6 clients, where every DASH client can afford the highest video bit rate most of the time, UAV works, but only improves the cross traffic as shown in Fig. 3.5. As the number of clients increases, multiple benefits start appearing in Figures 3.3, 3.4 and 3.5, but they have different behaviors. Firstly, the aggregate UAV savings in each simulation run first increases because more clients do UAV, but then decreases because the savings from each client becomes less due to limited overall capacity. Secondly, the demand (used for reducing rebuffering probability or supporting better video quality) from UDASH clients for the UAV savings increases all the way, because more UDASH clients need help. Since UDASH traffic (assigned to the 802.11e Video Access Category) has priority over FTP traffic (assigned to the Best Effort Access Category), the proportion of the UAV savings consumed by UDASH traffic increases. Thirdly, the UAV savings consumed by FTP clients increases first (the aggregate UAV savings increases but UDASH clients do not need much), then decreases (the aggregate UAV savings is predominantly consumed by UDASH clients). Remember that UDASH clients not only produce but also consume UAV savings at different times.

In terms of rebuffering probability, as shown in Fig. 3.3, the biggest relative reduction is achieved in the case of 8 clients, which is 5x and 6.5x for the algorithms with fixed and variable weights, respectively. In addition, we observe that the rebuffering probability of UDASH clients climbs more slowly than DASH clients as the total of number of clients increases, which shows that UDASH clients enable better scalability.

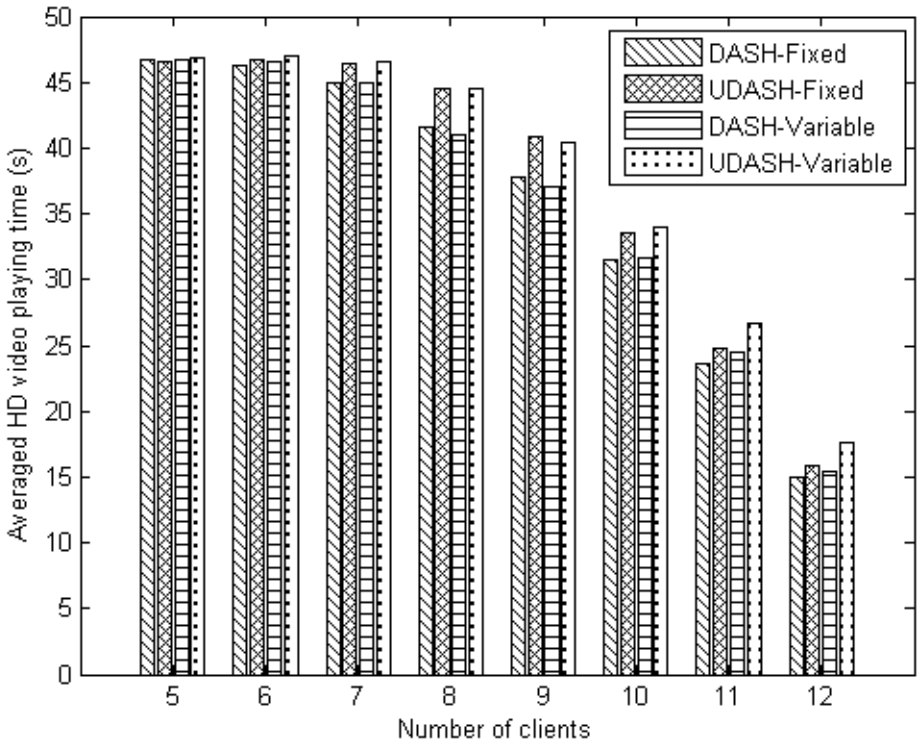
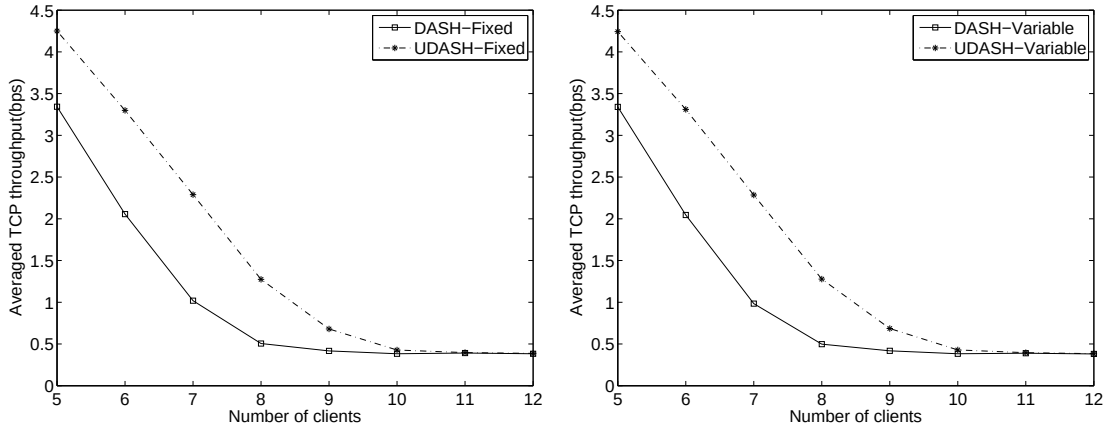


Figure 3.4: Total playing time when the playback is HD quality

Besides rebuffering probability, as Section 3.2.3 explains, the bandwidth saved by UAV offers several other benefits. One of the benefits, improving video quality, is subject to many dynamic factors. For example, the bandwidth saved by UAV may be used by another client to prevent buffer starvation rather than to allow a different client to request a higher quality segment. Fig. 3.4 shows the total time the playback is of HD quality. Due to those dynamic factors, the gain of UDASH over DASH client can be from 0.3% to 15%, where the highest gain is achieved in the case of 12 clients using the algorithm with a variable weight.

Fig. 3.5 presents the throughput improvement of the TCP cross traffic when UDASH is used. According to the earlier explanation in this section, the gain increases first and then decreases as the total number of clients increases. The biggest gain 2.5x is achieved in the case of 8 clients which is the turning point.



(a) (b)

Figure 3.5: Average TCP throughput for cross FTP traffic

3.5 Summary

In this chapter, we leverage a new user-aware video coding technique called UAV to improve the performance of DASH systems and non-DASH TCP traffic in a Wi-Fi network. UAV is an effective enhancement that can be applied to any DASH rate adaptation algorithm. We design a general architecture termed UDASH to make use of UAV in DASH clients, and explore the conditions when and how UDASH can provide benefits. The simulation results of UDASH show significant improvement over DASH, by considerably reducing the rebuffering probability, increasing the fraction of time of HD or better quality video, and improving the throughput of TCP cross traffic.

Chapter 4

PROTOCOL ADAPTIVE WIRELESS VIDEO TELECONFERENCING

4.1 Introduction and Related Work

Wi-Fi (IEEE 802.11) networks have been widely deployed and the adoption is still fast growing. However, in real-time video applications, such as video teleconferencing over time-varying, error-prone and bandwidth-fluctuating channels, Wi-Fi networks still face many challenges. A typical real-time video system may have Wi-Fi link(s) (shared with multiple competing peers) on the edge and Internet in the core as shown in Fig. 4.1, where the video sender transmits encoded video data to the video receiver for decoding and playback. Standard video codecs (such as H.264 and VP8) exploit the spatial and temporal redundancy in uncompressed video to achieve a high compression ratio, which, however, makes compressed video very sensitive to transmission errors. Packet losses due to transmission errors often lead to serious video quality degradation, like artifacts in the decoded video, which affects not only the current frame, but also subsequent frames because of error propagation resulted from the use of prediction from previous frames. Some error concealment techniques can help stop error propagation, for example frame copy that is used in WebRTC [92] where a frame that cannot be correctly decoded is replaced by the last correctly decoded frame. This however causes video freezes. Application layer Forward Error Correction (FEC) is a commonly used scheme for error protection. However, it introduces extra complexity, overhead and delay, which is undesirable to applications with stringent delay constraints like video teleconferencing. In Section 6.2, we will investigate the efficiency problem when FEC is used in our WebRTC-based testbench. One straightforward and preferable solution is to develop a lightweight mechanism to reduce packet losses caused by channel errors, which is the focus of this work.

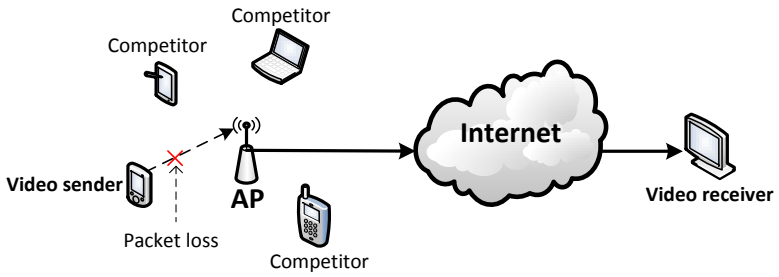


Figure 4.1: A typical real-time video system

In wireless networks, path loss, shadowing, fading and interference cause packet losses. Packet losses due to these reasons are classified as channel-caused losses. Note that such packet losses, which occur when the distance between a wireless station and an Access Point (AP) increases or when obstacles move temporarily between the station and the AP, are very frequent in Wi-Fi networks. When the channel is heavily loaded with traffic from multiple contending stations, and the available bandwidth shared by all stations is not enough to accommodate all incoming traffic, the network is congested. If congestion persists, packet losses due to buffer overflow will occur. In a congested network with many active contending stations, collision induced packet losses may also increase significantly. Packet losses due to these reasons are classified as congestion-caused losses.

In this work, we exploit the fact that packet losses on a Wi-Fi link can be inferred by not receiving a positive acknowledgement (ACK) packet after reaching the retry limit. We propose that if the packet loss is channel-caused, the MAC layer grants more transmit opportunities by temporarily increasing the retry limit. If the packet loss is congestion-caused, the MAC layer does nothing in order not to conceal the packet loss from higher layer's congestion control algorithms. Our approach implicitly assist existing congestion control mechanisms at higher layers. The dominant transport layer protocols are TCP and UDP. TCP is known for using congestion control. Video traffic accounts for a significant share of UDP traffic and congestion control at the RTP layer is recommended [41] and generally implemented in the newer video telephony systems such as WebRTC. For widely

used higher-layer congestion control mechanisms, packet losses are interpreted as indications of congestion which is not always correct especially in a wireless environment, and this may lead to unnecessary reduction in the data sending rate. To mitigate this problem, it is important to reduce channel-caused losses, which requires reliable differentiation between channel-caused losses and congestion-caused losses which in turn necessitates another task of this work: congestion detection in a Wi-Fi network.

Congestion detection in wireless networks has been extensively studied [60]. However, most of the proposed protocols make use of the detection results in order to perform congestion control [45, 87, 77, 5] or rate adaptation [54, 4] at the MAC layer. In [97], congestion status is part of the objective function for optimizing Packet-level FEC (PFEC) and packet scheduling. In [104], the TCP sender generates and sends a special Resource Discovery (RD) packet which travels a round trip and brings back information on the end-to-end capacity to help the sender adjust the sending window size accordingly. In [15], the authors propose to adapt the backoff window size to the current network contention level. In [56], a joint adaptation of link rate and backoff contention window is proposed to improve the performance of 802.11 multi-rate network. In contrast to the aforementioned prior work, in this work we leave the job of congestion control up to upper layers, like the transport layer or the RTP layer, and make use of the detection results in a very different way, which is adapting the retry limit to the congestion level in the wireless channel to indirectly assist congestion control mechanisms at higher layers.

The retry limit optimization has also been extensively studied in the literature. Representative work includes [46, 16, 81, 17, 69]. In [81], the authors propose adjusting the retry limit according to the MAC layer data rate. However, in Wi-Fi a higher data rate does not necessarily indicate better channel quality which often implies low likelihood of channel-caused losses because hidden terminal interference may still exist or the MAC layer rate adaptation algorithm may try a higher data rate that the current channel condition cannot sustain in order to potentially improve the overall throughput. Even though these mechanisms are shown to be effective since they require either major modifications in the already well-established IEEE 802.11 standard or cross-layer signaling, their applicability is limited.

Moreover, most of those prior work uses simulation to validate the proposed solutions. Our solution is implemented in a real testbench that allows for a more realistic evaluation and demonstration.

Note that even though the example networks evaluated in this dissertation are infrastructure based Wi-Fi networks, any other wireless networks [25] [98] [94] [93] that rely on retransmissions may also be applicable.

4.2 Problem Analysis and Motivation

We develop a testbench shown in Fig. 4.2 to perform experiments and identify the problems that we are about to solve.

4.2.1 Testbench Setup

As shown in Fig. 4.2, the testbench consists of three directly connected laptops via Ethernet cable, where Laptop A and Laptop B run WebRTC (version 6475) [92] based video teleconferencing while the third laptop called the OPNET Laptop runs OPNET with system-in-the-loop (SITL) [72] functionality. SITL allows real WebRTC packets (including RTP packets and RTCP packets) to enter the OPNET Laptop from Laptop A and Laptop B. In this way, we can emulate the delivery of these packets through various communication networks with high fidelity. Fig. 4.3 shows the screenshot of the emulated OPNET scenario.

Inside the OPNET Laptop of Fig. 4.2, a typical and realistic network scenario is created, which represents an important lossy communication network that consists of the Internet in the core and Wi-Fi links on the edge. Real WebRTC traffic from Laptop A and B via Ethernet interfaces arrives at SITL_1 and SITL_2 then STA_A and STA_B. We can consider STA_A and STA_B as virtual mapping nodes of Laptop A and B, respectively. Between STA_A and STA_B, a Wi-FI network and the Internet are simulated. A detailed description about the testbench is summarized shortly. Without loss of generality, this work only investigates packet losses that happen when STA_A sends WebRTC video packets to AP, and implement our algorithms in STA_A to improve the overall performance of the

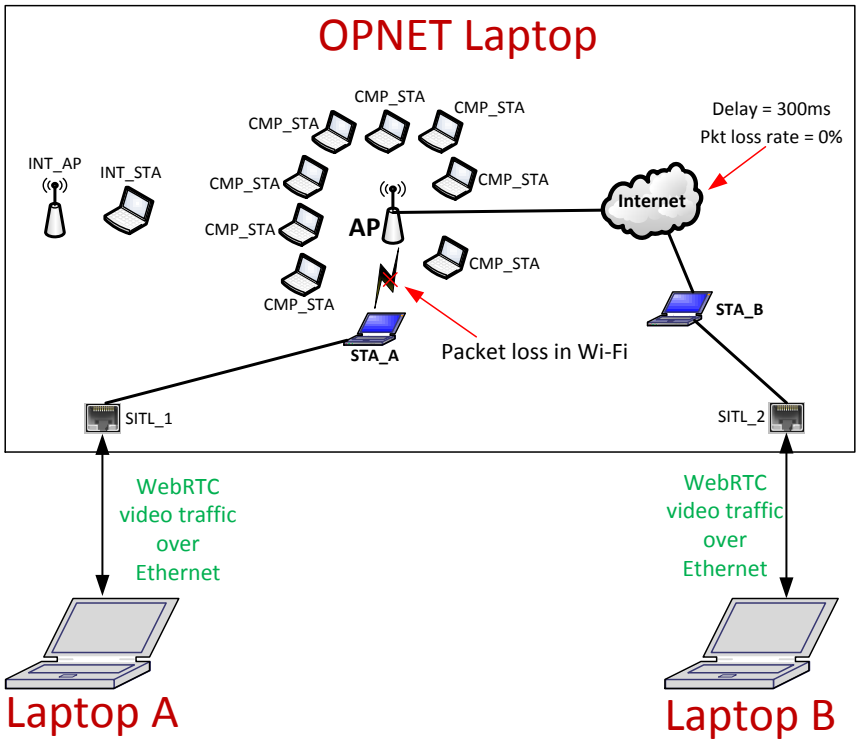


Figure 4.2: WebRTC generated real video traffic passes through an OPNET-based network emulator

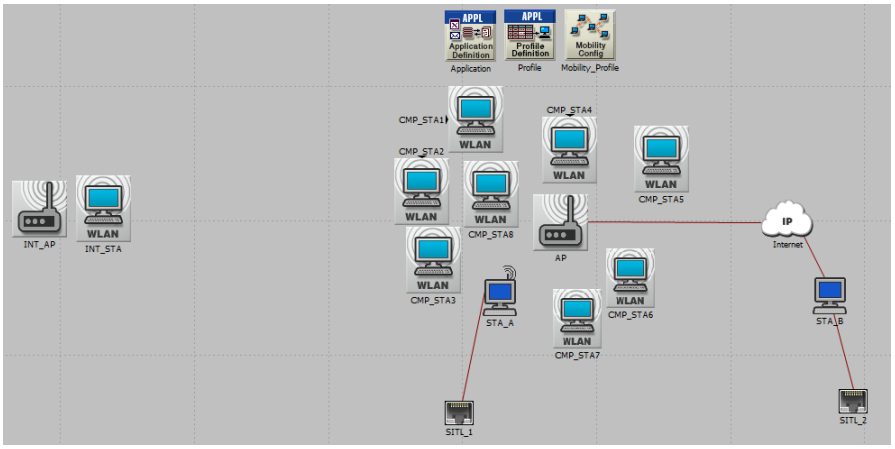


Figure 4.3: Screenshot of the emulated OPNET scenario

video flow from Laptop A to Laptop B. Simply put, our approach improves the video sender who has a lossy Wi-Fi link.

Summary of the testbench:

PHY and MAC layer: HT PHY at 2.4 GHz with IEEE 802.11n. PHY Data Rate is 65 Mbps. All other parameters use default values in OPNET 17.1.A. Since WebRTC video traffic uses Best Effort Access Category by default, we let all other traffic (cross traffic and hidden terminal traffic) in the Testbench use Best Effort as well.

Communication path: Laptop A via Ethernet ↔ SITL_1 ↔ STA_A ↔ AP ↔ Internet ↔ STA_B ↔ SITL_2 ↔ Laptop B via Ethernet.

Main Wi-Fi network: Consists of STA_A, AP and 8 CMP_STA with 802.11n 2.4G Hz. The AP is at a fixed position, while STA_A and CMP_STA are randomly placed (But the distance constraints required to create a hidden terminal network are satisfied).

Congestion: Adjust the cross traffic between CMP_STA and AP to create different levels of congestion.

Hidden terminal Wi-Fi network: It consists of INT_STA and INT_AP with fixed locations. INT_STA and STA_A cannot hear each other, but AP is within interference range of INT_STA. Since INT_AP cannot be interfered by anyone in the Main Wi-Fi network, INT_STA servers as a hidden terminal to STA_A but not vice versa.

Traffic definition: All three types of traffic (WebRTC, cross traffic and hidden terminal traffic) are UDP-based video traffic, but some differences are there. Hidden terminal traffic uses constant bit rate after it gets started. Cross traffic also uses constant bit rate for easier control but the rates are different in different time periods in order to create different levels of congestion. WebRTC starts from a minimum target bit rate (50 kbps) and then gradually evolves as explained in Section 4.2.2. The default maximum target bit rate is around 4 Mbps. The target bit rate goes into the video encoder which generates a final video bit rate generally not higher than the target bit rate. Another difference is that cross traffic and hidden terminal traffic do not perform congestion control but WebRTC uses GCC (refer to Section 4.2.2 for more detail) for congestion control.

4.2.2 Background of WebRTC

WebRTC uses the Google Congestion Control (GCC) algorithm [61] to perform congestion control, which is composed of two parts: the receiver-side controller computes the rate A_r and sends it to the sender; the sender-side controller computes the target sending bit rate A_s that cannot exceed A_r. Specifically the sender-side controller updates the maximum allowable sending rate $A_s(t_k)$ every time t_k the k-th RTCP report message carrying a fraction of lost packets $f_l(t_k)$ arrives at the sender. Usually but not always, the RTCP report message also includes a Receiver Estimated Maximum Bitrate (REMB) message carrying an REMB value A_r. Note that the fraction of lost packets $f_l(t_k)$ is calculated before FEC recovery is applied (if FEC is turned on). As described in [82], the RTCP reports include the fraction of lost packets $f_l(t_k)$ observed by the receiver, while REMB is based on the average delay jitter calculated by the receiver. The sender uses $f_l(t_k)$ to compute the sending rate $A_s(t_k)$, measured in kbps, according to the following equation:

$$A_s(t_k) = \begin{cases} max\{\ X(t_k), A_s(t_{k-1})(1 - 0.5 f_l(t_k))\ \} & f_l(t_k) > 0.1 \\ 1.08 \min_{t \in (t_k - \Delta, t_k)} A_s(t) + 1kbps & f_l(t_k) < 0.02 \\ A_s(t_{k-1}) & otherwise \end{cases} \tag{4.1}$$

where $X(t_k)$ is the TCP friendly rate control (TFRC) rate [42]. The logic behind Eq. 4.1 is: 1) when the fraction of lost packets is considered low ($0.02 \leq f_l(t_k) \leq 0.1$), the data sending rate is kept constant, 2) if the fraction of lost packets is considered high ($f_l(t_k) > 0.1$), the data sending rate is multiplicatively decreased (it is configured that the rate will not be decreased more than once in the last (0.3 + RTT) seconds, where RTT is the round trip time reported by the receiver), but not below $X(t_k)$, 3) when the fraction lost is considered negligible ($f_l(t_k) < 0.02$), the rate is adjusted to be 108% of the minimal value of A_s in the last Δ second (Δ is pre-configured as 1 by default). After $A_s(t_k)$ is computed from Eq. 4.1, the value of $A_s(t_k)$ is further updated as $A_s(t_k) \leftarrow min(A_s(t_k), the\ last\ received\ A_r)$, to ensure that $A_s(t_k)$ never exceeds the last received value of A_r carried in the REMB message.

WebRTC utilizes both proactive and reactive packet loss mitigation methods [80]. The proactive method used by WebRTC is packet-level FEC. FEC adds redundancy to achieve

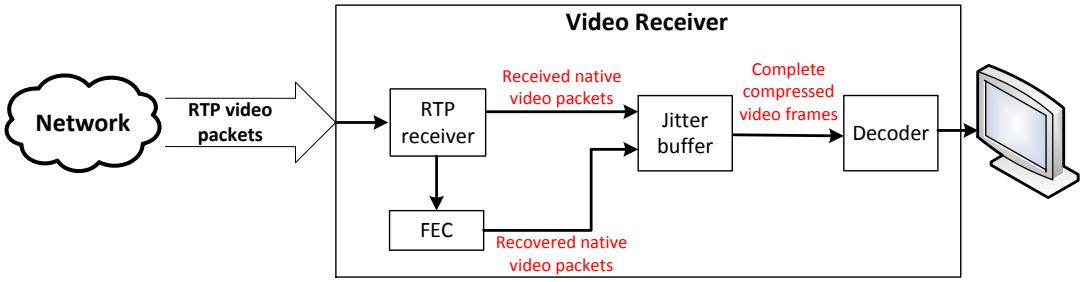


Figure 4.4: Video bit rate received by the video receiver

packet loss mitigation. In Section 4.2.4, we will explain the efficiency problem caused by the use of FEC. However, if not using FEC, the reactive packet loss mitigation method alone is not sufficient to mitigate packet losses. In Section 4.2.5, we will explain what is the reactive packet loss mitigation method used in WebRTC and why it is not good enough. Next we introduce the motivation of our proposed scheme as a better solution in Section 4.2.6.

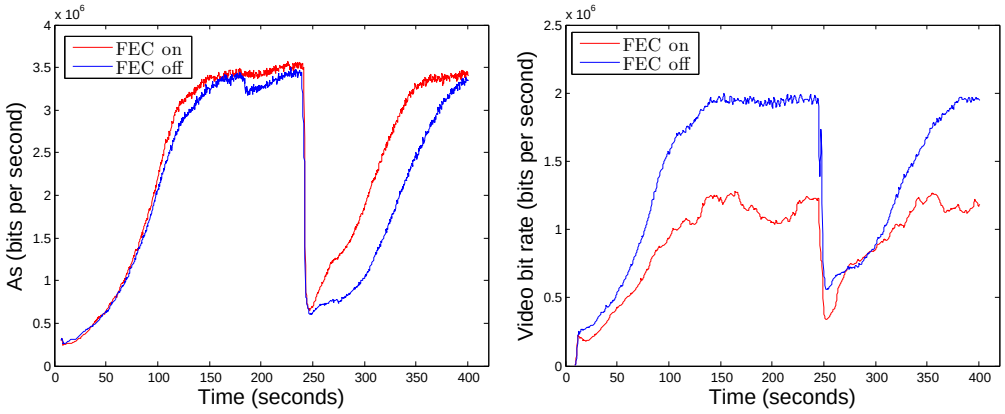
4.2.3 Experimental Setup

We combine different network conditions (hidden terminal traffic, competing traffic with both high and medium load) and carry out a large number of experiments, each lasting 400 seconds. The hidden terminal traffic starts at the beginning (0 second) until the end of each experiment. Competing traffic is active within two time periods, [180 sec, 210 sec] and [240 sec, 250 sec], where the former time period has competing traffic of medium load and the latter has competing traffic of high load.

In this chapter, all experimental results are collected by running emulations with the testbench and experimental setup described in Section 4.2.1 and 4.2.3 and are the average of 10 emulation runs so that we can do a fair comparison on the experimental results before and after using our proposed scheme.

4.2.4 Lower Received Video Bit Rate due to Packet Losses

In WebRTC, the application-layer FEC adapts the redundancy to the packet loss rate. Given the same target sending bit rate A_s calculated in Section 4.2.2, the higher the fraction of lost packets $f_l(t_k)$ reported, the higher the portion of A_s will be assigned to transmit FEC



(a) (b)

Figure 4.5: When FEC is turned On and OFF: (a) Target sending rate A_s at Laptop A (b) Received video bit rate at Laptop B

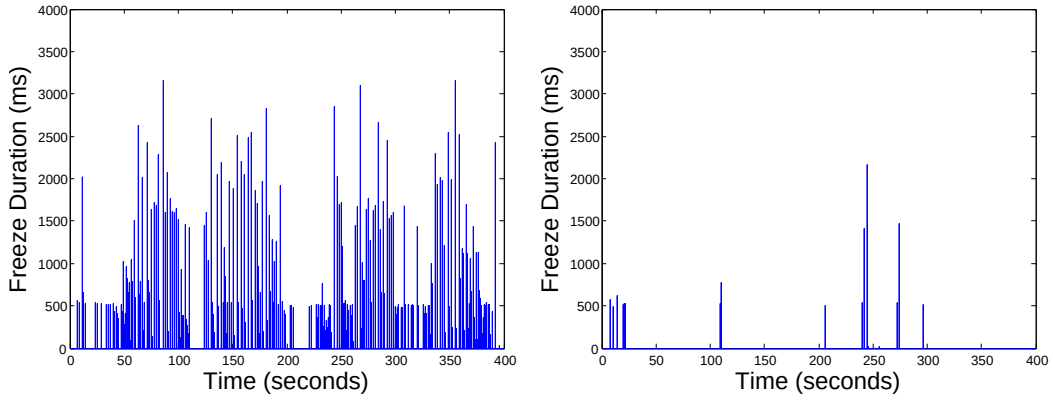
redundant packets, leaving a smaller portion of A_s for sending native video packets and consequently a lower received video bit rate, hence a lower video quality. As shown in Fig. 4.4, we define the received video bit rate as the arrival bit rate to the video decoder, which equals to the total size of complete compressed video frames received by the decoder per second. Due to packet losses in the end-to-end network, the video receiver relies on FEC to recover missing native video packets and fill the gap in the Jitter buffer. When all packets of a video frame arrive at the Jitter buffer, the video frame is sent to the decoder.

By following Section 4.2.3, we do experiments with FEC turned on or off, and get results in Fig. 4.5, where (a) shows the target bit rate A_s calculated at the video sender and (b) shows the received video bit rate at the video receiver. From Fig. 4.5, we observe that:

a. The maximum value of A_s is around 3.5 Mbps, while the maximum value of received video bit rate is only around 2Mbps, even when FEC is turned off. This is because WebRTC sets a upper limit which is 2 Mbps for the video encoder, no matter how big A_s is.

b. At the time of 240 seconds when congestion happens, congestion control algorithms works well and A_s drops to 0.6 Mbps in both curves.

c. Except the two transient time periods, [0 sec, 100 sec] and [240 sec, 350 sec], although the sending rate A_s are almost the same, the received video bit rate when FEC is turned off is 70% higher than the case when FEC is turned on, showing significant overhead



(a) (b)

Figure 4.6: Video freeze duration on Laptop B: (a) FEC is turned off (b) FEC is turned on

of FEC.

Due to the efficiency problem explained above, plus extra complexity and delay from FEC, in some commercialized WebRTC products, like Chrome browser, FEC in WebRTC is disabled. In Section 4.2.5, we will show that however, turning off FEC will bring about another problem.

4.2.5 Video Freezes due to Packet Losses

As explained in Section 4.2.2, WebRTC uses both proactive and reactive packet loss mitigation methods. The proactive method is application-level FEC. The reactive approach is based on an end-to-end feedback to request RTP layer retransmissions from the video sender, which is ineffective in the case of large round-trip-time (RTT). In our testbench setup where there is a 300 ms Internet delay, when packet loss happens on the Wi-Fi link from STA_A to AP, the receiver has to wait 600 ms (RTT may range from 50 ms up to 700 ms [23]) before a retransmission from the sender can be received, and this will make the playout buffer delay insufficient to conceal the packet loss. As a result, with the default error concealment technique of WebRTC, a type of frame copy where the video freezes at the last perfectly recovered frame until the lost frame is recovered, excessive video freezes will appear in the decoded video.

Fig. 4.6 shows the video freezes in a single experiment when FEC is turned off or on, where the freeze duration is calculated by subtracting the time when the next frame is actually displayed from the original scheduled time of the next frame. A significant percentage of the video freezes are close to 600 ms (not exactly 600 ms due to the dynamic value of the playout buffer delay) as shown in Fig. 4.6 because application layer retransmissions (the reactive approach described above) rely on end-to-end feedback which takes about an RTT (600ms in the experiment). If application layer retransmissions also get lost, the actual video freezes can be multiple times of one RTT. Although the use of FEC can significantly reduce the video freezes, as shown earlier, it can significantly reduce the received video bit rate.

4.2.6 Motivation

In summary, channel-caused packet losses on an edge Wi-Fi link at the video sender (such as the link from STA_A to AP in Fig. 4.2) poses a dilemma to WebRTC. If FEC is turned on, the almost 70% overhead can lead to lower video quality at the video receiver; if FEC is turned off, like some commercial products do, excessive video freezes would happen at the video receiver if the RTT is not negligible.

Supposing the differentiation between channel-caused losses and congestion-caused losses has already been perfectly done (we will present an imperfect but effective algorithm in Section 4.3), there are two candidate solutions based on our analysis:

1. (Cross layer approach) The MAC layer of the video sender detects a packet loss by not successfully receiving a positive acknowledgement within the maximum allowed number of retransmissions. On behalf of the video receiver, inside the video sender the MAC layer sends a spoofed negative acknowledgement(NACK) to RTP layer to trigger an RTP layer retransmission in time, which effectively reduces the RTT and makes the receiver possibly unaffected by the packet loss.
2. (MAC layer only approach) The MAC layer adaptation grants the packet which will experience an imminent loss (i.e., the packet is about to be discarded and considered as lost by a higher-layer protocol after reaching the retry limit) higher priority or more opportunities to prevent the loss.

The first solution relies on deep packet inspection of the lost packet in order to find the associated sequence number so that the spoofed NACK can include the sequence number and tell the video sender which packet gets lost. In cases when Real-time Transport Protocol (SRTP) and Transport Layer Security (TLS) are used, there are some encryption problems that have to be taken care of. For more details of this approach, please refer to Section 4.4.

The second solution, unlike EDCA in 802.11e, which depends on the type of the packets (e.g., voice vs. video) and grant different priorities to different Access Categories, ignores the type of the packets and treats every packet equally to avoid relying on cross layer assistance. The rationale is that in the case of non-congestion, network resource anyway is under-utilized, granting more transmission opportunities only utilizes otherwise wasted network bandwidth. There are still some challenges, like how to guarantee fairness in contention, how to avoid incurring congestion if granting too many opportunities, and how to make sure higher priority or more opportunities indeed prevent loss. We will answer these questions partly as follows and partly in Section 4.5.1. For more details of this approach, please refer to Section 4.5.

Basically, we have two paramters to do MAC layer adaptation: Retry limit and contention window (CW). The duration of CW is used for resolving contention when several stations are competing to access the same channel. So a change in the CW of a STA means a change in the medium access priority, which affects fairness and is undesirable for other stations that do not use our mechanism. In contrast, changing the retry limit does not affect each single contention and we can still maintain equal channel access success probabilities. In a lightly loaded network, increased retransmissions can better utilize the network resources; in a heavily loaded network, the packet loss is classified as contention-caused loss, and retry limit will not be increased. Regarding change in the retry limit, there are open challenges, which will be discussed in Section 4.5.1.

For comparisons between these two approaches, please see Section 4.6.

4.3 Congestion Detection

4.3.1 Congestion Metric

For each Wi-Fi station, a single queue is typically used for traffic with the same priority (called an access category in 802.11e), and the queue length at any time instant t_0 is given by:

$$Q_{len}(t_0) = \int_0^{t_0} (Ar(t) - Deliv(t) - Disc(t))\,\mathrm{d}t \tag{4.2}$$

where $Ar(t)$ is the data arrival rate from the IP layer at time t; $Deliv(t)$ is the delivery rate (number of successfully delivered bits per unit time), determined by the available shared bandwidth; $Disc(t)$ is the discard rate (due to reaching the retry limit), determined by the retry limit, the random back-off time and the channel occupancy by contending peers.

From a quantitative perspective, a congested network is defined as one of which the aggregate data arrival rate is persistently higher than the network capacity. Similarly, a congested Wi-Fi station can be defined as one of which the data arrival rate is more than the share of bandwidth available to that station, i.e., $\int_0^{t_0}(Ar(t) - Deliv(t))dt$ is greater than a positive threshold. From Eq. 4.2, we note that the queue length does not fully characterize congestion, as a successfully delivered packet is treated the same way as a discarded packet.

Now we consider how to get the average transmit delay of a discarded MPDU, say, TD, and then use its reciprocal $1/TD$ to obtain an upper bound on $Disc(t)$. Here the transmit delay is defined to be the time interval from the time the MPDU reaching the head of its MAC queue for transmission, to the time an acknowledgement for this packet is received or discarded upon reaching its retry limit. In other words, the queueing delay due to waiting for the service of previous packets to be completed is not included. As specified in the 802.11 standard, upon a packet loss, the Wi-Fi station randomly chooses a backoff time and retransmits the packet after the backoff time expires. However, other contending stations may occupy the channel during the backoff time and force the backoff timer to freeze until the channel is sensed idle again after a DIFS/AIFS period. Therefore, the actual length of the backoff period can be much longer than the original randomly picked backoff time. Borrowing the concept of conditional collision probability and the equation from [14] and

[62], and assuming that the channel is occupied by other contending stations with a constant and independent probability p at each time slot, it can be shown that the average transmit delay of a discarded MPDU is:

$$\overline{TD(R,L)} = \sum_{i=1}^{R} \left(\frac{min(2^{i-1} \cdot (CW_{min}+1) - 1, CW_{max})}{2} \times (p \cdot T(L) + aSlotTime) + T(L) \right) \quad (4.3)$$

where $T(L_i) = T_{DATA}(L_i) + aSIFSTime + T_{ACK} + aDIFSTime$ is the transmission time for sending an L-byte long packet. p is the conditional collision probability [14]. R is the retry limit with a default value 7 (including the first transmission and maximum 6 retransmissions). L is the length of a packet. Assume that the PHY data rate = 65 Mbps, L = 1224 bytes, ACK = 76 bytes, PLCP overhead = 40 us, aSlotTime = 9 us, aSIFSTime = 16 us, and aDIFSTime = 34 us. Then T(1224) = 0.09 ms + 0.16 ms = 0.25 ms.

In a legacy network with $CW_{min} = 15$ and $CW_{max} = 1023$, assuming $p = 0.1$, we have $\overline{TD(7,1224)} = 36.175ms$. In an EDCA network, for the video AC with $CW_{min} = 7$ and $CW_{max} = 15$, assuming $p = 0.1$, we have $\overline{TD(7,1224)} = 3.399ms$.

Apparently, a packet of the video access category in an EDCA netowrk ($3.399ms$) takes much less time before being discarded than the time ($36.175ms$) taken by a packet in legacy network. As a result, by the upper bound $Disc(t) \leq 1/\overline{TD(R,L)}$ (assuming each discarded packet takes the same average delay), the $Disc(t)$ in Eq. 4.2 could take a larger value that is non-negligible in an EDCA network. The observation inspired us to find a better congestion metric other than the queue length to make our mechanism work in a broad range of networks. To accommodate this purpose, we define the excess data rate as follows:

$$EDR_{\tau}^{w}(t) \leftarrow [Ar_{\tau}^{w}(t) - Deliv_{\tau}^{w}(t)]^{+}/(w\tau) \quad (4.4)$$

where $[x]^{+} = max(x, 0)$. $Ar_{\tau}^{w}(t)$ is the total amount of arrival data (aggregate size of arrival MSDUs) in bits and $Deliv_{\tau}^{w}(t)$is the total amount of successfully delivered data (aggregate size of successful MPDUs) in bits during the time period $[t-w\tau, t)$. τ is the sampling interval. To reduce short-term fluctuation, a sliding window w is introduced, which represents the

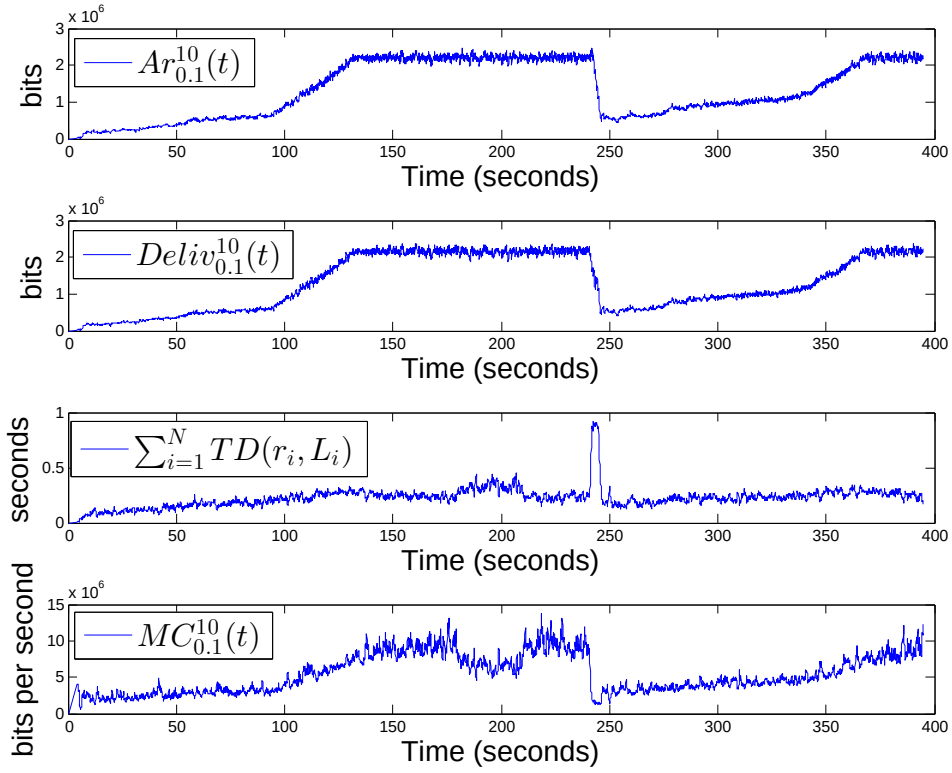


Figure 4.7: Realtime calculations of Eq. 4.6 in WebRTC-based video teleconferencing experiments

number of time intervals (each of τ long) so that all statistics falling in and only in the time period $[t - w\tau, t)$ contribute to the calculation in the Eq. 4.4. In other words, τ determines the update granularity, and w determines the update smoothness. For example, if $\tau = 0.1$, $w = 10$, $EDR^{10}_{0.1}(t)$ stands for the average excessive data rate (measured in bits per second) during the most recent $0.1 \times 10 = 1$ second.

Given the same excess data rate, different networks may experience different levels of congestion because the available bandwidth may be different. To manifest the severity of congestion, we divide excess data rate by the estimated MAC layer capacity to get the congestion level

$$CL(t) = EDR^{w}_{\tau}(t)/MC^{w}_{\tau}(t) \tag{4.5}$$

where the estimated MAC layer capacity $MC^{w}_{\tau}(t)$ is given by

$$MC^{w}_{\tau}(t) = Deliv^{w}_{\tau}(t) \Big/ \sum_{i=1}^{N} TD(r_i, L_i) \tag{4.6}$$

Where $TD(r_i, L_i)$ is the actually measured transmit delay for the i-th transmitted packet. i stands for the i-th packet and r_i is the number of transmissions (including the first transmission and the subsequent retransmissions) that the i-th packet performes. L_i is the packet length of the i-th packet. The rationale behind Eq. 4.6 is: assuming during the most recent time window $(t - w\tau, t)$, a station transmits N fresh (excluding retransmissions) packets, the available MAC layer capacity for this station is estimated as the total successfully delivered data (measured in bits) divided by the total amount of transmit delay experienced.

4.3.2 Evaluation of the congestion metric

We propose to calculate the value of Eq. 4.5 in realtime to detect the congestion level in the network. As shown in Eq. 4.5, the ratio of $EDR^w_\tau(t)$ to $MC^w_\tau(t)$ determines the congestion metric. We start to analyze the estimated MAC layer capacity in Eq. 4.6, which is used in Eq. 4.5.

Fig. 4.7, from top to bottom, shows the curves for $Ar^{10}_{0.1}(t)$, $Deliv^{10}_{0.1}(t)$, $\sum_{i=1}^{N} TD(r_i, L_i)$ and $MC^{10}_{0.1}(t)$ respectively. Two interesting observations of the curve of $MC^{10}_{0.1}(t)$ can be noticed:

a. Large fluctuations of $MC^{10}_{0.1}(t)$ at the very beginning.

b. $MC^{10}_{0.1}(t)$, although fluctuating over small time scales, follows a trend of increasing when the traffic arrival rate increases in the time interval [0, 140 sec] during which there is no cross traffic.

The reason for observation **a** is that the sample size is too small(small number of packets, and very short time duration) at the beginning, and we will explain later that this will not affect the correctness of Eq. 4.5. We explain observation **b** using the theorem shown below. Since an RTP layer video packet is typically much bigger than the Maximum Transmission Unit (MTU) size of the Ethernet, it is usually fragmented into multiple IP packets, all of which have approximately the same size as the MTU size, except the last one which carries the remainder. Thus, in the theorem below we assume that every packet at the MAC layer has the same length.

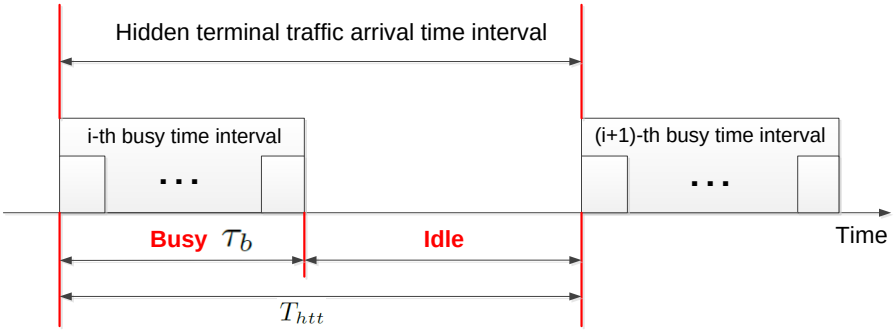


Figure 4.8: Network resource occupation of hidden terminal traffic

Theorem 1. (MAC layer capacity estimation)

Given that every packet has the same length, i.e., $L_i = L$, for all i, $TD(R, L) > \tau_b$, where R is the retry limit and τ_b is the busy time interval of the hidden terminal traffic as shown in Fig. 4.8, in a Wi-Fi network with no competing traffic, we have:

(i) $MC_\tau^w(t)$ monotonically increasing with $Ar(t)$ but slower than $Ar(t)$;

(ii) $MC_\tau^w(t)$ is bounded as follows:

$$\big(Ar(t)/(Ar(t) + \rho C)\big)C \leqslant MC_\tau^w(t) \leqslant C \tag{4.7}$$

where ρ is the channel utilization of the hidden terminal, and C is the MAC layer capacity if there is no hidden terminal traffic.

(iii) if further assuming that $Ar(t)$ follows the Poisson traffic model [13], then

$$MC_\tau^w(t) = \Big(\lambda L/(\lambda L + \rho C(\lambda\tau_b + e^{-\lambda\tau_b} - 1)/\lambda\tau_b)\Big)C \tag{4.8}$$

where λ is the packet arrival rate in the Poisson traffic model.

Proof. **Part (i):** Since there is no cross traffic, collision with hidden terminal traffic is the only reason for transmission failure. The assumption that $TD(R, L) > \tau_b$ means that the transmit delay of one packet is so big that all subsequent packets are forced to be transmitted in an idle time interval as shown in Fig. 4.8. Therefore there is at most one packet that will be transmitted during a busy time interval of the hidden terminal within each hidden terminal traffic arrival time interval T_{htt} (in Fig. 4.8). Assuming $M(M \geqslant 1)$ fresh MPDUs

(excluding retransmissions) are transmitted over the network within one T_{htt}, the aggregate transmit delay is given by

$$\sum_{i=1}^{M} TD(r_i, L) = \sum_{i=2}^{M} TD(1, L) + TD(r_1, L) \tag{4.9}$$

where $TD(r_1, L)$ means that the first fresh MPDU is possibly transmitted during a busy time interval, and the number of retry count is r_1 ($r_1 \in [1, 7]$, assume retry limit is 7.) which depends on at which time instant the packet arrives in a busy time period. Each of the other $(M-1)$ packets which must arrive during an idle time period, only needs to be transmitted once ($TD(1, L)$), where 1 stands for only one transmission will be taken for each one of the $(M-1)$ packets. Since $TD(r_1, L)$ is usually much bigger than $TD(1, L)$ due to the exponential growth of the contention window (CW) of successive retransmissions, $\sum_{i=1}^{M} TD(r_i, L)$ would increase as M increases but at a pace slower than M. Now going back to Eq. 4.6, as all M packets gets delivered (the first one may be delayed but finally falls within an idle interval and also goes through), the total delivered bits $Deliv(t)$ grow at the same speed of M. By Eq. 4.6, the $MC_{\tau}^{w}(t)$ monotonically increases but at a pace (in percentage per unit time) slower than $Ar(t)$ does, where $Ar(t)$ is the total amount of data in the M packets.

Part (ii): Considering a time interval $(t - w\tau, t)$ longer than a single T_{htt} at any time t, if there are N fresh MPDUs of data traffic and Q hidden terminal traffic arrival time intervals ($Q \times T_{htt}$), where $Q = \frac{w\tau}{T_{htt}}$ and is an average value, then there will be at most Q fresh MPDUs colliding with the hidden terminal traffic and each will experience a transmit delay not longer than τ_b. The total transmit delay $\sum_{i=1}^{N} TD(r_i, L)$ satisfies

$$\sum_{i=1}^{N} TD(1, L) \leqslant \sum_{i=1}^{N} TD(r_i, L) \leqslant \sum_{i=Q+1}^{N} TD(1, L) + Q\tau_b \tag{4.10}$$

Since $\sum_{i=1}^{N} TD(1, L) = Ar_{\tau}^{w}(t)/C$ and $\rho = \tau_b/T_{htt}$, Eq. 4.10 can be relaxed and simplified to

$$Ar_{\tau}^{w}(t)/C \leqslant \sum_{i=1}^{N} TD(r_i, L) \leqslant Ar_{\tau}^{w}(t)/C + w\tau\rho \tag{4.11}$$

Considering Eq. 4.11 and Eq. 4.6, and noting that $Ar_{\tau}^{w}(t) = Deliv_{\tau}^{w}(t)$ when there is no loss, we can get

$$\left(Ar_{\tau}^{w}(t)/(Ar_{\tau}^{w}(t) + w\tau\rho C)\right)C \leqslant MC_{\tau}^{w}(t) \leqslant C \tag{4.12}$$

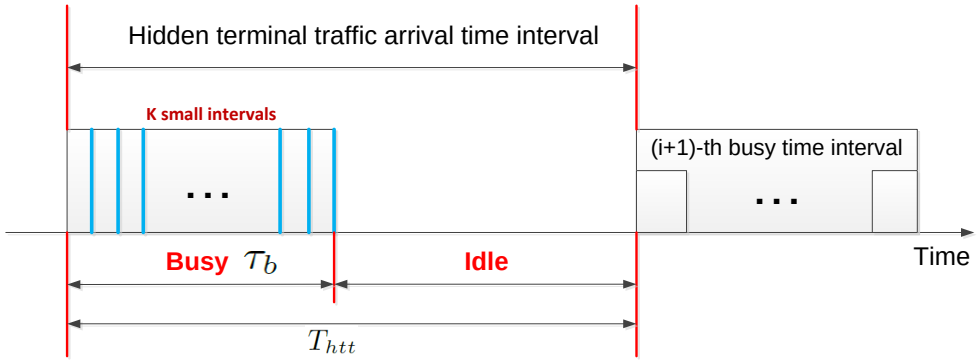


Figure 4.9: Segmented network resource occupation of hidden terminal traffic

Inside Eq. 4.12, $Ar^w_\tau(t)$ is the total number of bits arriving during time duration $(t - w\tau, t)$ at an average arriving rate $Ar(t)$, so by replacing $Ar^w_\tau(t)$ with $w\tau Ar(t)$, we get Eq. 4.7. When the arrival rate is very small($Ar(t) \ll \rho C$), we get the lower bound $Ar(t)/\rho$, and when the arrival rate is large ($Ar(t) \gg \rho C$), the upper bound gets closer to C and the range in Eq. 4.7 becomes tighter.

Part (iii): As we explained earlier, within each hidden terminal traffic arrival time interval T_{htt}, only the transmit delay of the first packet possibly gets affected, so we calculate the expected delay of each first packet. As shown in Fig. 4.9, the busy time interval is divided into K small intervals, each of equal length τ_b/K. If there is at least 1 packet arriving during the first time interval, then the delay will be approximately $(1 - 1/K)\tau_b$ while the probability that this event happens is $Prob(\boldsymbol{N_1} \geqslant 1)$, where the random variable $\boldsymbol{N_1}$ means the number of packets that arrive in the first small interval. If there is at least 1 packet arriving in the second time interval but there is no packet arriving before the second time interval, then the delay will be approximately $(1 - 2/K)\tau_b$ while the probability that this event happens is $Prob(\boldsymbol{N_2} \geqslant 1)Prob(\boldsymbol{N_1} = 0)$. Similarly, if there is at least 1 packet arriving in the i-th interval but there is no packet arriving before the i-th interval, then the delay will be approximately $(1 - i/K)\tau_b$ while the probability that this event happens is given by

$$Prob(\boldsymbol{N_i} \geqslant 1)Prob(\sum_{j=1}^{i-1} \boldsymbol{N_j} = 0) \tag{4.13}$$

The average delay of these first packets is given by

$$\begin{aligned} Delay_K^{\tau_b} &= \sum_{i=1}^{K-1}\Big((1-i/K)\tau_b Prob(\boldsymbol{N_i} \geqslant 1) \times Prob(\sum_{j=1}^{i-1}\boldsymbol{N_j}=0)\Big) \\ &= \sum_{i=1}^{K-1}\Big((1-i/K)\tau_b(1-e^{-\lambda\tau_b/K})e^{-(i-1)\lambda\tau_b/K}\Big) \\ &= (1-1/K)\tau_b - \tau_b e^{-\lambda\tau_b/K}(1-e^{-\lambda\tau_b(K-1)/K})/(K(1-e^{-\lambda\tau_b/K})) \end{aligned} \tag{4.14}$$

where λ is the packet arrival rate in the Poisson traffic model. If there is no packet arriving before the K-th interval, then the delay $(1-K/K)\tau_b$ will be zero. And this is why the summation excludes K in Eq. 4.14. As $K \to \infty$, our approximation becomes arbitrarily accurate, and by the L'Hopital's Rule we get the actual average transmit delay,

$$Delay^{\tau_b} = \lim_{K\to\infty} Delay_K^{\tau_b} = \tau_b - (1-e^{-\lambda\tau_b})/\lambda \tag{4.15}$$

Considering $\sum_{i=1}^{N} TD(1,L) = Ar_\tau^w(t)/C$, $\rho = \tau_b/T_{htt}$ and $Q = \frac{w\tau}{T_{htt}}$, now we can rewrite the total transmit delay $\sum_{i=1}^{N} TD(r_i,L)$ as

$$\begin{aligned} \sum_{i=1}^{N} TD(r_i,L) &= \sum_{i=1}^{N} TD(1,L) + QDelay^{\tau_b} \\ &= Ar_\tau^w(t)/C + w\tau\rho(\lambda\tau_b + e^{-\lambda\tau_b} - 1)/\lambda\tau_b \end{aligned} \tag{4.16}$$

Substituting Eq. 4.16 into Eq. 4.6 and noting $Ar_\tau^w(t) = w\tau Ar(t) = w\tau\lambda L$ (since $Ar(t) = \lambda L$), we get Eq. 4.8. □ □

In the above proof, we use the technique of subdividing a time interval into K smaller ones, and then taking the limit $K \to \infty$. Similar techniques have been used elsewhere, for example, [43, p. 7].

We now look at whether our experimental results match the above theorem. In the experiment, the hidden terminal traffic consists of only UDP-based video packets, each of the same length (1224 bytes) at the MAC layer, with constant frame rate and constant frame size. So the T_{htt} and τ_b in Fig. 4.9 are constant and take values $0.033s$ and $0.01075s$, respectively. According to Eq. 4.3, a packet discarded after the retry limit takes $TD(7,1224) = 10.894ms$ ($R = 7$ and $L = 1224$), where $p = 0$ is used because the interference from the hidden

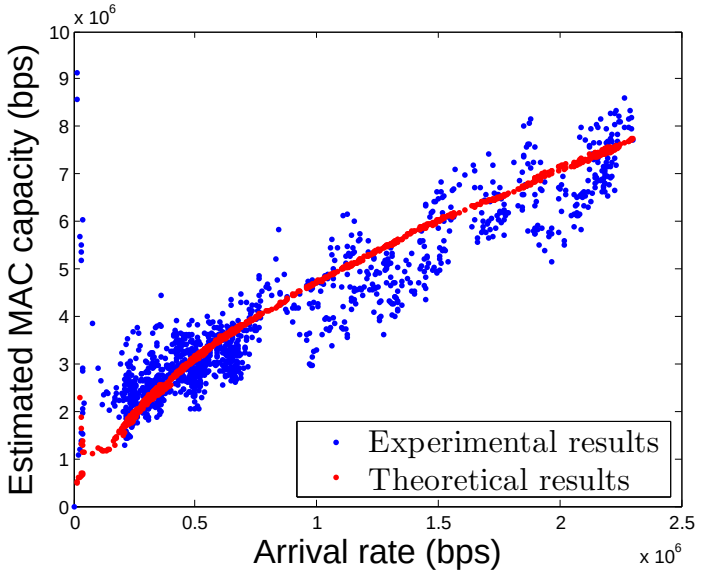


Figure 4.10: Comparison between theoretical results and experimental results of estimated MAC layer capacity as a function of the arrival rate

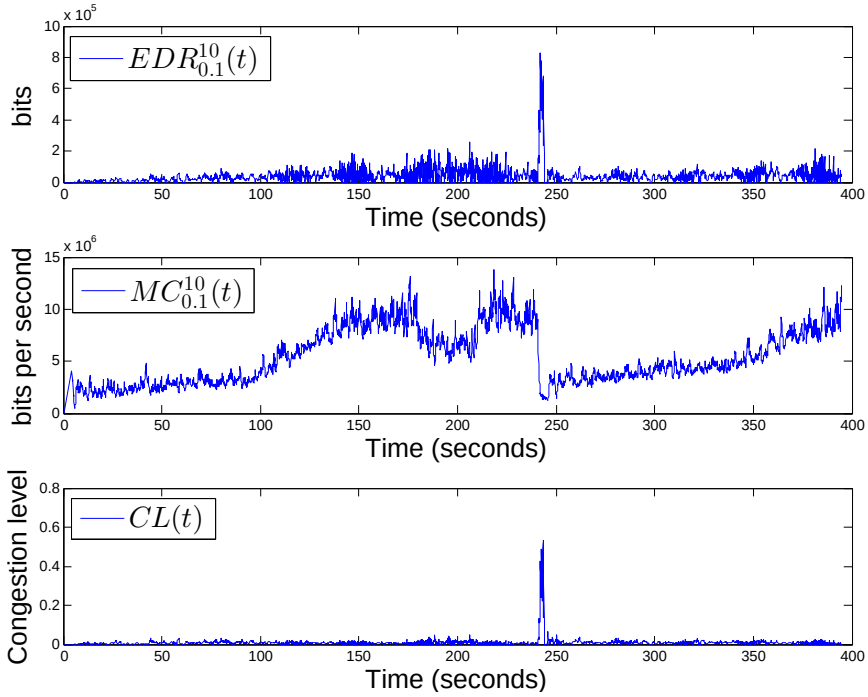


Figure 4.11: Realtime calculations of Eq. 4.5 in WebRTC-based video teleconferencing experiments

terminal traffic cannot be heard and the backoff timer will not freeze. Since $\tau_b = 0.01075s < 10.894ms$, the condition in Theorem 1, $TD(R, L) > \tau_b$, is satisfied. However, packets from WebRTC do not have the same length because audio packets usually are much shorter than video packets, which does not meet the assumption that every packet has the same length L in Theorem 1. So we use the average packet size as the value for L. As the sending rate As gradually increases according to Eq. 4.1, the average packet size is expected to increase as well because video packets account for an increasing portion of the total packets. Since MAC layer capacity C in the theorem refers to the MSDU throughput, i.e., $(L/(L/DataRate + ACK/DataRate + PLCP_{overhead} + aSIFSTime + aDIFSTime))$, where the $DataRate$ is the PHY data rate and the other parameters are fixed and use the same value as in Section 4.3.1, the value of C dynamically changes with the value of average packet size L.

The explanation above allows us to draw the theoretically estimated MAC layer capacity as a function of arrival rate according to Eq. 4.8, and compare the theoretical results with the experimental results in Fig. 4.10, where we only take the data falling in the time window [0, 140 sec] of Fig. 4.7 because there is no competing traffic during this time window. As Fig. 4.10 shows, the two results match well, even though WebRTC traffic is not memoryless and does not follow the Poisson traffic model. Note that for the same value of the arrival rate, the estimated MAC layer capacity may be different because the total number of packets and averaged packet size may be different.

Fig. 4.11 shows the congestion level $CL(t)$ as a function of time t, which is the ratio of the $EDR^{10}_{0.1}(t)$ shown in Fig. 4.11 to the estimate of the MAC layer capacity $MC^{10}_{0.1}(t)$ also shown in Fig. 4.11. Before congestion happens and the network is lightly loaded, the estimate of MAC layer capacity does not accurately manifest the true MAC layer capacity. But this inaccuracy does not lead to inaccurate congestion detection because the excess data rate $EDR^{10}_{0.1}(t)$ is zero most of the time, and the congestion level $CL(t)$ is close to zero regardless of the value of the estimate of the MAC layer capacity. At time 240 seconds, when congestion happens, the estimate of the MAC layer capacity sharply decreases and the excess data rate increases dramatically, resulting in a clear jump in the value of congestion level $CL(t)$. This significant jump allows a proper threshold value to be easily set for congestion

detection.

Note that in Fig. 4.11, the value of the congestion level $CL(t)$ is around 0.55. This is because the threshold value we set for the experiments is 0.35. Based on our analysis, the chosen threshold value is important but not critical. Specifically, if the threshold value of the congestion level $CL(t)$ is set to a smaller value, it makes the detection more sensitive to network load and reacting fast when congestion really happens. But the cost is higher false alarm rate. On the other hand, if the threshold value of the congestion level $CL(t)$ is set to a higher value, it makes the congestion detection process reacting more slowly when congestion really happens. And if a network congestion persists (e.g., excessive network traffic is not reduced by itself), the congestion will still be detected. Using the Fig. 4.11 as an example, if we use a threshold value at 0.8, instead of 0.35, the proposed algorithm still works. This is because before congestion is detected, MAC layer keeps retransmitting an otherwise lost MPDU and the excess data rate $EDR_{0.1}^{10}(t)$ would continue to be increased, leading to a higher jump of the value of congestion level $CL(t)$ than the value shown in Fig. 4.11. As a result, congestion will still get detected. In this thesis, we propose this new congestion algorithm and claim that this new algorithm is able to detect network congestion properly. However, finding an optimal threshold value for the congestion level $CL(t)$ is not our focus (left for future work).

4.4 Cross Layer Approach

The cross layer approach we propose is a new reactive method called early packet loss feedback (EPLF). We exploit the fact that a Wi-Fi link is a likely place where packet loss occurs and the fact that a loss can be detected by the MAC layer of the video sender (not receiving a positive acknowledgement (ACK) packet after reaching the retry limit). If the packet loss is due to channel errors (e.g., collisions, deep fading, etc), the MAC layer directly feeds back the loss information to the RTP layer with a spoofed RTCP packet that carries a NACK message so that the RTP layer can retransmit the lost RTP packet. Since the whole feedback process occurs in the same device (the video sender), the latency is negligible in relation to the RTT, and hence the term 'early' in EPLF. On the other hand, if the packet

loss is due to congestion, the MAC layer does nothing in order to not conceal the packet loss from the congestion control algorithm running at the RTP layer. Theoretical analysis and prototype-based experimental results show that EPLF almost completely eliminates channel-caused video freezes in the decoded video while improving congestion control.

4.4.1 Approach Description

Our proposed method is illustrated in Fig. 4.12, where the packet flow of spoofed RTCP packets (generated by the MAC layer locally) is indicated by blue solid lines, the packet flow of the normal RTCP packets (generated by the video receiver remotely and received from the network) is indicated by red dashed lines and the mixed flow is indicated by purple solid lines. Our method consists of the following steps:

1. The video sender (e.g., the Wi-Fi station (STA) in Fig. 4.1) collects information necessary for generating a spoofed NACK by inspecting packets going in the opposite direction (toward the video sender): it identifies an IP packet that carries a UDP packet that in turn carries an RTCP packet, and records the source IP address and the destination IP address in the IP packet header, the source port number and the destination port number in the UDP packet header, and the Synchronization Source (SSRC) in the RTCP packet header. This 5-tuple is recorded for each RTCP packet flow. Note that such information is accessible even if the RTP/RTCP packets are encrypted by the widely used Secure RTP (SRTP) protocol [11], which encrypts only the RTP/RTCP payload, leaving the RTP/RTCP header in clear text.

2. Next the cause of a transmission failure (which occurs if no ACK is received after the maximum number of retransmissions or retry limit [48, p. 2134] is reached) is determined. If the result from congestion detection(4.3) is positive, we say that the cause is congestion; otherwise, we say that the cause is channel error.

3. If the cause of a transmission failure is determined to be channel errors, the sender creates a spoofed NACK packet at the MAC layer and sends it to its own RTP layer. Specifically, the MAC layer first figures out which RTP packets are lost via deep packet inspection. The MAC layer inspects the payload of the MPDU, which is an IP packet, and checks if the IP packet carries a UDP packet which in turn carries an RTP packet whose Payload Type (PT)

field is video. If so, the MAC layer gets the source IP address, the destination IP address and the RTP sequence number. Then the MAC layer looks up the 5-tuple recorded in step 1 that has the matching IP addresses (with sender and receiver IP addresses reversed) and gets the port numbers and the SSRC from the 5-tuple. Finally the MAC layer builds a NACK packet with the 5-bit feedback message type (FMT) field [73] in the RTCP packet header set to 6 (i.e., FMT=6) and with the payload type (PT) field set to 205 (indicating a transport layer feedback message), and sends it to the sender's own IP layer, which then passes it to the transport layer and RTP layer. The choice of FMT=6 for a spoofed NACK differentiates a spoofed NACK from a normal NACK (generated by the video receiver) in which the FMT field is set to 1, and we assign the value 6 to FMT as the value 6 is not assigned in the RTCP standard [73]. Note that the payload of the spoofed NACK is not encrypted as the MAC layer may not have access to the encryption key shared by the RTP sender and receiver. In contrast, with SRTP, the payload of a normal NACK is encrypted. On the other hand, if the cause of a transmission failure is determined to be congestion, no action is taken.

4. The RTP layer lets a spoofed NACK (with FMT=6 and PT=205) described in Step 3 bypass the decryption module. Without the bypass, the spoofed NACK, which is un-encrypted, will undergo decryption, resulting in garbage output. In comparison, the normal NACK (sent by the video receiver) will be decrypted.

5. The video sender retransmits the lost RTP packet upon receiving the spoofed NACK packet.

Alternatively, a callback can be used. The RTP layer registers a callback at the MAC layer for a MAC layer transmission failure. If the MAC layer records a channel-caused failure, it passes the relevant information (SSRC and RTP sequence number) to the RTP layer via the callback. The callback approach offers the same performance benefit as the spoofed NACK approach. The callback approach eliminates the need for generating a spoofed NACK, but it still needs to perform deep packet inspection and pass the same information (i.e., SSRC and RTP sequence number) from the MAC layer to the RTP layer.

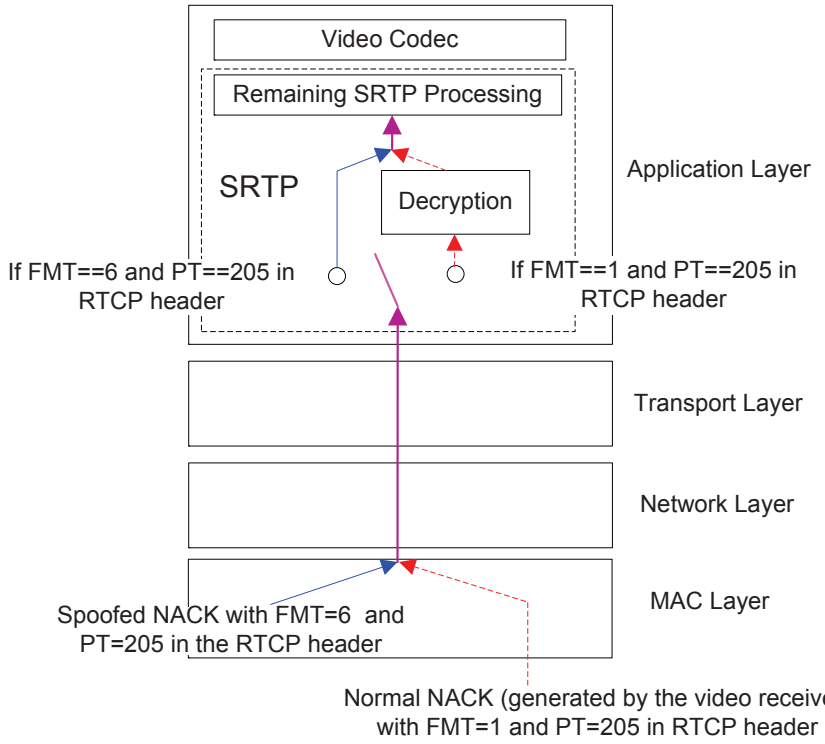


Figure 4.12: In the cross layer approach, the packet flow of the spoofed RTCP packets is indicated by blue solid lines, the packet flow of the normal RTCP packets (generated by the video decoder and received from the network) is indicated by red dashed lines, and the mixed flow is indicated by purple solid lines.

4.4.2 Analytic Model

To gain insight while maintaining tractability, we consider the simple case of isolated packet losses which are far apart such that the next loss happens only after the previous loss is recovered. The default error concealment technique of WebRTC is a type of frame copy: the video freezes at the previous frame if the current frame cannot be perfectly reconstructed before the scheduled display deadline [19]. The video freeze duration is the extra amount of time for which the previous frame is displayed.

The duration of an individual video freeze can be calculated if we know the time that it takes each RTP packet to travel from the sender to the receiver, the feedback delay for each RTP packet loss, the playout delay, the video decoding delay, and the video rendering delay. Following a common definition in the literature, here the playout delay is defined as the difference between the time at which a video frame is generated at the video sender

Table 4.1: Notations for the Analytic Model

Notations	Meaning
T_{playout}	playout delay
T_{oneway}	delay for an RTP packet to travel from the sender to the receiver
T_{feedback}	packet loss feedback delay
f	frame rate
d	per-frame decoding delay
r	per-frame rendering delay
I	time interval for analyzing video freeze durations
δ	some positive constant (in seconds)

and the time by which the video frame must be displayed at the video receiver. To capture the dynamic behavior of the video freeze duration while making the analysis tractable, we partition time into disjoint intervals of duration I, and assume that those parameters remain constant in each time interval.

For each time interval of duration I, we analyze the freeze duration. We use the notations listed in Table 4.1. To make the analysis non-trivial, we require $T_{\text{playout}} > T_{\text{oneway}} + d + r + \delta$, where δ is some positive constant.

A freeze consists of two parts: a deadline missing delay, and a catchup delay in decoding the backlogged video frames, denoted as τ_1 and τ_2, respectively. The deadline missing delay τ_1 is the difference between the arrival time of the retransmitted RTP packet and the time when the video decoder needs to start decoding the corresponding frame in order to meet the display deadline. Thus,

$$\tau_1 = \max(T_{\text{oneway}} + T_{\text{feedback}} - (T_{\text{playout}} - d - r), 0). \tag{4.17}$$

If $\tau_1 = 0$, no video freeze will occur. Otherwise, a video freeze will occur, and the freeze further includes a decoding-catchup delay τ_2 because upon receiving the retransmitted RTP packet, the video decoder has to decode the video frame (to which the retransmitted RTP packet belongs) and the subsequent frames until decoding catches up with displaying. The inter-frame arrival time is $T_{\text{frame}} = 1/f$. The number of frames received during τ_1 is τ_1/T_{frame},

and these frames need to be decoded. The additional number of frames that need to be decoded is τ_2/T_{frame}. We have

$$(\tau_1/T_{\text{frame}})d + (\tau_2/T_{\text{frame}})d = \tau_2, \tag{4.18}$$

which yields

$$\tau_2 = \tau_1 d/(T_{\text{frame}} - d), \tag{4.19}$$

which together with $\tau = \tau_1 + \tau_2$ and Eq. 4.17 gives

$$\tau = \begin{cases} 0, \text{ if } T_{\text{oneway}} + T_{\text{feedback}} + d + r \leq T_{\text{playout}}; \\ (T_{\text{oneway}} + T_{\text{feedback}} + d + r - T_{\text{playout}}) \\ \quad \times (1 + d/(1/f - d)), \text{ otherwise.} \end{cases} \tag{4.20}$$

The video freeze duration as a function of the feedback delay T_{feedback} is shown in Fig. 4.13. We clearly see a turning point at $T_{\text{feedback}} = T_{\text{playout}} - T_{\text{oneway}} - d - r$, below which there is no video freeze.

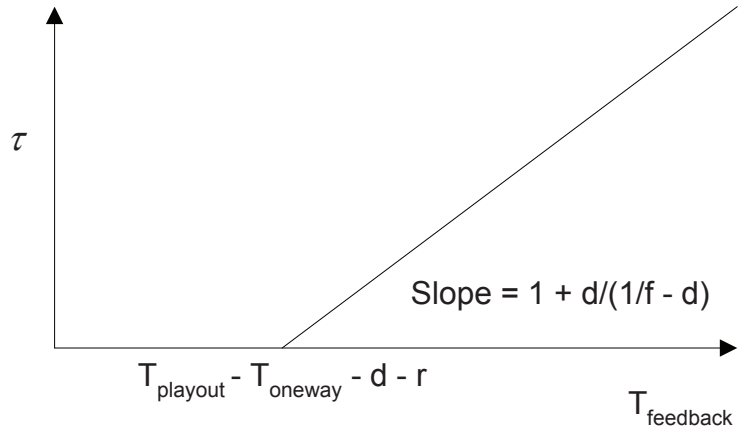


Figure 4.13: The freeze duration τ as a function of the feedback delay T_{feedback}.

Remarks: (i) Note that for a non-trivial analysis we require $T_{\text{playout}} > T_{\text{oneway}} + d + r + \delta$. It follows from Eq. 4.20 that video freeze will be avoided if $T_{\text{feedback}} < \delta$, which typically holds for EPLF because the spoofed NACK is sent within the same device. (ii) Without EPLF, if a NACK-triggered RTP packet fails again at the MAC layer, T_{feedback} will increase by RTT, while with EPLF the increment will be only $t_{\text{local}} \ll RTT$, where again t_{local} is the feedback delay of a single round of EPLF.

4.4.3 Performance Evaluation

We show that EPLF can significantly reduce the video freeze duration and that our analytic model agrees well with the experimental results. The video codec is VP8 (part of WebRTC) with the default setting. The video sequences resulting from the experiments are available at [63]. The reader is encouraged to watch them to experience the improved smoothness in the video playout due to the use of EPLF.

4.4.3.1 Controlled Scenario

We drop an MPDU from the 802.11n MAC layer once per second. By dropping an MPDU, we completely discard the MPDU and do not allow attempts to retransmit it. The RTT is 250ms. The freeze duration (blue solid line) for the case where EPLF is off is shown in Fig. 4.14(a), where the high freeze durations at the beginning are largely due to the small initial playout delays. For the analytic model, we take the partition time interval $I = 1.0$ second. We plug into the analytic model the measurements of T_{playout}, the average value of d and value of f (updated only once per 1 second) and $r = 10$ ms. The freeze duration given by the analytic model is represented by the red dots. For the purpose of clarity, we only plot the analytic predictions for the frames that experience a freeze. Fig. 4.14(b) plots the results for the case where EPLF is on. The amount of video freeze is almost completely eliminated, and the few spikes after frame 500 are due to the large variation in the inter-frame delays, which is not captured by the analytic model.

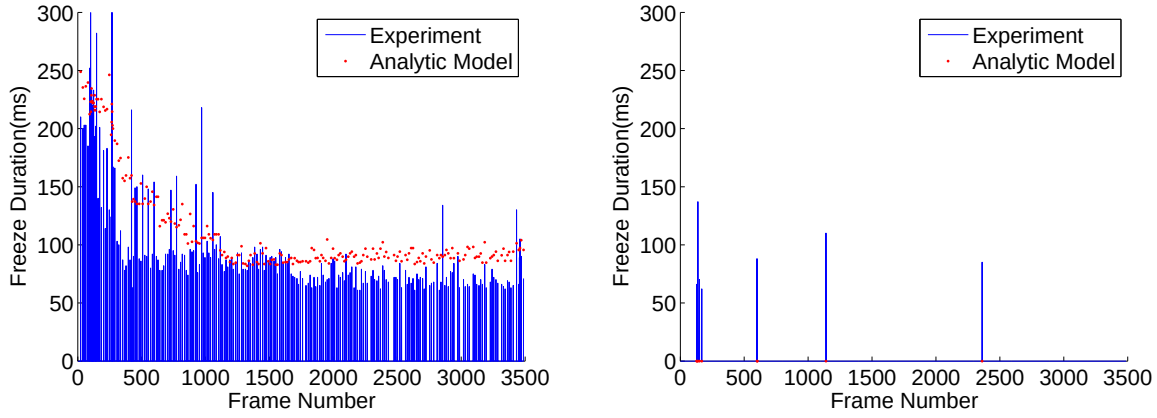


Figure 4.14: The video freeze duration resulting from controlled packet losses at 1 packet per second (a) when EPLF is off, and (b) when EPLF is on.

4.4.3.2 Realistic Scenario

In this scenario, packet losses are determined by the network condition and protocols. Laptop A, which runs EPLF and starts a video call at time 0 seconds, corresponds to a virtual station in a WLAN with an AP and 15 competing STAs, which start generating cross traffic at time 100 seconds. There is another WLAN consisting of an AP and a STA which is a hidden terminal to the video sender. The hidden terminal starts transmitting at time 120 seconds and causes packet losses. The experimental results are shown in Fig. 4.15. The spikes in Fig. 4.15(a) that are much higher than RTT are due to the loss of retransmitted RTP packets. With EPLF, video freezes are almost completely eliminated, except for frames near frames 1700, 2500 and 2900, where the WLAN is experiencing congestion and EPLF does not send a spoofed NACK to avoid disrupting congestion control. Because the majority of the losses are channel-caused, EPLF is able to eliminate most packet losses and video freezes.

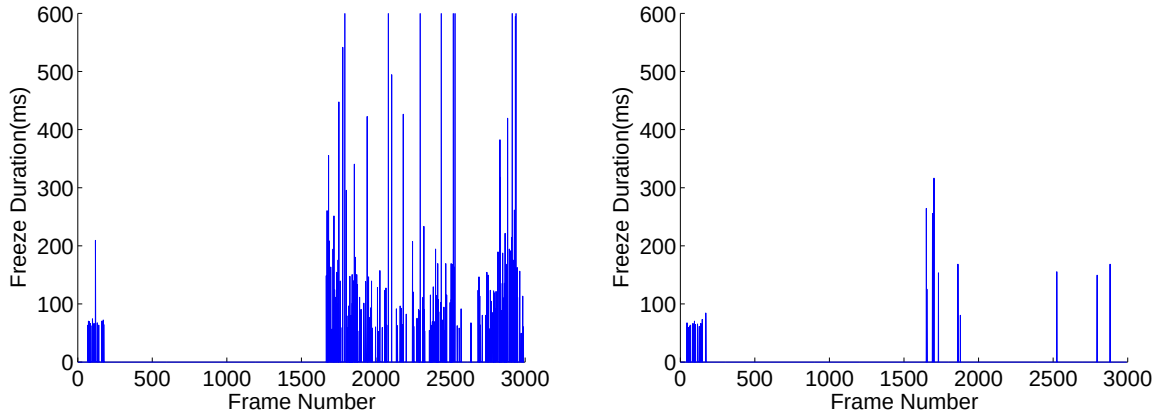


Figure 4.15: The video freeze duration with realistic packet losses for (a) the case where EPLF is off and (b) the case where EPLF is on.

4.5 MAC Layer Only Approach

4.5.1 Challenges in Adjusting The Retry Limit

It is important to find a good compromise for value of the the retry limit, because:

1. The random backoff period in a noisy channel or a heavily loaded channel can be quite different (due to backoff timer freeze), so excessive retransmissions in heavily loaded channel make performance worse. According to Eq. 4.3, in a lightly loaded channel, assuming $p = 0$, we have $\overline{TD(7,1224)} = 10.894ms$; in a heavily loaded

channel,assuming $p = 0.9$, we have $\overline{TD(7, 1224)}$

$= 1.75ms + 236.93ms = 238.68ms$. If a packet takes 238.68 ms to transmit, then the transmit time for a video frame (which usually includes multiple packets) can be hundreds of milliseconds, which is much longer than a typical inter-frame interval. Even worse, delay in transmitting the current packets will introduce delays to subsequent packets, causing more and more delay, that is, the delay is cumulative.

2. Excessive retransmissions also reduce the drainage speed of the MAC layer buffer and increase the probability of buffer overflow.
3. Excessive retransmissions also make a packet's delay larger such that the packet may miss the decoding deadline at the receiver, making all retransmissions of that packet a waste of network resources.
4. Insufficient retransmissions not just add extra traffic at the MAC layer but also leave the packet loss problem unresolved.

In our design, the retry limit is not increased in the case of congestion, so problem 1 does not happen. Our approach checks the value of $CL(t)$ in Eq. 4.5 before performing each extra retransmission. If the buffer gradually builds up, this means that the value of excess data rate in Eq. 4.4 is consistently positive while the output of Eq. 4.5 remains below the predefined threshold. In this case, our algorithm checks the available buffer size to avoid buffer overflow so problem 2 does not happen. Regarding 3 and 4, we will show in Section 4.5.3 that they rarely happen.

4.5.2 Algorithm

We now present the details of the algorithms discussed earlier. Algorithm 1 runs periodically in the background to collect the total amount of arrived data, the total amount of delivered data and the total transmit delay within the most recent time interval τ. Algorithm 1 also makes sure that the three lists in Eq. 4.5 contain the newest data within the time window $w\tau$. Algorithm 2 calculates the current congestion level $CL(t)$ and is called as needed. Algorithm 3 determines whether the retry limit will be increased or not, and it is called before each retransmission. If the MPDU is a retransmission and has reached the default retry

limit, the algorithm will check if any of the three conditions is satisfied: the network is congested, the available buffer size is too small or the predefined extended retry limit is about to be exceeded. If yes, this retransmission will be given up; if not, this retransmission will be performed. Note that the predefined extended retry limit is configurable.

Algorithm 1: UpdatePacketsStatistics()

```
// This function runs periodically every τ seconds. Default
   value for τ is 0.1.
Input: TotalBitsFromIP, TotalBitsSucess and TotalTXDelay
Output: Updated lists: ListAr{}, ListServ{}, ListTXDelay{}
// w is the window size. Default value of w is 10. Replace
   the oldest element with the new one
1  if ListAr.size() ⩾ w then
2      ListAr.pop_back() ; // Delete the oldest element
3      ListServ.pop_back() ;
4      ListTXDelay.pop_back();
5  ListAr.push_front(TotalBitsFromIP) ; // Insert an element at the
    beginning
6  ListServ.push_front(TotalBitsSucess);
7  ListTXDelay.push_front(TotalTotalTXDelay);
8  TotalBitsFromIP = 0 ; // Reset these value to collect new
    statistics for the next τ interval
9  TotalBitsSucess = 0 ;
10 TotalTXDelay = 0 ;
```

Algorithm 2: CalculateCongestionLevel()

Input: $ListAr\{\}$, $ListServ\{\}$, $ListTXDelay\{\}$
Output: $CL(t)$

1 $Ar_\tau^w(t)$ = Sum of all elements in list $ListAr\{\}$;
2 $Deliv_\tau^w(t)$ = Sum of all elements in list $ListServ\{\}$;
3 $\sum_{i=1}^{N} TD(r_i, L)$ = Sum of all elements in list $ListTXDelay\{\}$
4 $EDR_\tau^w(t) = max\{Ar_\tau^w(t) - Deliv_\tau^w(t), 0\}/(w\tau)$;
5 $CL(t) = EDR_\tau^w(t) \sum_{i=1}^{N} TD(r_i, L)/Deliv_\tau^w(t)$;
6 return $CL(t)$;

Algorithm 3: MACLayerTransmitMPDU()

```
1  ...
2  if this is a retransmition then
       // r is retry count with initial value 0.  R is the
          default retry limit.
3      if r ⩾ R then
           // CL_th is the threshold for determining congestion
              status.  BF_th is the buffer size threshold.  R_ex is
              retry limit extension.
4          if CalculateCongestionLevel() ⩾ CL_th OR
5          Current Buffer size ⩾ BF_th OR
6          r ⩾ (R + R_ex) then
7              Delete(MPDU) ; // Network is congested, or the
                available buffer size is too low, or the extended
                Retry Limit will be exceeded.
8              return ;
9          else
10             r = r + 1 ; // update retry count value.
11             if r == (R + 1) then
12                 reset CW ; // Reset the contention window so that
                    subsequent extended retransmissions would be
                    like belonging to a new fresh MPDU.
13 Transmit this MPDU ;
14 ...
```

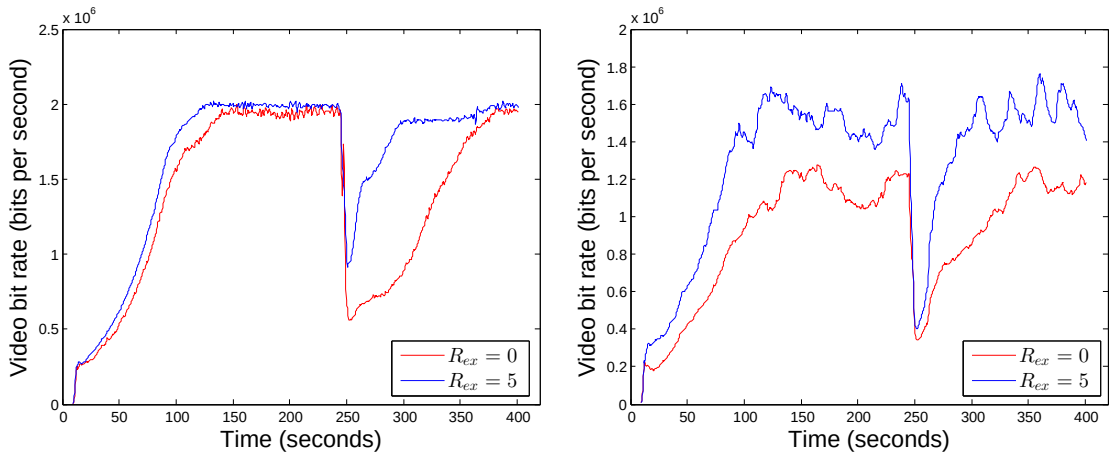


Figure 4.16: Received video bit rate on Laptop B: (a) FEC is turned off (b) FEC is turned on. R_{ex} stands for retry limit extension in Algorithm 3. $R_{ex} = 0$ means that our adaptation algorithm is not used. This definition applies to all following figures.

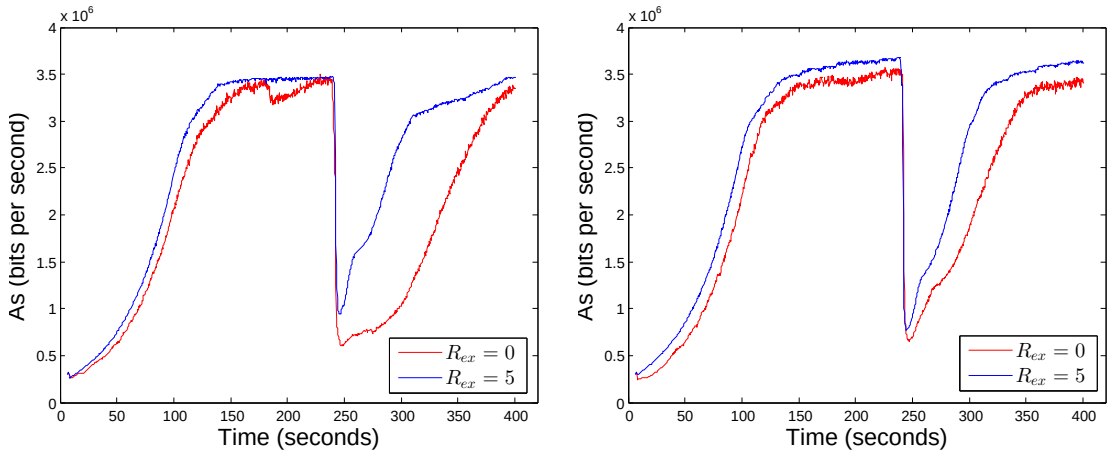
4.5.3 Performance Evaluation

4.5.3.1 Improved Video Bit Rate

As explained in Section 4.2.2, the video sender of WebRTC uses loss-based FEC. If we could reduce the number of packet losses, we expect to see fewer FEC redundant packets being sent out, which results in a higher video bit rate from the video sender. In Fig. 4.16 (b) where FEC is turned on, we are able to confirm the expected improvement which is around 40%. In Fig. 4.16 (a) where FEC is turned off, we can still see some gain because the target bit rate A_s is higher if our approach is used, which is shown in Fig. 4.17 (a).

4.5.3.2 Improved Target Sending Rate

Our approach reduces packet losses so that the fraction of lost packets $f_l(t_k)$ reported is smaller. According to Eq. 4.1, target bit rate As will be smooth and increase steadily, and this can be seen In Fig. 4.17 (a) and Fig. 4.17 (b). When the network is congested, our approach is able to detect the congestion and does not increase the retry limit, so that the fraction of loss $f_l(t_k)$ increases, allowing WebRTC's congestion control algorithm to be aware of the congestion status and properly reduce the value of A_s to mitigate the congestion, as shown in both Fig. 4.17 (a) and Fig. 4.17 (b) at time 240 sec.



(a) (b)

Figure 4.17: Target sending rate of Laptop A: (a) FEC is turned off (b) FEC is turned on.

4.5.3.3 Reduced Video Freezes

Fig. 4.18 shows that the total amount of video freezes is significantly reduced by using our approach when FEC is turned off. When the value of R_{ex} increases, more transmission opportunities are granted for each packet and consequently more channel-caused packet losses can be reduced, which in turn results in less video freezes at the video receiver. The use of FEC can also significantly reduce video freezes but at the cost of sending a lower video bit rate and generally lower quality video due to its high overhead, and FEC typically is disabled in commercial WebRTC product as explained in Section 4.2.5. Comparing Fig. 4.18 (d) with Fig. 4.6 (b), we see that when a higher value of R_{ex} is used, our approach performs as well as FEC in terms of reducing video freezes, but does not have the high overhead problem which is explained in Section 4.2.4.

Note that while the results shown in Fig. 4.18 (d) and Fig. 4.6 (b) provide a visual comparison between the FEC approach and our proposed MAC adaptation approach in terms of the reduced number of video freezes, we also perform a statistical level comparison. Based on 5 times of experiments, when the FEC approach is enabled and our proposed approach is disabled, the average number of freezed video frames is 21.8, and average freeze duration for

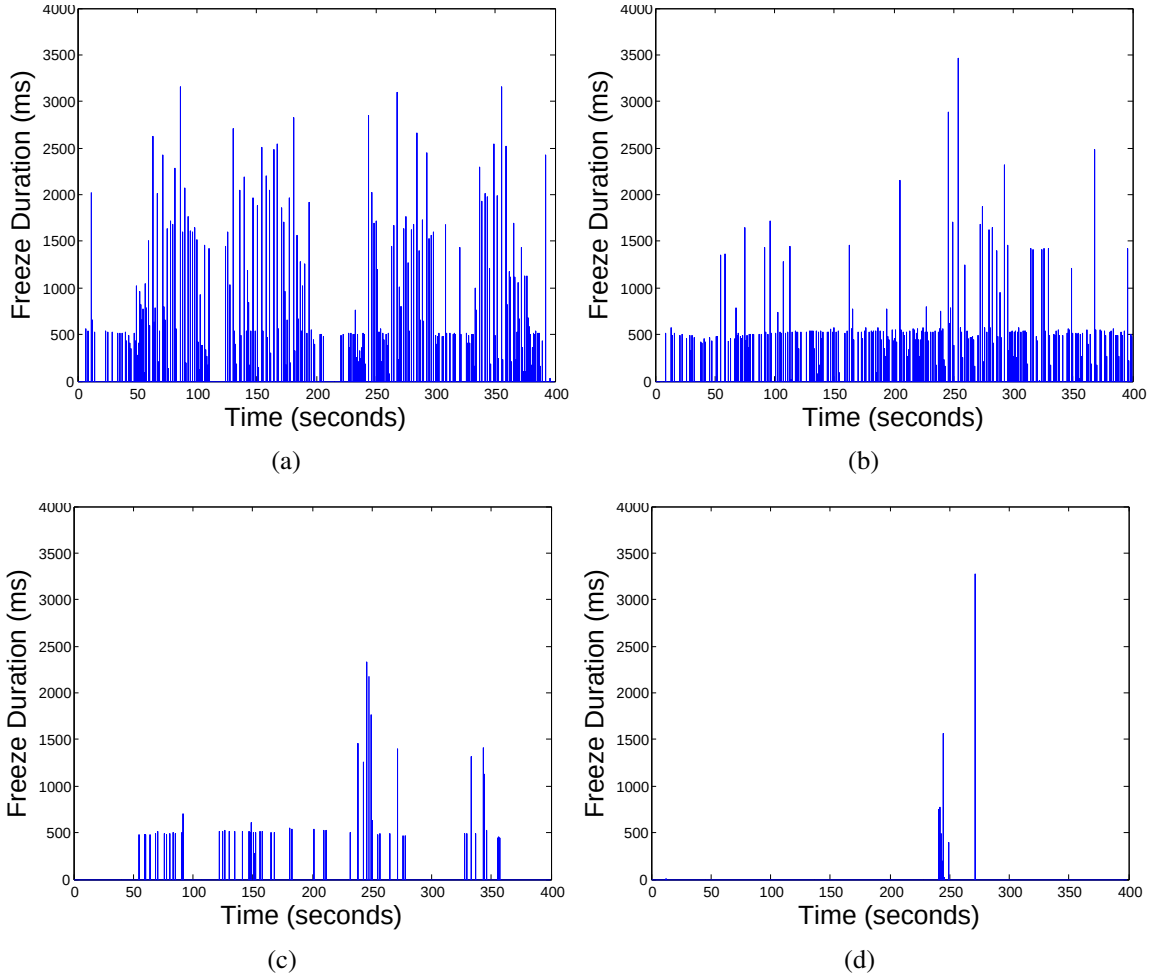


Figure 4.18: Video freeze duration on Laptop B (FEC is turned off): (a) $R_{ex} = 0$, (b) $R_{ex} = 3$,(c) $R_{ex} = 5$ and (d) $R_{ex} = 7$.

each individual freezed video frame is about 687 ms. On the other hand, when the FEC approach is disabled and our proposed approach is enabled, based on 10 times of experiments, the average number of freezed video frames is 7.8, and the average freeze duration for each individual freezed video frame is about 983 ms. The difference on the average number of freezed video frames is because our approach makes the transmit delay (see corresponding definition and analysis of the Eq. 4.3 in Section 4.3.1) of each MPDU longer and the initial playout delay (see corresponding definition in Section 4.4.2) increasing more fast so that there is almost no video freezes at the beginning of each experiment. The difference on the averaged freeze duration is because when our approach is enabled, most video freezes occur at the time when network is congested (around the time moment 240 seconds), where the increased transmit delay of each MPDU makes the packet loss feedback delay (see corresponding definition in Section 4.4.2) even longer. With this statistical level comparison, even though we can observe clear differences between the two approaches in terms of mitigating video freezes due to packet losses, we can still claim that these two approaches perform same well because in the baseline case (without using either approach), the average number of freezed video frames is about 789.7, and the average freeze duration for each individual freezed video frame is about 837 ms. Compared to the base line case, both approaches are able to significantly reduce the average number of freezed video frames.

During the experiments, the traffic generated and simulated within OPNET (e.g. the hidden terminal traffic) is controlled by the simulation seed and could be reproducible if the simulation seed is known. However, the WebRTC traffic is generated and controlled by real machines (see Fig. 4.2). In addition, the WebRTC traffic is not memoryless and does not follow any pre-configured or known traffic generation model. As a result, the traffic pattern (e.g., video frame rate, number of fragmented video MPDUs, frame type, etc) of the WebRTC traffic flow at the same simulation/emulation time instance among different experiments may be different. For this reason, in the four subfigures of the Fig. 4.18, we just show the results from one experiment for each subfigure, in stead of the averaged results from multiple experiments. For a better understanding from the statistical perspective, we still have some quantitative results. Based on 10 times of experiments, the average number of

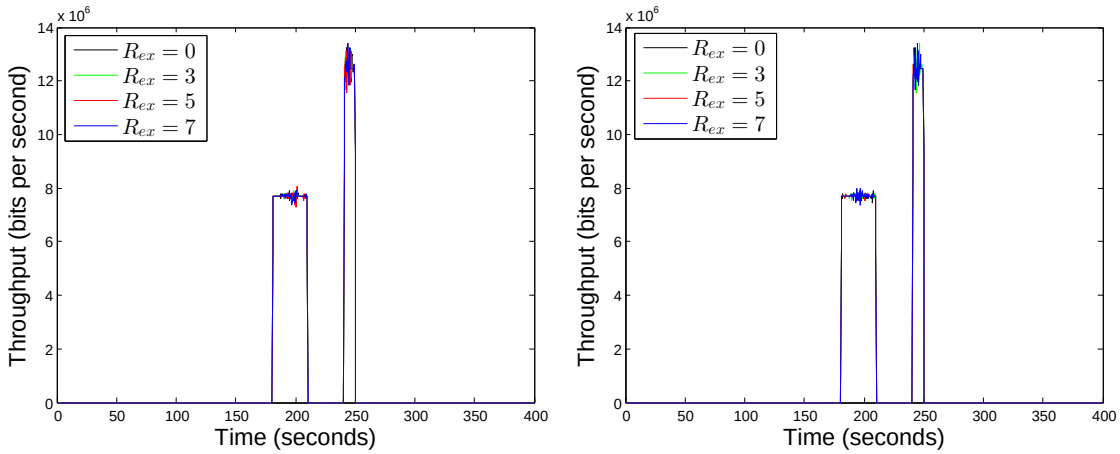


Figure 4.19: Aggregate throughput of competing traffic, when FEC in WebRTC is: (a) turned off; (b) turned on.

freezed video frames when $R_{ex} = 0$, $R_{ex} = 3$, $R_{ex} = 5$, or $R_{ex} = 7$, is 789.7, 471.3, 143.4 and 7.8, respectively. And the average freeze duration for each individual freezed video frame is about 837 ms, 614 ms, 595 ms and 983 ms, respectively. Again, the high freeze duration when $R_{ex} = 7$ is because the total number of freezed video frames is very small and most video freezes occur when network is congested.

4.5.3.4 Impact on Competing Traffic

So far, we have shown that our proposed scheme indeed improves WebRTC-based video telephony over Wi-Fi. In this section, we investigate whether our scheme would affect the performance of competing traffic that does not use our approach.

Fig. 4.19 shows the aggregate throughput of the 8 competing stations named CMP_STA in Fig. 4.2. Since WebRTC behaves differently when FEC is turned off or on, we evaluate our scheme in both cases and present the results in Fig. 4.19 (a) and (b), respectively. As shown in Fig. 4.19, the aggregate throughput of the 8 competing stations does not change appreciably when our approach is used or not, even with different values of the parameter R_{ex}. This result agrees with our expectation. Changing the retry limit does not change the

success probability for each single contention so that the fairness in contention among different stations is maintained. When the network is not congested, more retransmissions lead to the use of otherwise wasted network resources, and the cross traffic is not being hurt because network capacity is enough to accommodate all traffic. When the network is congested, our scheme can detect congestion and does not increase the default retry limit in order to not conceal the packet losses so that the high-layer congestion control algorithm could properly reduce the target sending rate to relieve the network congestion.

4.6 Comparisons between The Two Approaches

In the MAC layer only approach, we proposed to increase the retry limit from a preconfigured value at the Wi-Fi MAC layer if and only if the transmission failure (after the initial retry limit is reached) is caused by channel errors. Since retry limit is left configurable in the IEEE 802.11 standard, and does not require cross-layer coordination, our scheme can be easily implemented and incrementally deployed.

However, the cross layer approach (EPLF) has some advantages over the MAC-layer-only approach. First, the EPLF approach is exclusively applied to real-time video packets, while the MAC-layer-only approach is applied to all types of traffic. A retry limit optimized for video traffic may cause excessive delay for voice traffic, which typically has a much stringent delay requirement than video traffic does. Second, even if one enhances the MAC-layer-only approach to adopt different retry limits for different access categories, the retry limit for traffic flows within the same access category would still be the same assuming a constant congestion level. In contrast, the EPLF approach leaves the retransmission decision to the upper layer, i.e., the RTP layer at the video sender, which may make different decisions for different video traffic flows.

The price to pay for EPLF is to perform deep packet inspection at the MAC layer, generate a spoofed NACK packet, and add additional logic in the RTP layer to let the spoofed NACK packet bypass the encryption module. Also, in terms of applicability, the MAC layer adaptation approach provides benefits not only to the local wireless link, but also to the

remote wireless link (the one connecting the video receiver if the video receiver is also on a wireless link).

4.7 Summary

In this chapter, we present two methods to improve video teleconferencing over Wi-Fi. Inspired by the fact that the bottleneck of an end-to-end connection usually happens on the 'last-mile' wireless link, we investigate possible negative impacts when packet losses occur on the video sender's local Wi-Fi link. We develop a WebRTC-based video teleconferencing testbench which allows us to do more realistic investigations than simulation based evaluation. We confirm that packet losses on an edge Wi-Fi link may cause serious degradation to the receiver's quality of experience.

The first method is a cross-layer approach, which is called early packet loss feedback (EPLF). If and only if transmission failure is caused by channel errors, EPLF lets the MAC layer of the video sender send a spoofed NACK packet to application layer which does immediate retransmissions without waiting for a feedback from the video receiver. Theoretical analysis and experimental results show that EPLF almost completely eliminates video freezes while improving congestion control. EPLF can be used together with other techniques such as application-layer FEC to mitigate the impact of packet losses that occur not only on the local wireless link, but also somewhere else in the network.

The second method is a MAC-layer adaptation algorithm which is to directly reduce the channel-caused packet losses at MAC layer. We also propose a lightweight and passive congestion detection algorithm to distinguish channel-caused and congestion-caused packet losses. As most current applications (such as TCP or RTCP based applications) rely on the packet loss rate to do congestion control, our proposed MAC layer adaptation algorithm helps a higher-layer congestion control algorithm avoid unnecessary reduction in the data sending rate, but leaves congestion-caused losses intact. Experimental results confirm that our approach significantly and accurately reduces channel-caused losses so as to improve the video quality, and does not adversely impact the performance of competing traffic.

Chapter 5

USER ADAPTIVE WIRELESS VIDEO TELECONFERENCING

5.1 Related Work and Motivations

The human visual system (HVS) cannot perceive spatial frequencies in an image that are above a certain limit (or cutoff frequency), which is influenced by viewing conditions such as the viewing distance, ambient luminance and display characteristics (see chapter 2). Frequency components above the limit can be removed to reduce the information in an image before the conventional video compression is applied, thereby improving the efficiency of image/video compression.

In this work, we use the contrast sensitivity function (CSF) model[18][67][10] to characterize this phenomenon. This model establishes a relationship between the *spatial frequency* (in cycles per degree or cpd) and the *contrast sensitivity*. The CSF model characterizes the visibility of a sinusoid as a function of spatial frequency and contrast sensitivity. At a given spatial frequency, there is a contrast sensitivity above which the sinusoid becomes invisible. Pairs of such spatial frequency and the contrast sensitivity form a function, which is called the CSF function, and the function is concave with a peak at a moderate spatial frequency. To get the cutoff frequency, we need to solve for the spatial frequency given the contrast sensitivity, i.e., we need to get the inverse CSF function. Since the CSF function is not monotonic, a monotonic portion of the CSF function is used to obtain the inverse function. Because different regions of an image may have different contrast sensitivity, the cutoff frequency is calculated on a per-region basis. The cutoff frequencies in cpd are then converted to cycles per pixels to low-pass filter different regions of the image before the conventional video encoding is applied. With perceptual pre-filtering, it is reported in [90] that the x264 video encoder can achieve up to 70% improvement in compression efficiency under certain viewing conditions.

This work differs from previous studies [30][90][89][19] in several aspects. First, all prior schemes require explicit feedback for communicating the viewing condition from the user to the video encoder or server. In our scheme, the network entity (MCU) estimates the viewing condition by analyzing the passing video in the opposite direction, which is feasible due to the bidirectional nature of video teleconferencing traffic, and thus eliminates the need for such feedback. Second, these prior studies focus on video streaming, whereas our scheme focuses on video teleconferencing, which differs significantly from video streaming in quality of service requirements and the underlying transport protocols. Third, in the prior studies the perceptual pre-filtering is applied to the uncompressed video, while in our scheme it is applied to a video which has undergone compression once at a client. Since the original compression process has not been adapted to viewing conditions, the proposed filtering and transcoding process still leads to bit rate savings. Fourth, unlike prior studies which focus only on video codecs, we design and prototype the complete system based on the latest video teleconferencing platform WebRTC [92] and a real-world MCU platform Licode [59].

5.2 System Design

We first describe the overall architecture of our proposed system, and then discuss specific implementation techniques.

5.2.1 System Architecture

The system architecture is shown in Fig. 5.1, where two clients are shown to communicate via video teleconferencing with the assistance of an MCU which resides in the Internet. Each client implements WebRTC [92], an open-source real-time video communication application. The MCU runs Licode [59], an open-source platform that implements some basic functions of a typical MCU. Although Licode manages the call setup, for delivering the video (and audio) content, it serves the function of a router only, without any video (or audio) processing capability. To enable user adaptive video coding in the MCU, we implement additional functions including video decoding, video encoding, video analysis, viewing

condition inference. The implementation of WebRTC on the clients is not changed. By utilizing those implemented functions in MCU, we are able to calculate the viewing distance and pixel density of display screen, and then follow the procedures proposed in [90][89] to perform user-adaptive video encoding. Specifically, the perceptual pre-filtering proposed in [89] could be implemented and released as a stand-alone library, and the determined viewing distance and pixel density by us can be given as input parameters when use the library. For design details, please refer to Section 5.2.2, 5.2.3 and 5.2.4.

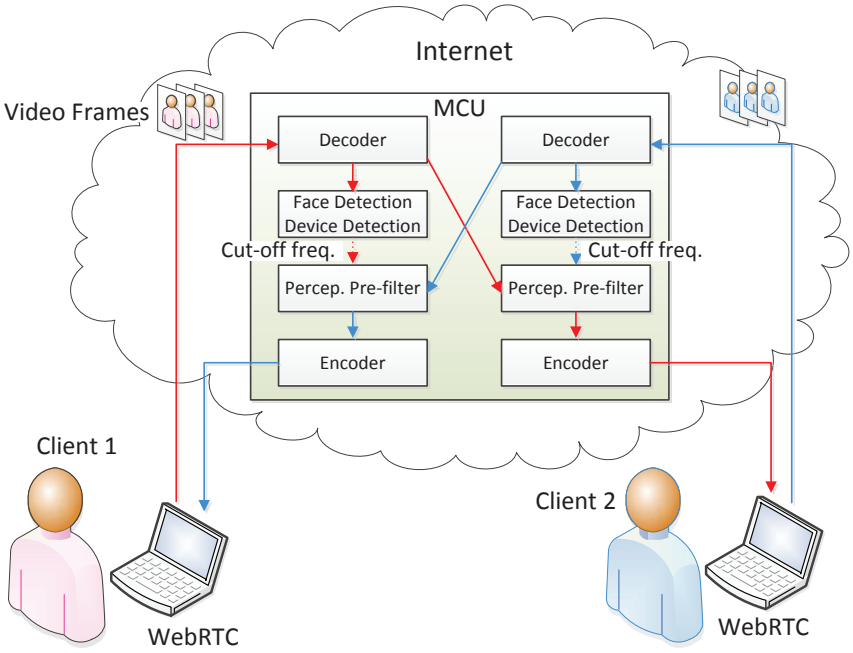


Figure 5.1: System architecture with two clients and an MCU.

Both clients benefit from user adaptive video coding. For clarity, we describe the process that leads to benefiting Client 1. The MCU first analyzes the video frames (i.e., does face detection on the video frames) sent from Client 1 to Client 2 to infer the viewing distance of Client 1. The MCU also extracts the control signaling (i.e., does device detection) from the data sent from Client 1 to determine the pixel density of the display watched by Client 1. The MCU then determines the contrast sensitivity of each region of a video frame of the video flow in the opposite direction sent by Client 2 and consequently the cutoff frequencies and performs perceptual pre-filtering, followed by conventional video encoding. In theory, the conventional video encoder can use the same quantization parameter configuration as

the one used to encode the video arriving at the MCU. In practice, a target video bit rate can be first calculated via a heuristic formula that maps the incoming video bit rate and the viewing conditions to a target video bit rate for video teleconferencing type of content and then passed to the conventional video encoder. Lastly, the MCU sends the encoded video frame to Client 1.

5.2.2 Inferring the Viewing Distance

A well known approach to estimating the viewing distance is to estimate the depth information, which can be done by using dedicated sensors such as infra-red and ultrasonic sensors. Exemplary schemes include Microsoft Kinect [53] and those in [47][27]. However, the existence of such dedicated sensors on the clients may not be guaranteed in practice. In addition, these sensors may collect information other than the viewing distance, raising concerns about privacy. Therefore, this approach may not be the best to the wide deployment of user adaptive video coding.

A less known but more attractive approach is to analyze an image without using any custom sensor, as explained in Fig. 5.2 with a top-down view of the setup. The viewing distance, i.e., the distance between the eyes and the lens, is denoted as s_1. We use the face detection capability of the open source computer vision (OpenCV) [70] library to detect the face and identify the pupils on an image, and then measure the inter-pupil distance d on the image in pixels. The field of view (FOV) or viewing angle of the camera is β. The distance between the lens and the image sensor is s_2. The width of the image is w (in pixels). The real-world inter-pupil distance is D. The viewing angle of the eyes is α. It is easy to get the viewing distance

$$s_1 = \frac{D}{2\tan(\alpha/2)}, \tag{5.1}$$

where

$$\alpha = 2\arctan\left(\tan\left(\frac{\beta}{2}\right)\frac{d}{w}\right). \tag{5.2}$$

It is shown that the value of D for most adults varies in the range from 50 to 75mm [28]. A population average 63mm is used in our estimation. The accuracy will improve if D can be calibrated.

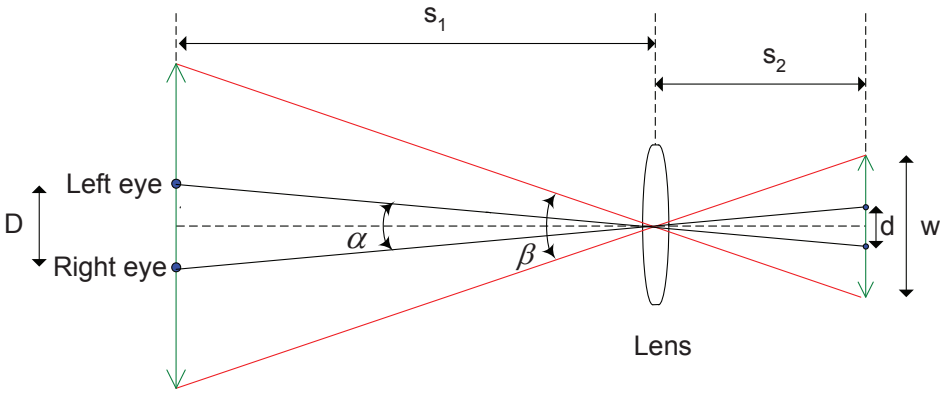


Figure 5.2: The calculation of the viewing angle.

A similar but less accurate method is proposed in [29]. The only difference is that the approximations $\tan(x) \approx x$ and $\arctan(x) \approx x$ are used in [29]. As a result, instead of having (5.2), which is exact, [29] has $\alpha = \beta d/w$. The approximation is inaccurate for large β.

5.2.3 Determining The Pixel Density

Without explicit signaling from the clients to the MCU to inform the latter of the pixel density of the display, the MCU has to extract such information from the control messages sent by the clients. The control message we exploit is the HTTP request messages, where the UserAgent field often contains the device information and the operating system information. As an example, Android devices provide detailed information about the device type, and the MCU can look up a device table to find the pixel density and the reflectivity of the display on that device. The device table lists the pixel density and the reflectivity of the display for all major devices and is updated when a new device becomes available in the market.

5.2.4 User-adaptive Video Encoding

Once the viewing distance and the pixel density are determined, we can follow the procedures in [90][89] to perform user-adaptive video encoding. Specifically, the contrast sensitivity is determined for each location in a video frame. The contrast sensitivity takes into account factors such as the contrast of each location in the video frame, the ambient luminance, and the reflectivity of the display. Then a cutoff frequency (in cycles per degree)

is determined for each location. The viewing distance and the display pixel density are then used to convert the cutoff frequency from cycles per degree to cycles per pixel. Next, these cutoff frequencies are used to low-pass filter different locations of the video frame, and the output is passed to a conventional video encoder. Finally, the MCU transmits the encoded video frame to the client expecting it.

5.3 System Implementation

Our scheme is implemented in Licode (version 0.1.0) [59], an open source MCU platform designed for WebRTC. Licode sets up video teleconferencing sessions, and routes the media and control traffic among endpoints (clients). The delivery of the media follows a publishing/subscribing process: each client publishes its own video, which is sent to the MCU, and the other clients get the video by subscribing to the video via the MCU. As mentioned in Section 5.2, since using user adaptive video coding requires video decoding and encoding, functions which the original Licode does not provide, we integrate the VP8 video codec with Licode, where VP8 is used in WebRTC.

As described in Section 5.2, the MCU needs both the inter-pupil distance d and the pixel density of the display in order to infer the viewing distance. To get d, we write a face detection module which makes use of the OpenCV library APIs. The pixel density of the display is obtained as follows. When a client connects to the MCU, it automatically downloads a Javascript program, which inspects the UserAgent field of the outgoing HTTP request messages to extract the device type information, which is then sent across the network to the MCU for table lookup. The table is locally maintained on the MCU, and it lists device types along with the respective display pixel densities.

The enriched Licode runs on a Linux Ubuntu 12.04 computer which serves as the MCU. The clients are Chrome web browsers, each running on a MacBook laptop. To capture the effect of Internet latency, we add artificial delays on each client using the Linux *tc* utility.

5.4 Evaluation

The one-way delay from each client to the MCU is set to 50ms, which results in an RTT of 200 ms between the two clients. In the experiment, there are two subjects, each looking at a laptop, which is connected via a simulated network to the MCU.

We look at the video bit rate savings resulted from user adaptive transcoding. In the experiment, the viewing distance is fixed at 25 inches. Fig. 5.3(a) shows the arrival bit rate (blue solid line) and the departure bit rate (red dashed line) as functions of time for the case where user adaptive transcoding is used in the MCU. Let the average arrival bit rate be r_a, and the average departure bit rate be r_d. The bit rate savings, i.e., $\eta := (r_a - r_d)/r_a$, is 36.7%. As comparison, we also plot the bit rates in Fig. 5.3(b) for the case where user adaptive transcoding is not used, with the average arrival bit rate denoted as R_a and the average departure bit rate as R_d. The reduction in the bit rates $(R_a - R_d)/R_a$ is only 2.7%. Since the contents of the videos are similar in the two cases, we can compare the two departure bit rates. The reduction in the average departure bit rates, i.e., $(R_d - r_d)/R_d$, is 26.9%. We do the experiment 10 times, and the average bit rate savings is 25%. The more severe fluctuation in the arrival video bit rate in Fig. 5.3(a) is due to the impact of not implementing Google Congestion Control (GCC) [92] in the MCU, and the difference in the arrival video bit rate between Fig. 5.3(a) and Fig. 5.3(b) will go away if GCC is implemented in the MCU.

To confirm that our scheme does not lead to significant degradation in subjective video quality as predicted by the theory, we carry out subjective testing. There are 20 subjects, and we use 3 viewing distances: 20 inches, 25 inches and 30 inches. We ask each subject to evaluate the quality of the video captured and delivered across the network in real time during a video teleconferencing, and the evaluation method is a 5-grade scale where 1 is for bad, 2 for poor, 3 for fair, 4 for good, and 5 for excellent. The results are shown in Fig. 5.4. We see that with 95% confidence interval, the use of user adaptive video coding does not result in degradation in subjective video quality.

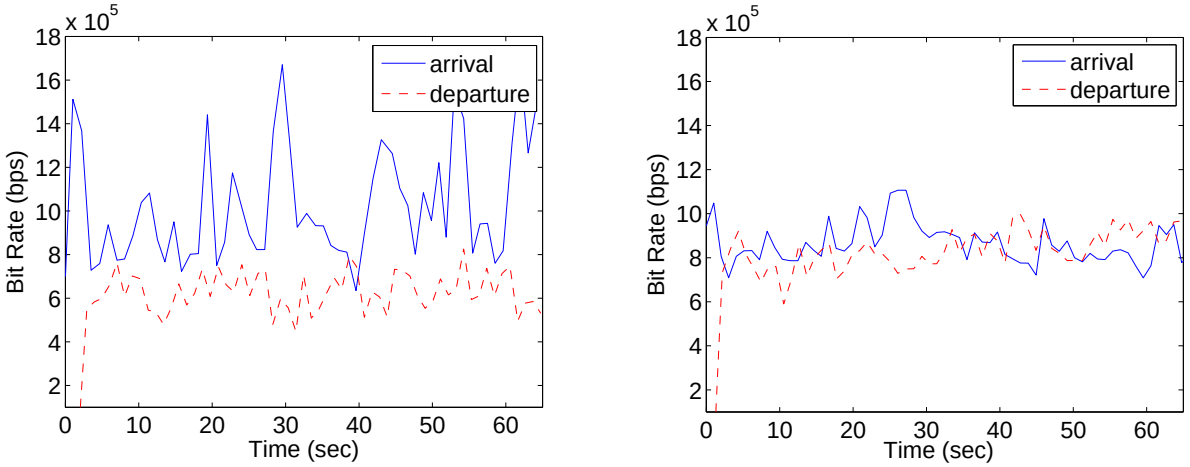


Figure 5.3: The arrival video bit rate (blue solid line) v.s. the departure video bit rate (red dashed line), for the cases of user adaptive transcoding (a) being turned on, and (b) being turned off.

5.5 Conclusion

In this chapter, we propose a user adaptive network-based transcoding scheme for video conferencing to improve the video coding efficiency using perceptual pre-filtering. By analyzing the video sent from a client, the MCU infers that client's viewing conditions, which are then used to adapt the encoding of the video destined to the client. The scheme is implemented in a real-world video teleconferencing system. Experimental results show that our approach can significantly save the bandwidth without affecting subjective video quality.

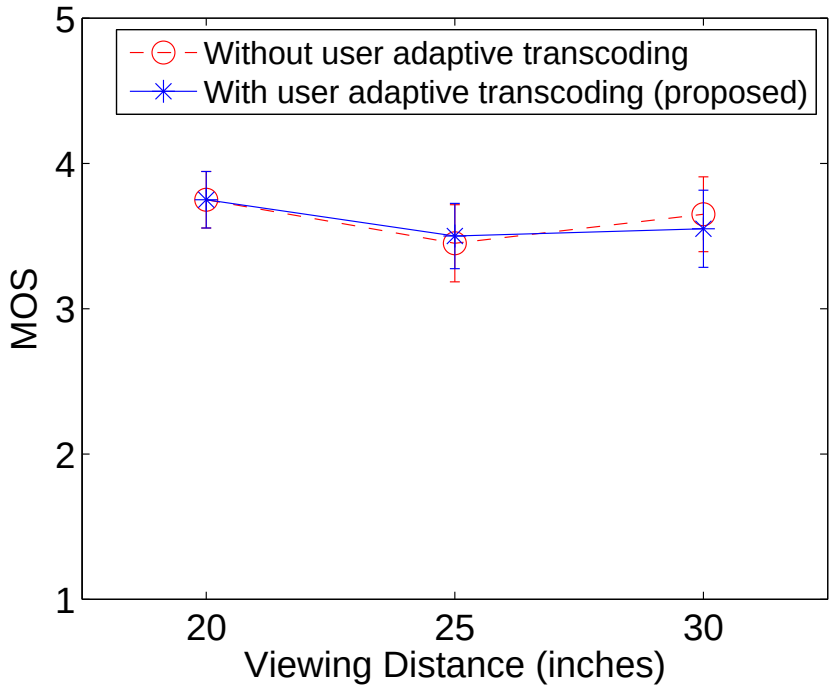


Figure 5.4: Subjective testing scores for the case where user adaptive transcoding is used (blue stars) and the case where user adaptive transcoding is not used (red circles). The error bars stand for 95% confidence intervals.

Chapter 6

MAC LAYER ADAPTATION TO IMPROVE TCP PERFORMANCE OVER WI-FI

6.1 Introduction and Related Work

Wi-Fi has gained growing popularity as the last-mile Internet access technology. At the same time, HTTP based video streaming has been widely adopted for multimedia delivery. As mentioned in Section 3.1, DASH is an emerging standard for adaptive HTTP streaming to enable interoperability in the industry. In a DASH system, a video receiver is strictly based on the performance statistics of the underlying Transmission Control Protocol (TCP) to estimate the available bandwidth of networks in order to select the most suitable video quality dynamically. In addition, not only video streaming, about 90% of the data traffic in the Internet today is carried by TCP [55], and a majority of that traffic may be preferably transferred via a path with Wi-Fi which may be significantly faster and cheaper than a cellular connection. Therefore, a satisfactory performance of TCP over Wi-Fi networks is thus essential to effectively design, deploy and manage a large number of applications on the Internet.

As explained in Section 4.1, there may be channel-caused or congestion-caused packet losses in wireless networks. Packet losses due to path loss, shadowing, fading and interference are classified as channel-caused losses. When the channel is heavily loaded with traffic from multiple contending stations, and the available bandwidth shared by all stations is not enough to accommodate all incoming traffic, the network is congested. If congestion persists, packet losses due to buffer overflow will occur. In a congested network with many active contending stations, collision induced packet losses may also increase significantly. Packet losses due to these reasons are classified as congestion-caused losses.

To overcome the channel-caused losses in Wi-Fi networks, MAC frames containing payload are protected with Forward Error Correction (FEC) that can recover payload data

from up to a certain number of bit-level errors. However, the protection provided by FEC is not enough. In wireless channels, in addition to background noises, fading and path loss, there are many other uncertainties. For example, interference comes from the same network or other networks. Due to these unpredictable uncertainties, it is hard to find optimal values for FEC coding rate and redundancy factor. Lower redundancy than necessary is not enough to protect from bit errors. But higher redundance than necessary leads to high error recovery overhead and low effective data coding rate (throughput).

In today's Internet, several variants of TCP are deployed. These variants differ in their congestion control and segment loss recovery techniques. The basic congestion control algorithms, namely slow start, congestion avoidance, and fast retransmit, were introduced in TCP Tahoe[91]. In TCP Reno, the fast recovery algorithm was added. This algorithm uses duplicate acknowledgements (ACKs) to trigger the transmission of new segments during the recovery phase, so that the network "pipe" does not become empty following a fast retransmit. TCP NewReno[79] introduced an improved fast recovery algorithm that can recover from multiple losses in a single window of data, avoiding many of the retransmission timeout events that Reno experiences. In this work, we focus on TCP NewReno. This is suggested by previous studies that TCP NewReno is widely deployed on the Internet [2]. Furthermore, [2] indicates that NewReno is preferable to Reno, as NewReno provides better support for TCP peers without Selective Acknowledgment (SACK). In OPNET 17.1.A [71], the simulator we used in this work, Reno and NewReno are the only two TCP algorithms implemented. If Reno is used, fast recovery as defined in [91] will be executed when the TCP sender receives triple duplicate ACKs[1]. TCP will perform congestion avoidance, but not slow start. If NewReno is used, fast recovery as defined in [79] will be executed, where two modifications are made on top of Reno algorithm: 1> fast retransmit will be executed

[1] For the typical implementation of the TCP Fast Recovery algorithm described in [RFC2581], the TCP data sender only retransmits a packet after a retransmit timeout has occurred, or after three duplicate acknowledgements have arrived triggering the Fast Retransmit algorithm. A single retransmit timeout might result in the retransmission of several data packets, but each invocation of the Reno Fast Retransmit algorithm leads to the retransmission of only a single data packet.

only once within one window of data; 2> if a partial acknowledgement (ACKs which cover new data, but not all the data outstanding when loss was detected) is received, then TCP will immediately retransmit the next unacknowledged segment. For these reasons, when multiple segments are dropped from the same window of data, Reno may enter and leave fast recovery process several times, causing multiple reductions of the congestion window. While NewReno recovers from multiple segment losses in the same window by retransmitting one lost segment per RTT, remaining in fast recovery until a full ACK is received, and this makes NewReno having better performance than Reno when recovering from multiple segment losses within the same window.

MAC layer optimization for TCP performance improvement in wireless networks has been extensively studied. In [84] the authors present a framework to model TCPs Congestion Avoidance dynamics and evaluated adaptive power control measures for TCP throughput enhancement over wireless channels. They show that TCP-dynamics-aware power adaptation measures lead to substantial enhancement of TCP throughput. The approach in [85] is based upon selecting the best transmission modes depending on the optimal success probability for TCP segments given by dynamic programming (DP)solutions.

The Retry Limit optimization has also been extensively studied in the literature. In [81], the authors propose adjusting the Retry Limit according to the MAC layer data rate. In [57], the author proposes an approach which is motivated by the observation that the Retry Limit settings in the MAC layer can be optimized in such a way that the overall packet losses which are caused by either link erasure or buffer overflow are minimized. In [83], the authors propose joint MAC Real-Time retry limit and link layer adaptation through PHY layer mode selection.

In this work, our proposed idea is different from aforementioned prior work. We differentiate MAC layer packet losses from different reasons by performing congestion detection. If the packet loss is channel-caused, the MAC layer grants more transmit opportunities by temporarily increasing the Retry Limit. If the packet loss is congestion-caused, the MAC layer does nothing in order not to conceal the packet loss from TCP's congestion control algorithm. The work in Section 4.3 also considers reduction of channel caused

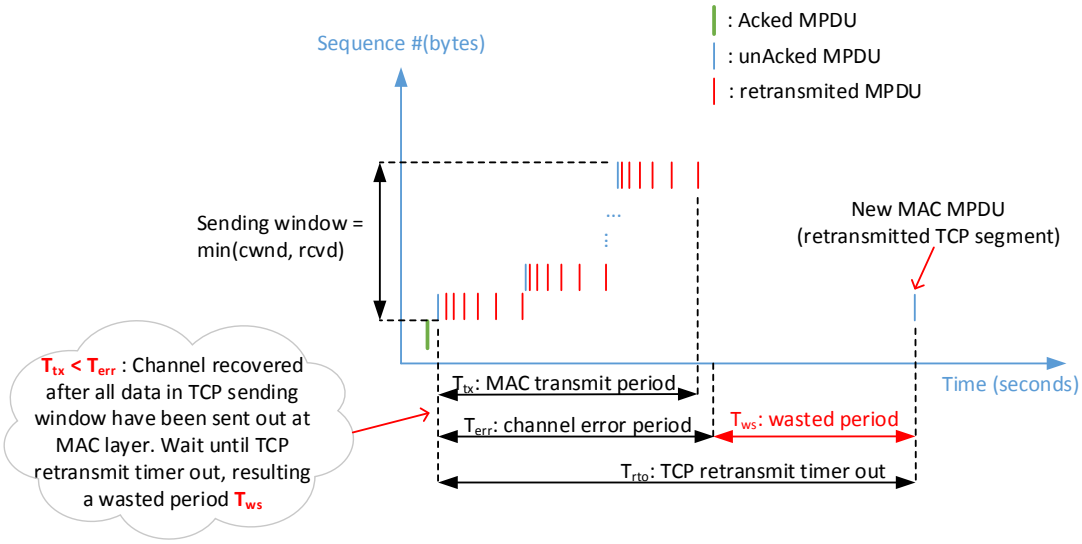


Figure 6.1: TCP performance degradation due to large channel error duration

packet losses, but it is for real-time video traffic. In Section 4.3, we have given details on a light-weight congestion detection algorithm, which was used to adapt the retry limit for the Wi-Fi MAC layer protocol for Wi-Fi stations that carry real-time video traffic. Here, we implement that function for all Wi-Fi stations. Even though the evaluation results in Section 4.5.3 have already shown that our proposed algorithm substantially improves system performance, we want to investigate how the proposed algorithm would help TCP traffic. Note that real-time video traffic uses User Datagram Protocol (UDP), relies on application layer congestion control and has different characteristics (i.e, delay sensitivity, traffic burstiness, decoding deadline and error propagation, etc), compared to TCP.

6.2 Problem Analysis

6.2.1 TCP performance degradation

Depending on the duration of bad channel conditions, the resulting packet losses may cause two types of TCP performance degradation explained as below.

Large channel error duration: Large channel error duration, i.g, more than 50 ms, may occur when moving obstacles and serious interference between a wireless station and an

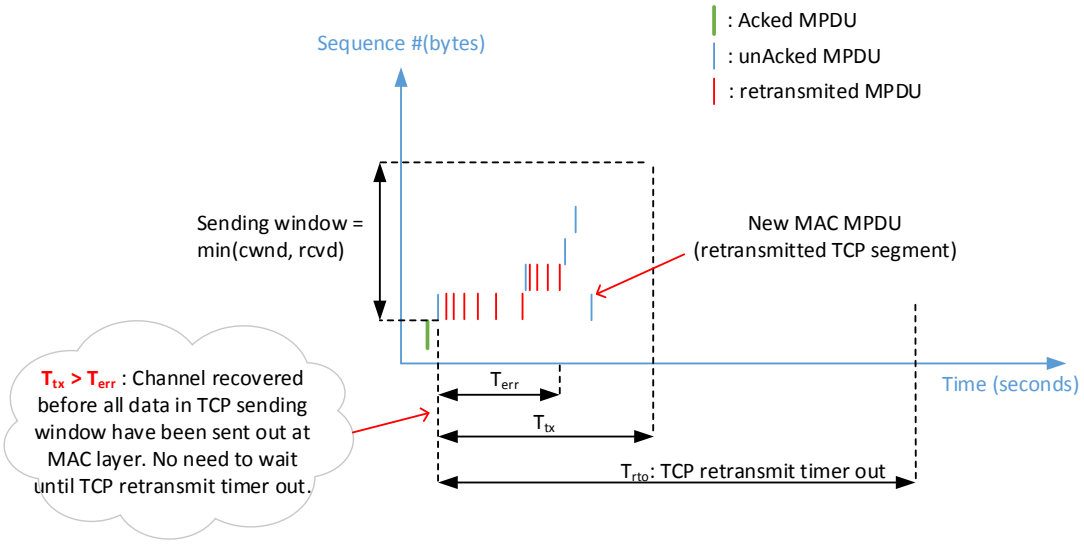


Figure 6.2: TCP performance degradation due to small channel error duration

AP continue to be existed, or a wireless station moves out of the coverage of an AP. During a large channel error duration, noted as T_{err} in Fig. 6.1, MAC layer keeps sending MPDUs corresponding to the TCP segments with sequence number specified by the sending window. For each MPDU, MAC layer will retransmit it until a predefined limit (called Retry Limit) is reached. This overall time duration during which the MAC layer keeps sending MPDUs in the MAC layer sending queue until all MPDUs get removed after reaching Retry Limit is noted as T_{tx} in the Fig. 6.1. Since T_{err} is large enough and $T_{tx} < T_{err}$ is satisfied, MAC layer has no data for this TCP connection to be sent even if the channel is recovered after the duration T_{err}. According to [91] and [79], below actions will be performed at the TCP sender:

- TCP sender will not receive any duplicate ACKs from the TCP receiver, fast retransmit and fast recovery process will not be executed.
- TCP sender will not detect TCP segment losses until the retransmit timer out is triggered, leaving to a wasted time period, noted as T_{ws} in Fig. 6.1.
- Half of the current congestion window size is saved as the slow start threshold.
- Reduce the congestion window to 1 maximum segment size (MSS).

- Changes to the slow start phase.

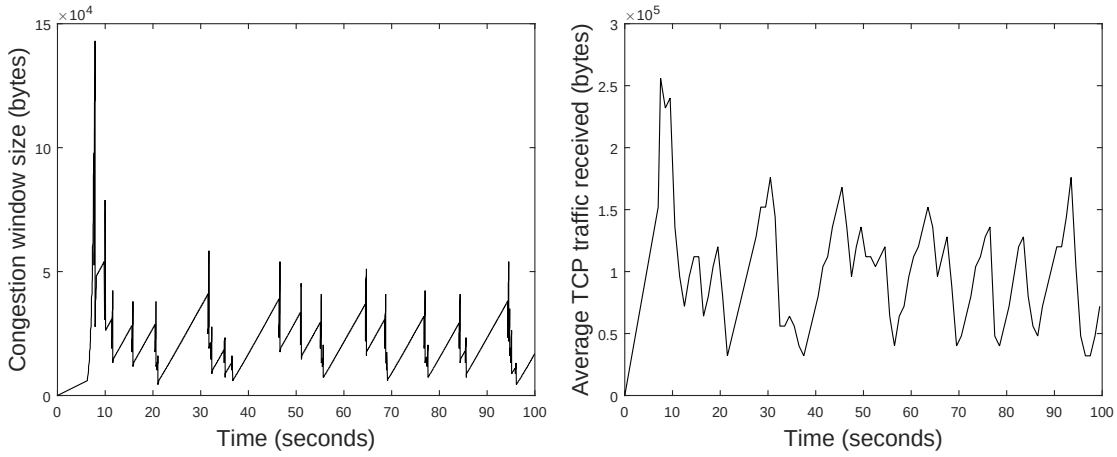
Small channel error duration: Small channel error duration, i.g, less than 50 ms, may occur when temporary interference happens and/or obstacles moves between a wireless station and an AP, or a inappropriate modulation scheme and coding rate used at physical layer of a wireless station due to mobility caused change of channel fading and/or path loss. Note that small channel error duration typically happens much often than large channel error duration. During a small channel error duration, $T_{tx} > T_{err}$ is likely to be satisfied, so that MAC layer still have data for the TCP connection to be sent when channel is recovered. According to [91] and [79], below actions will be performed at the TCP sender:

- TCP sender will receive duplicate ACKs from the TCP receiver, fast retransmit and fast recovery process will be executed (no retransmit timer out is triggered).
- Change the slow start threshold $ssthresh$ to $max(FlightSize/2, 2 * SMSS)$, where $FlightSize$ is the amount of data that has been sent but not yet acknowledged, and $SMSS$ is the sender maximum segment size.
- Change the current congestion window size to $ssthresh + 3 * SMSS$.
- Changes to fast recovery phase instead of slow start.

The details of the fast recovery and fast retransmit algorithms are not the focus of this work, but at least from both above two cases, we can see that TCP sender considers packet losses as indications of congestion and reduce the size of the congestion window accordingly, leading to false alarms of network congestions and performance degradations.

6.2.2 Unnecessarily Reduced and Fluctuated TCP Performance

In this section, a set of simulations are performed in order to verify that the analytic study in Section 6.2 is valid. We use the simulation testbench described in Section 6.4.1. For easier tractability, competing traffic is disabled. Hidden terminal traffic is kept to create random collisions and packet loses at the station STA_A. The FTP traffic from STA_A to STA_B is the TCP traffic under evaluations. Internet delay in the Fig. 6.5 is configurable for deep analysis of TCP behaviors in the next few sections, but is set to a constant value of 100



(a) (b)

Figure 6.3: TCP performance degradations: (a) Congestion window at the TCP sender (b) Average TCP traffic received at the TCP receiver.

ms in this section. Packet loss rate in the Fig. 6.5 is set to be 0% so that packet losses of the end-to-end FTP flow only happen at the Wi-Fi link between STA_A and AP.

Fig. 6.3 shows the growth of congestion window at the TCP sender (the STA_A in Fig. 6.5) and the Average received TCP traffic at the TCP receiver (the STA_B in Fig. 6.5). Due to the random packet losses caused by the hidden terminal traffic, the TCP sender detects loss by duplicate ACKs, retransmits unacknowledged TCP segments, and reduces the congestion window as explained in Section 6.2.1. This produces seesaw oscillations in both the congestion window size and received TCP traffic. These oscillations greatly increase not only delay, but also delay variance for the applications, especially for TCP based HTTP streaming, where fluctuated underlying TCP bandwidth may significantly reduces the user's quality of experience [19].

6.3 System Design and Implementation

Motivated by the issues identified in the previous section, a simple and yet effective MAC layer adaptation scheme is proposed. In this section, we discuss the design principles and implementation details.

6.3.1 Principles

Our proposed MAC layer adaptation scheme relies on the dynamic adjustment of the Retry Limit parameter defined in the IEEE 802.11 standard. The reasons for using the Retry Limit parameter are listed as below:

- Fairness of channel access.
- Practicality.
- TCP traffic is generally not delay sensitive.
- Able to resolve or at least significantly mitigate the TCP performance issues identified in Section 6.2.

Fairness of channel access. To overcome the packet losses as well as the TCP transmission interruptions explained in Section 6.2.1, two parameters in MAC layer are available to be adapted: Retry Limit (RL) and contention window (CW). The duration of CW is used for resolving contentions when several stations are competing for channel access. Changes made to CW simply change medium access priorities, which affects fairness and is undesirable for other stations that do not use our proposed scheme. In contrast, changing the retry limit does not affect each single contention and equal success probabilities can be maintained. In a lightly loaded network, increased retransmissions can better utilize the network resources; while in a heavily loaded/congested network, packet losses are classified as congestion-caused losses (will be explained in Section 6.3.3), and retry limit will not be increased.

Practicality. Note that Retry Limit is left configurable in the IEEE 802.11 standard, and Retry Limit adaptation in our algorithm (be explained in Section 6.3.3) does not require cross layer information or coordinations, our proposed scheme can be easily implemented and incrementally deployed.

TCP traffic is generally not delay sensitive. Due to the integrated retransmission and congestion control algorithms, TCP offers benefits of error detections/corrections, robustness and adaptive use of network resources. However, those benefits are at the cost of increased delay and overhead, compared to UDP. Therefore, applications relying on TCP are generally not delay sensitive, such as web browsing and http video streaming, where up to

several seconds of buffering/cache delay are allowed at the beginning in the web browser or video player [19]. Otherwise, UDP should be used in order to meet requirements of low delay and overhead, such as cloud gaming, real-time video or telephony applications, where the protocol layers above UDP may provide error correction and rate control functions [20].

Resolve the issues of TCP performance degradation. In the case of small channel error duration, the MAC layer grants more transmit opportunities for each MPDU by temporarily increasing the retry limit. Channel-caused packet losses are likely to be completely avoided as the channel error duration is small and channel quickly gets recovered. In the case of large channel error duration, as shown in Fig. 6.1, a straightforward solution increase the T_{tx} to make it larger than the channel error duration T_{err}. Thus, at least one segment can be correctly delivered towards the TCP receiver and the TCP flow will not be completely interrupted, allowing the fast retransmission and recovery mechanism to be triggered instead of waiting for the expiration of TCP retransmit timer T_{rto}. To this end, it is time to formulate the expression of T_{tx} as below:

$$T_{tx} = \sum_{j=1}^{cwnd} TD_j(R_j, L_j) \tag{6.1}$$

and

$$TD(R,L) = \sum_{i=1}^{R} \left(\frac{min(2^{i-1} \cdot (CW_{min}+1) - 1, CW_{max})}{2} \times (p \cdot T(L) + aSlotTime) + T(L) \right) \tag{6.2}$$

Where $R_j \leq Retry\ Limit$, L_j stands for an L-byte long MPDU with payload as the TCP segment j in the current TCP sending window not beyond congestion window *cwnd*. Eq. 6.2 is the transmit delay defined in [20]. The value of $TD(R,L)$ is to represent the time interval from the time the MPDU reaching the head of its MAC queue for transmission, to the time an acknowledgement for this packet is received or discarded upon reaching its retry limit. Note that the queueing delay due to waiting for the service of previous packets to be completed is not included. To increase the T_{tx} duration, we may enlarge *cwnd*, Retry Limit or CW values according to the Eq. 6.1 and Eq. 6.2. Change of *cwnd* and CW are not desirable as it requires modifications of current TCP and MAC standard. Therefore, Retry Limit is selected.

Although the increase of retry limit may bring aforementioned benefits, there may be some adverse impacts which are summarized as below:

- Increased RTT leads to lower TCP throughput.
- Increased queuing delay of subsequent MPDUs potentially leads to lower drainage speed of MAC layer buffer and higher probability of buffer overflows.
- Increased retransmissions are essentially adding more traffic to the network, potentially leading to network congestions.

In the following subsections, we will address the three concerns.

6.3.2 Retry Limit Impact on RTT and TCP Performance

In order to capture the impacts on TCP performance when adjusting Retry Limit, we need first review TCP throughput model. Ignoring the initial slow start phase, it follows from the arguments given in [75] that the evolution of the congestion window can be viewed as a concatenation of statistically identical cycles, where each cycle consists of a congestion avoidance period, followed by detection of segment loss and a fast recovery period. Each of these cycles is called a congestion avoidance/fast recovery (CAFR) period. Based on this CAFR concept, in [76], the authors develop a convincing stochastic model for the steady-state throughput of TCP NewReno. If assuming typical realistic channel conditions, where loss events are usually identified by triple duplicate ACKs so that no timeouts occur, a model called *Model Without Timeout* (NoTO) can be developed here to calculate the steady-state throughput of TCP NewReno as below:

$$T_{NoTO} = \frac{\frac{1}{p} + \frac{W^2 q}{1+Wq}}{(\frac{w}{2} + Wq + \frac{5}{2})R} \tag{6.3}$$

where the window value W is computed as:

$$W \approx \frac{10pq - 5p + \sqrt{p(24 + 32q + 49p)}}{p(3 + 4q)} \tag{6.4}$$

Where R is average round trip time, p is loss event rate, q is segment loss rate within a loss event and W is average value of the peak congestion window size. The parameters used in Eq. 6.3 and Eq. 6.4 are based on a two-parameter segment loss model [76] that captures both

the frequency of loss events and the burstiness of segment losses within a loss event. [76] defines a loss event (LE) to begin with the first segment loss in a round that eventually causes TCP to transition from the congestion avoidance phase to either the fast recovery phase or the timeout phase. Note that an LE can start at any segment, but once it starts, it spans at most one RTT R in Eq. 6.3. The loss events are assumed to occur independently with probability p in Eq. 6.4. Segments transmitted during an LE (except the first) are assumed to be lost independently with probability q in Eq. 6.4 (i.e., parameter captures the "burstiness" of the segment losses within an LE). The two parameters can be set separately, to model either homogenous ($p = q$) or nonhomogeneous ($p \neq q$) loss processes [102].

After the review of TCP throughput model, now we come back to Retry Limit. In a typical communication system , end-to-end round trip time RTT consists of two way (forward route for TCP data and the reverse route for TCP ACK) queueing delay, processing delay and transmit delay at each network node, and propagation delay at each link. As shown in the Fig. 6.5, if we assume all the other network nodes and links between STA_A and STA_B stay as the same status (i.e. same propagation delay, transmit delay, queueing delay, etc), we can focus our analysis on STA_A, which is the TCP sender, to see how retry limit would affect TCP round trip time.

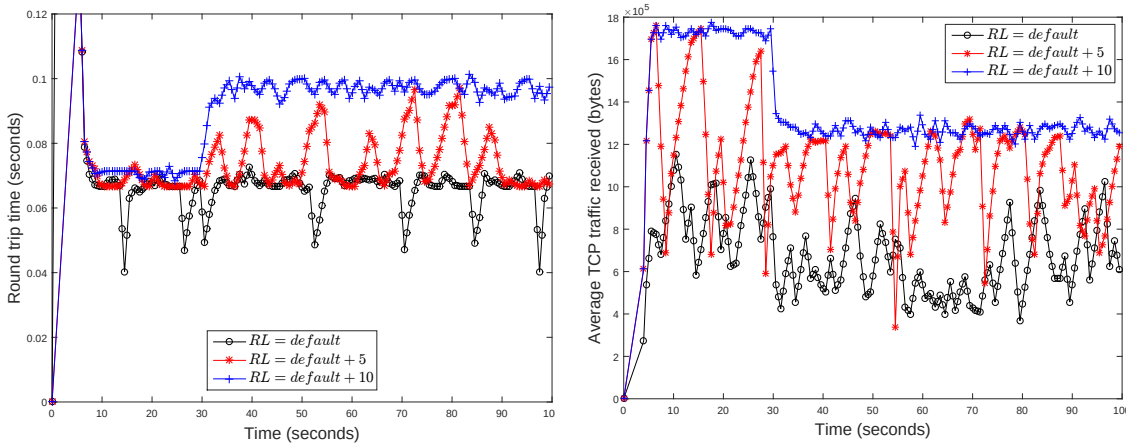
As defined in Eq. 6.2, transmit delay at the STA_A is a function of exponential growth of contention window, Retry Limit and the conditional collision probability p (assuming other parameters remains constant). Since the exponential growth of contention window is upper bounded by CW_{max}, transmit delay is essentially a function of only the Retry Limit and conditional collision probability p. As specified in the 802.11 standard, upon a packet loss, a Wi-Fi station randomly chooses a backoff time from the growing contention window and retransmits the packet after the backoff time expires. However, other contending stations may occupy the channel during the backoff time and force the backoff timer to freeze until the channel is sensed idle again after a DIFS/AIFS period. Therefore, the actual length of the backoff period can be much longer than the original randomly picked backoff time, and this effect can be reflected by the value of the conditional collision probability p. For example, the random backoff period in a noisy channel, where p is very small, or a heavily

loaded channel, where p is bigger, can be quite different (due to backoff timer freeze). For a more concrete example, According to Eq. 6.2, in a lightly loaded channel, assuming $p = 0$, we get $TD(7, 1224) = 10.894ms$; in a heavily loaded channel,assuming $p = 0.9$, we have $TD(7, 1224) = 1.75ms + 236.93ms = 238.68ms$.

We perform a set of simulations (each with 150 seconds simulation time) based on the testbench Fig. 6.5 to verify our analytic study on how Retry Limit could affect the round trip time and TCP throughput, and present the simulation results in Fig. 6.4. In these simulations, Internet delay is set to be 20ms. Hidden terminal traffic begins at the beginning of each simulation and runs to the end. Competing traffic begins at the 30 seconds and runs to the end. The evaluated FTP/TCP traffic starts at around 10 seconds and runs to the end. Three cases, where Retry Limit uses the default value 7, default value plus 5 and default value plus 10, are simulated respectively and the results are drawn in the same Fig. 6.5 for better side by side comparisons.

In Fig. 6.4 (a), before the 30 seconds when competing traffic begins, round trip time values at the TCP sender are almost the same (ignoring those very large initial RTT values which are set by OPNET). This is because the interference from the hidden terminal traffic cannot be heard and the backoff timer will not freeze, so that $p \approx 0$ in the Eq. 6.2. When competing traffic starts at the 30 seconds, the difference between different values of Retry Limit appears. The round trip time value is highest when Retry Limit uses the highest value (default value plus 10) among the three curves.

In Fig. 6.4 (b), TCP throughput of all three curves is reduced after the 30 seconds, but the reasons are different. For the curve using default Retry Limit, round trip delay is the same after the 30 seconds (see subfigure (a) in the same figure), and the decrease of throughput is reduced a little bit due to bandwidth shared by the competing traffic. However, for the other two cases where Retry Limit is extended beyond the default value, throughput reduction is significant as the round trip time is obviously increased. Note that, when Retry Limit uses the highest value (default value plus 10) leading to the highest increase of round trip time, packet losses in the MAC layer are completely eliminated after enough number of retransmission and the throughput is the highest and also stable. For the other two cases,



(a) (b)

Figure 6.4: How the value of Retry Limit affects TCP performance (Internet delay = 30ms): (a) Round trip time calculated at the TCP sender (b) Average TCP traffic received at the TCP receiver.

packet losses at the MAC layer lead to TCP sender enters the costly fast recovery phase and reduces congestion window size multiple times according to the explanation in Section 6.2.1.

6.3.3 Retry Limit Adaptation Algorithm

According to the analysis in Section 6.3.1, the value of Retry Limit is preferred to be as large as possible, in order to mitigate or solve the issues due to channel error durations. Required value of Retry Limit depends on actual network conditions (e.g. overall traffic load in the network, interference level, etc) and channel error durations (i.e. 10ms, 100ms, or even several seconds, etc). However, there may be some cases where small value of Retry Limit is preferred, as listed at the end of Section 6.3.1. One of the case, where increased RTT leads to lower TCP throughput, has been discussed in Section 6.3.2. The conclusion is that the reduced throughout due to the large value of Retry Limit and increase of RTT is still much better than the situation using default Retry Limit, because packet losses are minimized and decrease of congestion window size is avoided. Therefore, a large value of Retry Limit is still preferred. Next, we will address the other two concerns.

Transmit delay defined in Eq. 6.2 refers to the aggregate time spent by an MPDU during the transmission, starting from the time when the MPDU reaches the head of the

sending queue until the MPDU is removed from the sending queue due to successful delivery (receiving an ACK from the destination MAC) or being discarded due to reaching the retry limit. When network conditions fluctuate, the actual transmit delay may vary significantly under the influence of several non-deterministic factors: number of retransmissions, value of random backoff time, backoff time freeze (see the example used in Section 6.3.2 where busy channel and idle channel are considered), and so on. For these reasons, Retry Limit should not be constrained by a specified value.

Inspired by all those analytical studies discussed so far, we propose our Retry Limit adaptation scheme to improve TCP performance as shown in **Algorithm 4**.

In **Algorithm 4**, an unacknowledged MPDU is retransmitted repeatedly until acknowledgement is received or any one of below conditions is satisfied:

- Congestion is detected. Follow the procedures in to perform congestion detection.
- Available Buffer size is below certain threshold. This is to avoid buffer overflow.
- Number of retransmissions have reach a large number. Note that this number is typically a big value, for example, 100, which is hardly to be exceeded during normal retransmissions. As explained before, Retry Limit should not be constrained by a specified value. The reason for defining this number is to provide certain level of flexibility (i.e. sanity check or debug purpose) and also avoid endless retransmissions and occupying the channel forever in case of software bugs.
- Transmit delay is beyond the maximum allowable round trip time. This is an optional check, in case the maximum RTT information is available at the MAC layer and this help to avoid occupying the channel for a long time. For some TCP applications, certain level of delay constraint may be required. For example, HTTP video streaming may tolerate delay only up to several seconds with the help of buffering.
- Radio Link failure is detected. Operating system usually provides radio link status check functions. When the radio link is failed, it does not make sense to continue doing retransmissions.

Algorithm 4: MACLayerTransmitMPDUOptimizedForTCP()

```
1  ...
2  if this is a retransmition then
3      if r ⩾ R then
           // r is retransmission count with initial value 0.  R
              is the default retry limit.  R_max is a big number
              that should not be exceeded.  Delay_max is to control
              transmit delay not exceeding the maximum RTT.
4          if CalculateCongestionLevel() ⩾ CL_th OR
5          Current Buffer size ⩾ BF_th OR
6          r ⩾ (R + R_max) OR
7          TD(r, L) ⩾ Delay_max OR
8          RadioLinkStatusCheck() OR
9          EnergyEfficiencyCheck() then
10             Delete(MPDU) ; // Network is congested, or the
                available buffer size is too low, or transmit delay
                constraint is reached, or the maximum allowed Retry
                Limit will be exceeded, or radio link failure is
                detected, or battery constrained devices have been
                configured energy concerns.
11             return ;
12         else
13             r = r + 1 ; // update retry count value.
14             if r%(R + 1) == 0 then
15                 reset CW ; // Let subsequent extended
                    retransmissions would be like belonging to a
                    brand new MPDU.
16 Transmit this MPDU ;
17 ...
```

- Energy wise concerns. For mobile devices with battery constraint, number of retransmissions may be under control. For example, when battery life is below certain threshold, Retry Limit should use default value.

When one of these events occurs, retransmission of a MPDU is terminated and Retry Limit is reset to its default value. The rational behind **Algorithm 4** is that when network is not congested, more retransmissions lead to the use of otherwise wasted network resources, and the competing traffic is not being hurt because network capacity is enough to accommodate all traffic. When the network is congested, our scheme can detect congestion and does not increase the default retry limit in order to not conceal the packet losses so that the high-layer congestion control algorithm could properly reduce the target sending rate to relieve the network congestion.

6.4 Evaluation

In order to analyze the TCP performance improvements by using our proposed **Algorithm 4**, a set of simulations are carried out and simulation results are presented in this section.

6.4.1 Simulation Testbench

As shown in Fig. 6.5, a typical and realistic network scenario is created, which represents an important lossy communication network that consists of the Internet in the core and Wi-Fi links on the edge. A detailed description about the testbench is summarized shortly. Without loss of generality, this work only investigates packet losses that happen on the link from STA_A to AP, and implement **Algorithm 4** in STA_A to improve the overall performance of the TCP traffic from STA_A to STA_B. Simply put, our approach improves the TCP sender who has a lossy Wi-Fi link.

Summary of the testbench:

PHY and MAC layer: HT PHY at 2.4 GHz with IEEE 802.11n. PHY Data Rate is 65 Mbps. All other parameters use default values in OPNET 17.1.A. All the simulated network traffic uses Best Effort Access Category [24] by default.

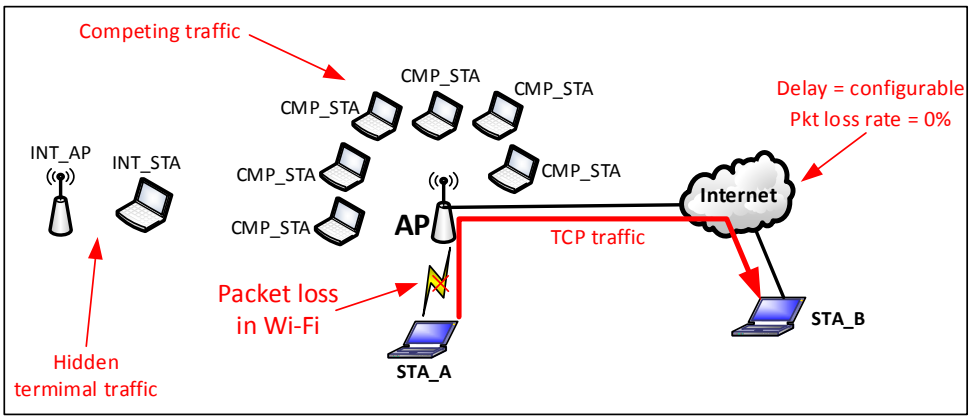


Figure 6.5: A typical lossy communication network simulated in OPNET

TCP traffic path: STA_A ↔ AP ↔ Internet ↔ STA_B.

Main Wi-Fi network: Consists of STA_A, AP and 8 CMP_STA with 802.11n 2.4G Hz. The AP is at a fixed position, while STA_A and CMP_STA are randomly placed (But the distance constraints required to create a hidden terminal network are satisfied).

Congestion: Adjust the cross traffic between CMP_STA and AP to create different levels of congestion.

Hidden terminal Wi-Fi network: It consists of INT_STA and INT_AP with fixed locations. INT_STA and STA_A cannot hear each other, but AP is within interference range of INT_STA. Since INT_AP cannot be interfered by anyone in the Main Wi-Fi network, INT_STA servers as a hidden terminal to STA_A but not vice versa.

Traffic definition: There are three types of traffic in the network. Main traffic between STA_A and AP (finally towards STA_B) is TCP traffic. Cross traffic between CMP_STA and AP, and hidden terminal traffic between INT_STA and INT_AP, are UDP-based video traffic. Hidden terminal traffic uses constant bit rate after it gets started. Cross traffic also uses constant bit rate for easier control but the rates are different in different time periods in order to create different levels of network traffic load.

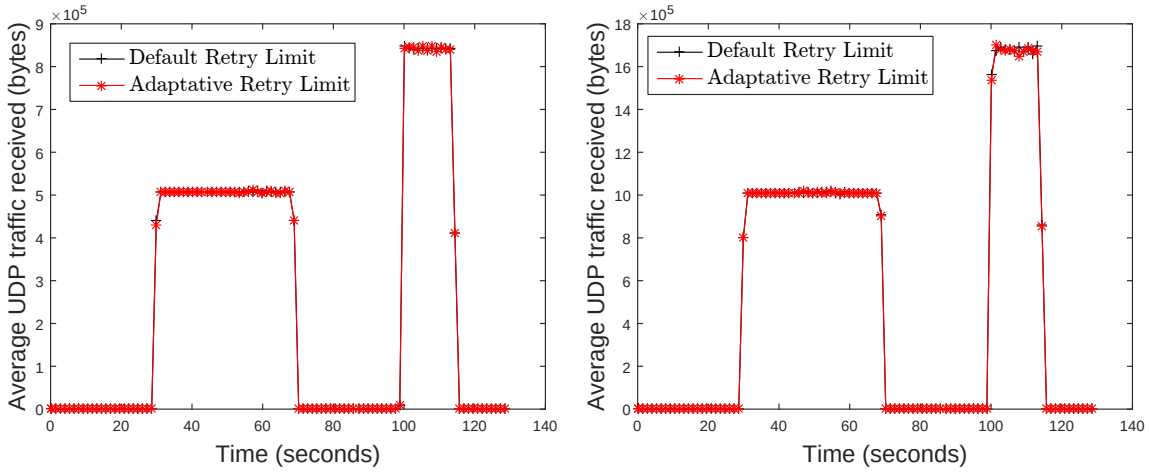


Figure 6.6: Aggregate throughput of competing traffic, when Internet delay is: (a) 30 ms; (b) 60 ms.

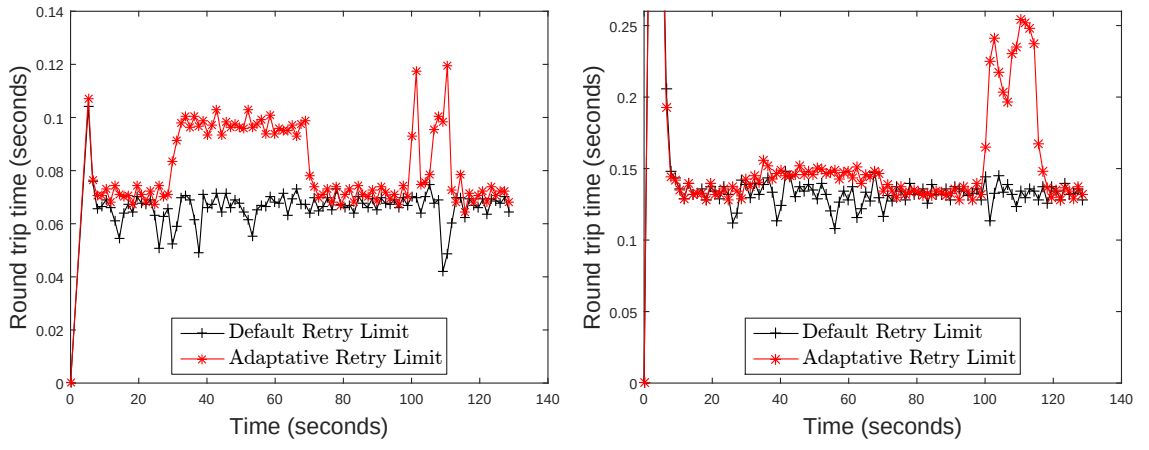
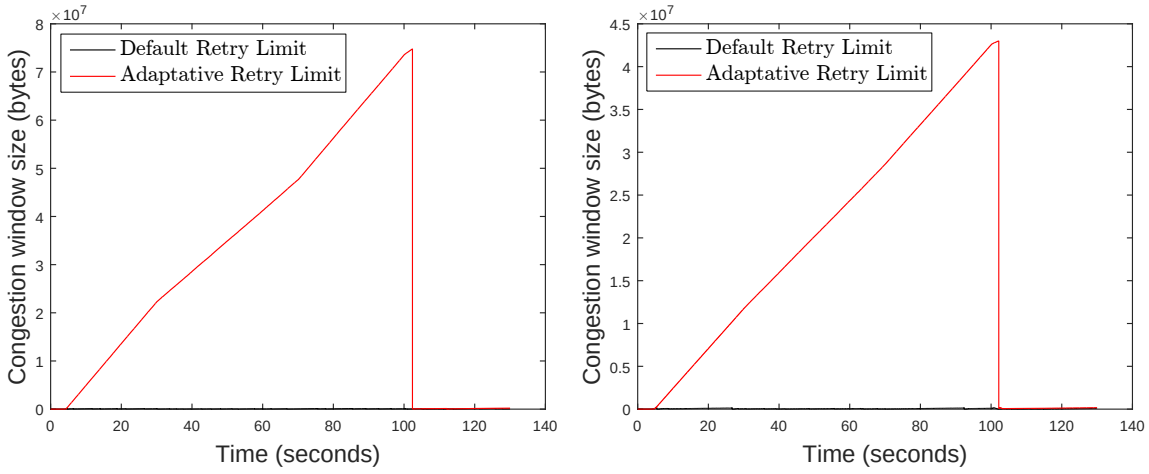
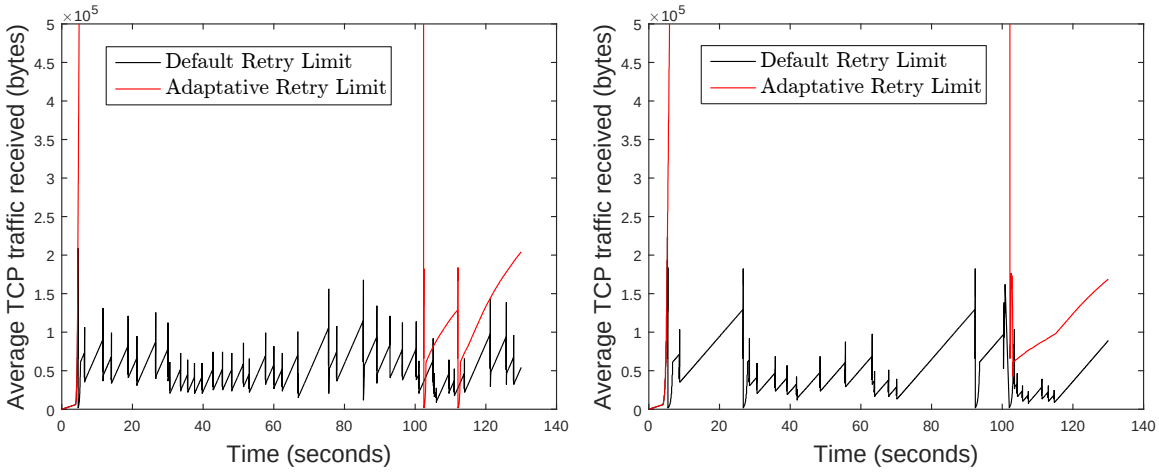


Figure 6.7: Round trip time calculated at TCP sender, when Internet delay is: (a) 30 ms; (b) 60 ms.



(a) (b)

Figure 6.8: Congestion window size at TCP sender, when Internet delay is: (a) 30 ms; (b) 60 ms.



(a) (b)

Figure 6.9: Zoom in details (extended figure of the Fig. 6.8) of congestion window size at TCP sender, when Internet delay is: (a) 30 ms; (b) 60 ms.

6.4.2 Impact on Competing Traffic

Fig. 6.6 shows the aggregate UDP throughput from the 8 competing stations CMP_STA. There are two traffic durations in each sub-figure, representing different levels of traffic load in the Wi-Fi network. As shown in Fig. 6.6, competing traffic throughput does not change appreciably when the proposed adaptation algorithm is used or not, even with different value of Internet delay. This result agrees with our expectation. When the network is not congested such as in the first traffic duration, more retransmissions lead to the use of otherwise wasted network resources, and the competing traffic is not being hurt because network capacity is enough to accommodate all traffic. When the network is congested such as in the second traffic duration, our scheme can detect congestion and does not increase the default retry limit. The decrease of TCP throughput in the case of congested network is presented in the next subsection.

6.4.3 Improved TCP Performance

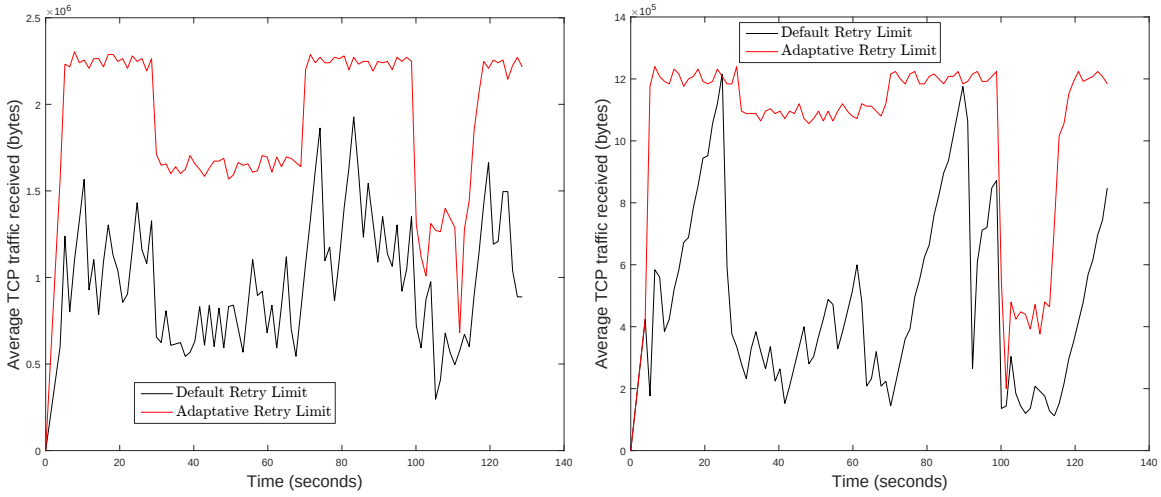
This section is to present experiment results of the evaluated TCP traffic from three aspects (round trip time, congestion window size and TCP throughput), when the proposed adaptation algorithm is used. In the Fig. 6.7(a), when competing traffic is active during the time durations [30 sec, 70 sec] and [100 sec, 120 sec], the round trip time is significantly higher when the proposed adaptive retry limit is used. This result is within our expectations. As captured in the Eq. 6.2, competing traffic causes higher value of p, so that transmit delay would increase accordingly. In addition, in the time duration [100 sec, 120 sec] of Fig. 6.7(a), where network is congested and packet losses are intentionally not reduced, the TCP sender occasionally reduces data sending rate and this causes frequent fluctuations of aggregate traffic in the network and also fluctuations of round trip time of the TCP traffic. When Internet delay is increased to 60 ms, as shown in Fig. 6.7(b), changes of transmit delay in the Wi-Fi link do not bring noticeable increase of overall round trip time when competing traffic is moderate. When the network is congested in the time duration [100 sec, 120 sec] and transmit delay is dramatically increased, we then see obvious increase of round trip time.

Fig. 6.8 (Fig. 6.9 shows zoom-in details of Fig. 6.8) illustrates that when the proposed algorithm is used, the curve of TCP congestion window size is smooth and increases steadily because channel-caused packet losses have been completely eliminated. When the network congestion starts at around the 100 second, the proposed algorithm is able to detect the congestion and does not increase the default retry limit. In this way, packet losses are not reduced so that the TCP congestion control algorithm could properly reduce the target sending rate to relieve the congested network. And we can see that congestion window size fluctuates after the 100 seconds. On the other hand, when default retry limit is used, packet losses happen from time to time, and the congestion window size fluctuates all the time.

Fig. 6.10 shows the average throughput received at the TCP receiver, when our algorithm is used and not, and when Internet delay is 30 ms and 60 ms. In both subfigure (a) and (b) of the Fig. 6.10, we can see that TCP throughput is always significantly improved when our algorithm is used. During the time durations [30 sec, 70 sec] and [100 sec, 120 sec], TCP throughput becomes lower than other time durations. This is due to exactly the same reason as round trip time. As explained above for Fig. 6.7, competing traffic enlarges the transmit delay and round trip time according to Eq. 6.2, and this finally leads to lower expected throughout according to Eq. 6.3. When network begins to be congested at around 100 seconds, the proposed algorithm detects congestion and intentionally let the TCP sender reducing the sending rate to relieve the network congestion. Another important observation from Fig. 6.10 is the fluctuations in the throughput, in the case when the adaptation algorithm is not used. These oscillations in TCP throughput obviously increase not only delay, but also delay variance for the applications, especially for TCP based HTTP streaming, where fluctuated underlying TCP bandwidth may significantly reduces the user's quality of experience [19].

6.5 Summary

In this chapter, we study TCP performance in a typical lossy Wi-Fi network. Based on an analytic study and a set of simulations, we identify two types of issues when TCP performance may degrade due to channel errors. An optimization scheme is thus proposed



(a) (b)

Figure 6.10: Average TCP traffic received at the TCP receiver, when Internet delay is: (a) 30 ms; (b) 60 ms.

to solve or at least mitigate the issues. This scheme relies on Retry Limit optimization at MAC layer. We study the relationship between Retry Limit and TCP round trip time calculation, and then propose mathematical expressions to quantitatively analyze TCP throughput in relation to the change of Retry Limit. Simulation results show that the proposed scheme is able to reduce channel-caused packet losses and significantly improve TCP throughput by helping TCP to avoid entering the costly loss recovery process. When network is congested, the proposed algorithm detects congestion and intentionally let the TCP sender reducing the sending rate to relieve the network congestion.

Chapter 7

DELAY CONSTRAINED MAC LAYER ADAPTATION TO IMPROVE WIRELESS VIDEO TELECONFERENCING

In Chapter 4, one of the proposed schemes exploits the benefits of Retry Limit optimization and performs MAC layer adaptation to help improve the performance and efficiency of wireless video teleconferencing. However, the scheme does not relate the setting of the Retry Limit to the delay constraint imposed by an application's QoS requirements. In this chapter, we investigate the interactions and propose new algorithms.

7.1 Related Work and Motivations

In IEEE 802.11 based wireless networks, transmit delay in the MAC layer refers to the aggregate time spent by an MPDU from the moment the MPDU reaches the head of the sending queue to the moment the MPDU is removed from the sending queue due to its successful delivery (receiving an ACK from the destination MAC) or being discarded when reaching the retry limit. In Eq. 4.6, accumulated transmit delay $\sum_{i=1}^{N} TD(r_i, L_i)$ is used to estimate average MAC layer capacity $MC_{\tau}^{w}(t)$ within a certain time period in order to detect network congestion. However, we ignore the direct relation between transmit delay and retry limit, and how the value of transmit delay would ultimately affect system performance.

As network conditions fluctuate, the actual transmit delay may vary significantly under the influence of several non-deterministic factors: number of retransmissions, value of random backoff time, backoff time freeze (busy channel), etc. Therefore, for the following reasons, the transmit delay needs to be upper bounded:

1. Video receiver imposes a video decoding deadline. The arrival of video packets experiencing a large transmit delay and after the deadline is a waste of network resources.

2. The random backoff period in a noisy channel or a heavily loaded channel can be quite different (see Section 4.5.1 reason 1 for detailed explanation). A sudden increase of transmit delay means the corresponding MPDU takes more time to be transmitted and hence fewer other MPDUs can be transmitted within the same time period, which leads to a dramatic decrease in the received bit rate at the receiver which in turn induces a reduction in the sending rate due to the reporting of a lower Receiver Estimated Max Bitrate (REMB) to the sender. Specifically, according to Section 4.2.2, the sender-side controller computes the target sending bit rate A_s (see Eq. 4.1) that is forced not to exceed A_r. Since the value of A_r is reported by the receiver and is upper bounded by the received bit rate, a dramatic increase in the transmit delay eventually leads to a smaller value of A_s.

Retry Limit adaptation considering video traffic QoS requirements has been studied extensively. In [58], video layers of different importance receive appropriate priority delivery and unequal protection depending on channel conditions. The priority delivery is performed by the proposed priority queuing discipline, while unequal protection is achieved through a retry limit vector with unequal elements that are maintained by the MAC for different video layers. This method requires frequent cross-layer collaborations so that information of the video layers is accessible at the MAC layer. In [16], back off time for each video MPDU retransmission is analyzed, so as to find a set of retry limits for packets in a GOP to minimize the total error-propagation of the GOP according to the delay constraint of packets for presentation at the receiver. This approach still requires the content information of each video packet (e.g., belongs to which video sequence) at the MAC layer so that the error propagation effect of each packet is estimated to guide the determination of its retry limit. [22] proposes an adaptive retransmission mechanism using jamming noises to resolve the contention between real-time traffic and separate the contention of real-time traffic from non-real-time traffic. Since this approach prioritize the contention window size of optimized traffic, it modifies the random access behavior of the current MAC layer protocol and also causes unfairness to cross traffic. In [51], the authors proposes a hybrid retransmission deadline and retry limit to save unnecessary packet waiting time, and a single-video multi-level queue to

prioritize I/P/B slice (packet) delivery. Again, this work also considers the differentiations of different video packets and does require frequent cross-layer collaborations between the application layer and the MAC layer. The proposed scheme in this thesis is different from those prior efforts. We intentionally do not consider unequal protections for different video packets to avoid dependence on application layer's cooperations. This makes our scheme easy to be adopted (not modifying current protocol), and also easy to be deployed (maintaining fairness when competing network resources with cross traffic).

7.2 Delay Constrained MAC Layer Adaptation Algorithm

This section presents the details of the algorithms. Algorithm 5 runs periodically in the background to collect the total amount of arrived data, the total amount of delivered data and the total transmit delay within the most recent time interval τ. Algorithm 5 also makes sure that the three lists in Eq. 4.5 contain the newest data within the time window $w\tau$. Algorithm 6 calculates the current congestion level $CL(t)$ and is called as needed. Algorithm 7 determines whether the retry limit will be extended or not, and it is called before each retransmission. If an MPDU is a retransmission and has reached the default retry limit, the algorithm will check if any of the three conditions is satisfied: the network is congested, the available buffer size is too small or the predefined extended retry limit R_{ex} is about to be exceeded. If so, this retransmission will be terminated; otherwise, this retransmission will be performed. In algorithm 8, the delay constraint $TD_{deadline}$ is checked. If transmit delay of the current MPDU exceeds the tolerable delay constraint $TD_{deadline}$, no more retransmission will be allowed; otherwise, the MPDU will be retransmitted again. Note that both R_{ex} and $TD_{deadline}$ are configurable.

7.3 Performance Evaluation

7.3.1 Testbench and Experiments Setup

We use the same testbench and experiments setup as defined in Sections 4.2.1 and 4.2.3.

Algorithm 5: UpdatePacketsStatistics()

// This function runs periodically every τ seconds. Default value for τ is 0.1.

Input: $TotalBitsFromIP$, $TotalBitsSucess$ and $TotalTXDelay$

Output: Updated lists: $ListAr\{\}$, $ListServ\{\}$, $ListTXDelay\{\}$

// w is the window size. Default value of w is 10. Replace the oldest with new one

1 **if** $ListAr.size() \geqslant w$ **then**

2 $\quad ListAr.pop\_back()$; // Delete the oldest element

3 $\quad ListServ.pop\_back()$;

4 $\quad ListTXDelay.pop\_back()$;

5 $ListAr.push\_front(TotalBitsFromIP)$; // Insert an element at the beginning

6 $ListServ.push\_front(TotalBitsSucess)$;

7 $ListTXDelay.push\_front(TotalTotalTXDelay)$;

8 $TotalBitsFromIP = 0$; // Reset statistics value

9 $TotalBitsSucess = 0$;

10 $TotalTXDelay = 0$;

Algorithm 6: CalculateCongestionLevel()

Input: $ListAr\{\}$, $ListServ\{\}$, $ListTXDelay\{\}$

Output: $CL(t)$

1 $Ar_\tau^w(t)$ = Sum of all elements in list $ListAr\{\}$;

2 $Deliv_\tau^w(t)$ = Sum of all elements in list $ListServ\{\}$;

3 $\sum_{i=1}^{N} TD(r_i, L)$ = Sum of all elements in list $ListTXDelay\{\}$

4 $EDR_\tau^w(t) = max\{Ar_\tau^w(t) - Deliv_\tau^w(t), 0\}/(w\tau)$;

5 $CL(t) = EDR_\tau^w(t) \sum_{i=1}^{N} TD(r_i, L)/Deliv_\tau^w(t)$;

6 return $CL(t)$;

Algorithm 7: MACLayerTransmitMPDUWithFixedRL()

```
1  ...
2  if this is a retransmition then
       // r is retry count with initial value 0.  R is the
          default retry limit.
3      if r ⩾ R then
           // CL_th is the threshold for determining congestion
              status.  BF_th is the buffer size threshold.  R_ex is
              the configurable retry limit extension.
4          if CalculateCongestionLevel() ⩾ CL_th OR
5          Current Buffer size ⩾ BF_th OR
6          r ⩾ (R + R_ex) then
7              Delete(MPDU) ; // Network is congested, or the
                available buffer size is too low, or the extended
                Retry Limit will be exceeded.
8              return ;
9          else
10             r = r + 1 ; // update retry count value.
11             if r%(R + 1) == 0 then
12                 reset CW ; // Reset the contention window so that
                    subsequent extended retransmissions would be
                    like belonging to a brand new MPDU.
13 Transmit this MPDU ;
14 ...
```

Algorithm 8: MACLayerTransmitMPDUWithDelayConstrainedRL()

```
1  ...
2  if this is a retransmition then
3      if r ⩾ R then
           // r is retry count with initial value 0.  R is the
              default retry limit.  R_max is a big number that
              should not be exceeded.  TD_deadline is the configurable
              delay constraint.
4          if CalculateCongestionLevel() ⩾ CL_th OR
5          Current Buffer size ⩾ BF_th OR
6          r ⩾ (R + R_max) OR
7          TD(r_i, L) ⩾ TD_deadline then
8              Delete(MPDU) ; // Network is congested, or the
                available buffer size is too low, or transmit delay
                constraint is reached, or the maximum allowed Retry
                Limit will be exceeded.
9              return ;
10         else
11             r = r + 1 ; // update retry count value.
12             if r%(R + 1) == 0 then
13                 reset CW ; // Let subsequent extended
                    retransmissions would be like belonging to a
                    brand new MPDU.
14 Transmit this MPDU ;
15 ...
```

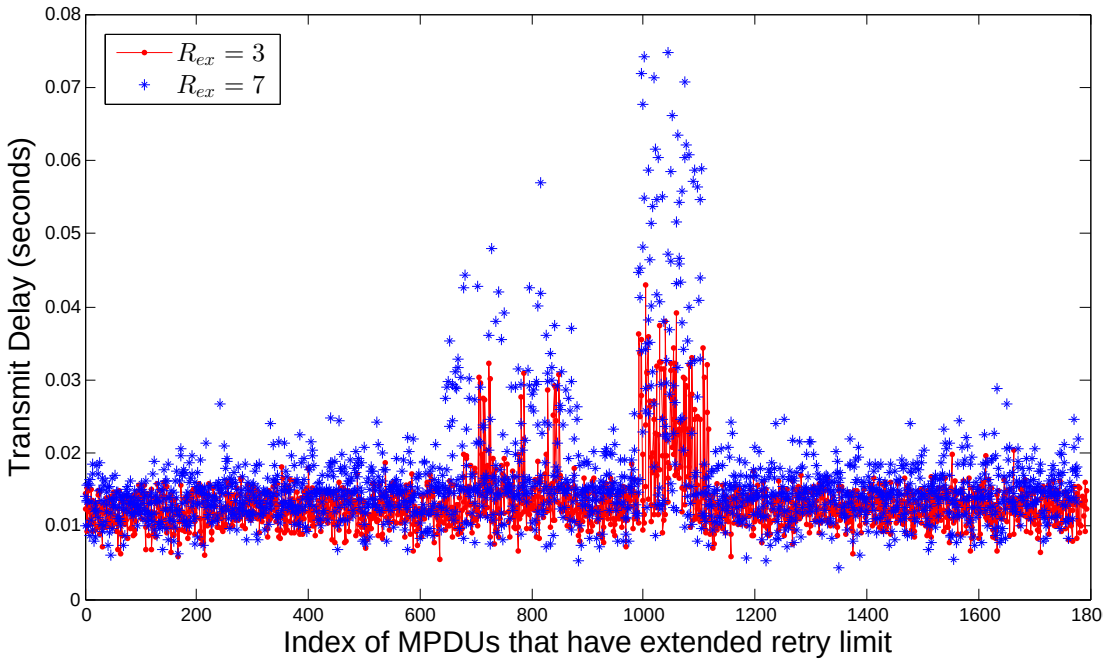
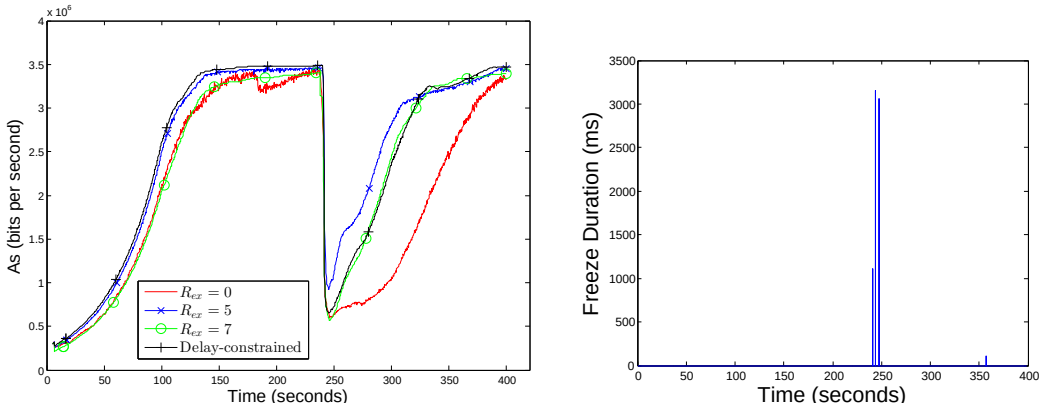


Figure 7.1: Difference in the transmit delay with different values of R_{ex}.

7.3.2 Experiments Results

As explained in Section 7.1, the random backoff period (including the random value of backoff timer and the defer duration of the backoff timer) is quite different when network conditions fluctuate. As a result, the transmit delay may vary significantly as well. This can be seen from Fig. 7.1, where the case with higher retry limit ($R_{ex} = 7$) has larger transmit delay, especially in the time periods [180 sec, 210 sec] and [240 sec, 250 sec] when the channel is busy. Similarly, based on the analysis in Section 7.1, when the transmit delay increases, it is expected that the target sending rate As would decrease accordingly since the Receiver Estimated Maximum Bitrate (REMB) is decreasing. And the same trend can also be observed from Fig. 7.2(a).

In terms of video freeze, we can compare Fig. 7.2(b) with Fig. 4.18(d). Both schemes are able to significantly reduce video freeze that would be present without retry limit adaptation (see Fig. 4.18(a)), but the delay constrained scheme performs even better. In Fig. 4.18(d), we can clearly see the sharp pulse-like freeze around the time of 270 seconds where there



(a) (b)

Figure 7.2: (a) target sending rate comparison when using fixed value of retry limit ($R_{ex} = 7$) and transmit delay constrained retry limit (b) Video freeze duration on Laptop B (FEC is turned off) when using transmit delay constrained retry limit.

is no network congestion and video freeze is supposed to be absent. However, Algorithm 3 based scheme does not take transmit delay into account but simply follows the retry limit rule. As a result, a MPDU may experience a large transmit delay due to excessive retransmissions, making this MPDU finally gets delivered but unable to meet corresponding decoding deadline. Note that there is an such MPDU in Fig. 7.2(b), where a sharp pulse-like freeze can be observed. From this example, we can see that excessive transmit delay not only wastes network resources but also makes video freeze even worse. On the contrary, this will not happen if we use the delay constrained scheme. For example, around the time of 360 seconds in Fig. 7.2(a), there is a short and negligible pulse. Regarding the video freezes happened within time period [240 sec, 250 sec], both schemes can detect network congestion and hence do not increase the default retry limit in order not to conceal the packet losses so that the high-layer congestion control algorithm could properly reduce the target sending rate to relieve the network congestion.

7.4 Joint Discussions on The Two Adaptation Algorithms

In Chapter 6 and this chapter, we propose two MAC layer adaptation algorithms to improve the system performance of wireless video teleconferencing and TCP traffic, respectively. In both algorithms, we point out that retry limit should not be a specific value, but be

adaptive. The difference is that, in the two algorithms, different factors have been considered to make retry limit adaptive.

In wireless video teleconferencing, each video frame must be delivered and decoded by its playback time, making it delay constrained (the end-to-end delay is typically less than 100 ms [9]). For this reason, the retry limit has to be upper bounded such that aggregate transmit delay of a MPDU does not exceed a delay constraint. In TCP traffic, as explained by the analytic study and simulation results presented in Chapter 6, retry limit is preferable to be unlimited until one or more of the conditions defined in the **Algorithm 4** are met. This is because the existing packet loss recovery process in TCP algorithm is rather costly in terms of overhead and delay, and TCP traffic is generally not delay sensitive (otherwise UDP is used). And the overall performance of TCP traffic after performing a large number of retransmissions in the MAC layer is still much better than the case use default number of retransmissions.

The two adaptation algorithms are based on a prerequisite that the type of an application is able to be distinguished, so that different factors can be considered and derived to fulfil the logics of an adaptation algorithm. Ideally, performance requirements (e.g., delay constraint, minimum bandwidth, minimum packet loss rate, etc) and the type of an application should be assigned by the application layer according to the application's specific requirements for the transmitted data, as diverse applications will place different requirements on the performance especially in the future 5G system [7]. One approach to achieve this is to rely on collaboration between the application layer and MAC layer [62]. Another approach is to leverage software defined networking (SDN) based Application-Layer Traffic Optimization (ALTO) [6], where the programmable Data Plane may contain the required application level information. However, the first approach requires cross-layer cooperation, which may be difficult to meet or deploy. In addition, strictly imposing an application's performance requirements on a device's MAC layer may not be appropriate. Taking the delay constraint as an example, network delay may happen elsewhere on the network path from the source to the destination if the source and the destination hosts are not directly connected. Using the SDN approach, the SDN controller can examine at the overall delay requirement of a traffic

flow, and allocate an appropriate delay bound for each segment of the path, where a wireless link may be one of the segments. Standard APIs may be created at the Wi-Fi MAC layer to receive from the SDN controller the configurations of the MAC layer parameters such as the transmit delay constraint of a particular traffic flow.

7.5 Summary

In this chapter, we take QoS requirements into account when propose MAC layer adaptation algorithms to help improve the performance and efficiency of wireless video teleconferencing. The basic Retry Limit optimization algorithm may cause large packet delay since transmit delay of an MPDU may vary significantly under different network conditions. We relate the setting of the retry limit to the delay constraint imposed by the QoS requirements, and propose that retry limit should not be specified as a constant value, but be aggregate delay constrained. Motivated by this idea, we propose a delay constrained MAC layer adaptation algorithm. Experiments performed on a real traffic based prototype platform confirm that the algorithm proposed in this chapter can further improve the performance of wireless video teleconferencing.

Chapter 8

CONCLUSIONS AND FUTURE WORK

8.1 Conclusions

The work presented in this dissertation addresses several issues in the field of wireless video communications. Mobile receivers are troubled by time-varying, error-prone and bandwidth-fluctuating wireless channels, making it difficult to achieve satisfactory QoE and QoS in video applications. Depending on the application scenarios, video services may have different emphases in terms of QoE and QoS. We therefore propose several innovative adaptation techniques to address the challenges in two main categories: wireless video streaming (non real-time) and wireless video teleconferencing (real-time).

By understanding the limits of human visual system and by analyzing the characteristics of human viewing environments, video transmission to mobile devices can be adaptive to user's viewing perception capability. Motivated by this observation, we integrate a technique called User Adaptive Video into our research. Our prototype platforms for video streaming and video teleconferencing, respectively, enable the design of intelligent video communication systems adapting not only to network conditions but also to factors affecting human perception of visual information. Experiments results show that such adaptation not only results in significant bandwidth saving without perceived loss of quality to the user for improved QoE, but also helps cross traffic to create a win-win situation.

In wireless networks, packet losses due to fading and interference are common. Those packet losses often lead to serious video quality degradation, like artifacts in the decoded video, which affects not only the current video frame, but also subsequent frames because of error propagation resulted from the use of prediction from previous video frames. In addition, packet losses are often interpreted as indications of congestion by the congestion

control protocol at the higher layers, causing decrease of data sending rate and thus lower video quality. We propose several adaptation techniques to different layers of the protocol stack to minimize packet loesses. We design a lightweight and effective congestion detection algorithm to help determine whether a packet loss is due to network congestion or not. A retry limit optimization algorithm is then used to minimize packet losses in the case of no congestion being detected. By taking QoS requirements into account, we analyze specific issues in the applications of video teleconferencing, and this analysis motivates us to further optimize the retry limit algorithm by imposing a delay constraint. We also consider cross layer adaptations. In one proposed cross layer approach, the MAC layer of a video sender sends feedback information to its application layer so that retransmission of a lost packet can be immediately triggered, without waiting for the costly and time-consuming feedback from the corresponding video receiver. In addition to application layer and MAC layer, we also analyze the behaviors of transport layer. Specifically, we investigate how to improve TCP performance in wireless networks. We identify issues of TCP performance degradation due to common channel errors via both analytical studies and simulations in a typical Wi-Fi network. We then analyze TCP traffic characteristics and throughput model to propose MAC layer adaptations. In all these proposed protocol adaptations, we perform extensive evaluations to validate our design and compare performance improvements against the cases without using our designs.

To demonstrate the impacts of our research work, we have chosen two widely accepted and deployed industry standards, MPEG-DASH and WebRTC, as the basis to build our own prototype platforms. Using these standards, we are able to identify issues that are likely to happen in realistic situations. Also, based on these realistic issues, we are able to contribute with practical solutions. When design our solutions, we try to not only minimize or even avoid modifications to the current protocol stack but also make sure our solutions are transparent to existing applications that do not use our solutions. These considerations add another benefit to our work, which is easy to be adopted and incrementally deployed.

8.2 Future Work

While we have developed a rich set of fully functional adaptation techniques that demonstrate significant performance improvements for wireless video streaming and wireless video teleconferencing, several parts of the system may still need further improvements and there exist open issues that must be addressed before those techniques are made available to the public.

In our congestion detection algorithm, although measurements are performed on real traffic, the traffic comes from a single machine. If multiple machines inject WebRTC traffic (or multiple WebRTC sessions are generated from one machine), and those traffic are active in different stages (e.g., multiple WebRTC traffic come and go before reaching a steady stage), measurements may interfere with each other. In our prototype, we do not consider this case.

In the chapter discussing TCP performance, we are particularly interested in the case where the application data above TCP is video streaming data. However, we did not consider web browsing. On a typical webpage, there may be different types of elements embedded, such as text, pictures and videos, and multiple TCP connections are established to retrieve those different objects simultaneously. Our proposed adaptation algorithm does not consider multiplexing of TCP connections with different life cycles (e.g., short lived TCP connections terminate before reaching a steady stage), and neither does it consider the overall responsiveness of web browsing when downloading and rendering those objects in a web browser.

The proposed adaptation algorithms in this dissertation are for unicast use cases. Further extension may be necessary for multicast scenarios, where multiple wireless capable devices are actively receiving video from the same video gateways. In the multicast case, adaptations of video relay or caching methods need to be considered. For example, system performance can be improved if users connect to peers that are geographically close to them (i.e., device to device communications).

Furthermore, the user adaptation technique proposed for MPEG-DASH relies on the fact that a DASH server maintains a video with many different resolutions. When considering different viewing conditions, multiple versions of the same resolution may be also needed.

The storage burden of having multiple streams of the same content may be addressed by using a technique called Scalable Video Coding. In addition, human visual system has many other limits, including but not limited to oblique effect, horizontal effect, contrast constancy, etc. Further research involving these factors would add additional benefits when performing user adaptive video encoding. Moreover, as the sensors in a mobile device are used to determine human viewing environment, they can also be used to collect data on individual user's preferences of different video contents. User attentiveness while the content is being played can be another area of research which will help the video/advertisement makers to provide specific contents (i.e., marketing strategies for specific movies and advertisements) to target people.

BIBLIOGRAPHY

[1] IEEE Std. 802.11e. 2005.

[2] M. Allman A. Medina and S. Floyd. Measuring the evolution of transport protocols in the internet. In *Comput. Commun. Rev., vol. 35, no. 2, pp. 3751*, April 2005.

[3] M. Cudak A. Talukdar and A. Ghosh. "Streaming Video Capacities of LTE Air-Interface". IEEE International Conference on Communications (ICC), pages 1–5, 2010.

[4] Prashanth Aravinda Kumar Acharya, Ashish Sharma, Elizabeth M Belding, Kevin C Almeroth, and Konstantina Papagiannaki. Congestion-aware rate adaptation in wireless networks: A measurement-driven approach. In *2008 5th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, pages 1–9. IEEE, 2008.

[5] ABM Alim Al Islam and Vijay Raghunathan. End-to-end congestion control in wireless mesh networks using a neural network. In *2011 IEEE Wireless Communications and Networking Conference*, pages 677–682. IEEE, 2011.

[6] Yang Yang Alimi, Richard and Reinaldo Penno. Application-layer traffic optimization (alto) protocol. In *rfc7285*, 2014.

[7] Jeffrey G Andrews, Stefano Buzzi, Wan Choi, Stephen V Hanly, Angel Lozano, Anthony CK Soong, and Jianzhong Charlie Zhang. What will 5g be? *IEEE Journal on Selected Areas in Communications*, 32(6):1065–1082, 2014.

[8] Apple. https://www.apple.com/mac/facetime/.

[9] Mario Baldi and Yoram Ofek. "End-to-end delay analysis of videoconferencing over packet-switched networks". Networking, IEEE/ACM Transactions on 8.4, pages 479–492, 2000.

[10] P. Barten. "contrast sensitivity of the human eye and its effects on image quality". SPIE Press, 1999.

[11] M. Baugher, D. McGrew, M. Naslund, E. Carrara, and K. Norrman. RFC3711, RTP: The secure real-time transport protocol (srtp), 2004.

[12] J. Bergquist. ”resolution and contrast requirements on mobile displays for different applications in varying luminous environments”. 2nd Int Symp Nanovision Sceince, pp. 143-145, 2005.

[13] D. Bertsekas and R. G. Gallager. Data networks(2nd edition). prentice hall. 1992.

[14] Giuseppe Bianchi. Performance analysis of the ieee 802.11 distributed coordination function. In *Selected Areas in Communications, IEEE Journal*, 18.3: 535-547. 2000.

[15] Luciano Bononi, Marco Conti, and Enrico Gregori. Runtime optimization of ieee 802.11 wireless lans performance. *IEEE Transactions on Parallel and Distributed Systems*, 15(1):66–80, 2004.

[16] Y. C. Chen C. M. Chen, C. W. Lin. Cross-layer packet retry limit adaptation for video transport over wireless lans. In *IEEE transactions on circuits and systems for video technology, VOL. 20, NO. 11*, November 2010.

[17] Y. C. Chen C. M. Chen, C. W. Lin. Packet scheduling for video streaming over wireless with content-aware packet retry limit. In *In IEEE 8th Workshop on Multimedia Signal Processing*, Oct 2006.

[18] F. W. Campbell and J. G. Robson. ”application of fourier analysis to the visibility of gratings”. J. of Physiology, vol. 197, no. 3, pp. 551566, Aug. 1968.

[19] W. Chen, L. Ma, G. Sternberg, Y. Reznik, and C.-C. Shen. User-aware dash over wi-fi. In *International Conference on Computing, Networking and Communications (ICNC)*, 2015.

[20] Wei Chen, Liangping Ma, and Chien-Chung Shen. Congestion-aware mac layer adaptation to improve video teleconferencing over wi-fi. In *Proceedings of the 6th ACM Multimedia Systems Conference*, pages 130–141. ACM, 2015.

[21] Wei Chen, Liangping Ma, and Chien-Chung Shen. Congestion-aware mac layer adaptation to improve video telephony over wi-fi. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 12(5s):83, 2016.

[22] Wen-Tsuen Chen, Bo-Bin Jian, and Shou-Chih Lo. An adaptive retransmission scheme with qos support for the ieee 802.11 mac enhancement. In *Vehicular Technology Conference, 2002. VTC Spring 2002. IEEE 55th*, volume 1, pages 70–74. IEEE, 2002.

[23] Y.-C. Chen, E. Nahum, R. J. Gibbens, and D. Towsley. Measuring cellular networks: characterizing 3G, 4G, and path diversity. In *Annual Conference of International Technology Alliance*, 2012.

[24] Sunghyun Choi, Javier Del Prado, Stefan Mangold, et al. Ieee 802.11 e contention-based channel access (edcf) performance evaluation. In *Communications, 2003. ICC'03. IEEE International Conference on*, volume 2, pages 1151–1156. IEEE, 2003.

[25] M Chuah and Wei Chen. Cooperative multichannel mac (commac) for cognitive radio networks. In *Vehicular Technology Conference Fall (VTC 2010-Fall), 2010 IEEE 72nd*, pages 1–5. IEEE, 2010.

[26] Cisco. Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update. 2014-2019 White Paper, February 3, 2015.

[27] C. Dickie, R. Vertegaal, C. Sohn, and D. Cheng. Eyelook: using attention to facilitate mobile media consumption. In *ACM Symposium on User Interface Software and Technology*, pages 103–106, 2005.

[28] N. A. Dodgson. Variation and extrema of human interpupillary distance. In *Proc. SPIE*, volume 5291, pages 36–46, 2004.

[29] J. Dostal, P. O. Kristensson, and A. Quigley. Estimating and using absolute and relative viewing distance in interactive systems. *Pervasive and Mobile Computing 10 (2014) 173186*, 10:173186, Feb. 2014. Available online July 2012.

[30] Y. A. Reznik *et al.* User-adaptive mobile video streaming. In *Visual Communications and Image Processing (VCIP)*, 2012.

[31] A. M. Mood et al. "introduction to the theory of statistics, third edition". McGraw Hill, page New York, 1974.

[32] C. Liu et al. "rate adaptation for adaptive http streaming". Proceedings of the second annual ACM conference on Multimedia systems, pages 169–174, 2011.

[33] G. Tian et al. "on adaptive http streaming to mobile devices". Packet Video Workshop (PV), 2013 20th International, pages 1–8, 2013.

[34] M. Li et al. "playout buffer and rate optimization for streaming over ieee 802.11 wireless networks". volume 5 of *ACM Transactions on Multimedia Computing, Communications and Applications (TOMCCAP)*, Aug. 2009.

[35] S. Akhshabi et al. "an experimental evaluation of rate-adaptation algorithms in adaptive streaming over http". Proceedings of the second annual ACM conference on Multimedia systems, February 23-25, 2011.

[36] S. Gouache et al. "distributed & adaptive http streaming". Proceedings of IEEE International Conference on Multimedia and Expo (ICME), pages 1–6, Jul. 2011.

[37] T. C. Thang et al. "adaptive streaming of audiovisual content using mpeg dash". IEEE Trans. on Consumer Electronics, pages 78–85, 2012.

[38] T. Wirth et al. "advanced downlink lte radio resource management for http-streaming". Proceedings of the 20th ACM international conference on Multimedia, pages 1037–1040, Oct. 2012.

[39] T.-Y. Huang et al. ”downton abbey without the hiccups: Buffer-based rate adaptation for http video streaming”. Proceedings of the 2013 ACM SIGCOMM workshop on Future human-centric multimedia networking, pages 9–14, 2013.

[40] W. Pu et al. ”video adaptation proxy for wireless dynamic adaptive streaming over http”. Packet Video Workshop (PV), pages 65–70, 2012.

[41] S Floyd, M Handley, J Padhye, and J Widmer. Rfc 5348: Tcp friendly rate control (tfrc): Protocol specification. 2008.

[42] S. Floyd, M. Handley, J. Padhye, and J. Widmer. RFC5348, TCP friendly rate control (TFRC): Protocol specification, 2008.

[43] M. Franceschetti and R. Meester. Random networks for communication. In *Cambridge University Press*, 2008.

[44] G. S. Rubin G. E. Legge and A. Luebker. ”psychophysics of reading. v. the role of contrast in normal vision”. Vision Research, vol. 27, no. 7, pp. 1165-1177, 1987.

[45] Omer Gurewitz, Vincenzo Mancuso, Jingpu Shi, and Edward W Knightly. Measurement and modeling of the origins of starvation of congestion-controlled flows in wireless mesh networks. *IEEE/ACM Transactions On Networking*, 17(6):1832–1845, 2009.

[46] etc H. Bobarshad, M. van der Schaar. Analytical modeling for delay-sensitive video over wlan. In *IEEE transactions on Multimedia, VOL. 14, NO. 2*, APRIL 2012.

[47] C. Harrison and A. K. Dey. Lean and zoom: proximity-aware user interface and content magnification. In *ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 507–510, 2008.

[48] IEEE. *Wireless LAN Medium Access control (MAC) and Physical Layer (PHY) Specification*. 2012.

[49] N. Sato C. Burmeister J. Ott, S. Wenger and J. Rey. ”RFC4585, extended RTP profile for real-time transport control protocol (RTCP)-based feedback (RTP/AVPF)”. 2006.

[50] Abhijith Jagannath. *Implementation and Analysis of User Adaptive Mobile Video Streaming using MPEG-DASH*. PhD thesis, UNIVERSITY OF TEXAS AT ARLINGTON, 2014.

[51] Hung-Chin Jang and Yu-Ti Su. A hybrid design framework for video streaming in ieee 802.11 e wireless network. In *22nd International Conference on Advanced Information Networking and Applications (aina 2008)*, pages 560–567. IEEE, 2008.

[52] J.Anshel. ”visual ergonomics in the workplace”. CRC Press, 2002.

[53] K. Khoshelham. Accuracy analysis of kinect depth data. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 38(5):133–138, 2010.

[54] Hyogon Kim, Sangki Yun, Heejo Lee, Inhye Kang, and Kyu-Young Choi. Wlc34-1: A simple congestion-resilient link adaptation algorithm for ieee 802.11 wlans. In *IEEE Globecom 2006*, pages 1–6. IEEE, 2006.

[55] DongJin Lee, Brian E Carpenter, and Nevil Brownlee. Media streaming observations: Trends in udp to tcp ratio. *International Journal on Advances in Systems and Measurements*, 3(3-4), 2010.

[56] An-Chih Li, Ting-Yu Lin, and Ching-Yi Tsai. Arc: joint adaptation of link rate and contention window for ieee 802.11 multi-rate wireless networks. In *2009 6th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, pages 1–9. IEEE, 2009.

[57] Qiong Li and Mihaela Van der Schaar. Providing adaptive qos to layered video over wireless local area networks through real-time retry limit adaptation. *Multimedia, IEEE Transactions on*, 6(2):278–290, 2004.

[58] Qiong Li and Mihaela Van der Schaar. Providing adaptive qos to layered video over wireless local area networks through real-time retry limit adaptation. *IEEE Transactions on Multimedia*, 6(2):278–290, 2004.

[59] Licode. http://lynckia.com/licode/. Available Online, 2014.

[60] Christian Lochert, Björn Scheuermann, and Martin Mauve. A survey on congestion control for mobile ad hoc networks. *Wireless Communications and Mobile Computing*, 7(5):655–676, 2007.

[61] H. Lundin, S. Holmer, and H. Alvestrand. A google congestion control algorithm for real-time communication on the world wide web. Internet-Draft, April 2012.

[62] P. Steenkiste M. H. Lu and T. Chen. A time-based adaptive retry stategy for video streaming in 802.11 wlans. In *Wireless Communications and Mobile Computing*, 2007.

[63] L. Ma. Video sequences from the experiments for eplf. with EPLF on: https://www.youtube.com/watch?v=yLrOediyv-8; with EPLF off: https://www.youtube.com/watch?v=gqgPyYuO6RE, April 2015.

[64] Liangping Ma, Wei Chen, Dharm Veer, Gregory Sternberg, Weimin Liu, and Yuriy Reznik. Early packet loss feedback for webrtc-based mobile video telephony over wi-fi. In *2015 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE, 2015.

[65] Liangping Ma, Dharm Veer, Wei Chen, Gregory Sternberg, Yuriy A Reznik, and Ralph A Neff. User adaptive transcoding for video teleconferencing. In *Image Processing (ICIP), 2015 IEEE International Conference on*, pages 2209–2213. IEEE, 2015.

[66] K. Michael and L. Charles. "visual acuity". volume Web Vision: http://webvision.med.utah.edu/book/part-viii-gabac-receptors/visual-acuity/, 5 June 2007.

[67] J. Movshon and L. Kiorpes. "analysis of the development of spatial contrast sensitivity in monkey and human infants". J. Opt. Soc. Am. A, vol. 5, no. 12, pp. 21662172, Dec. 1988.

[68] MPEG-DASH. https://en.wikipedia.org/wiki/Dynamic_Adaptive_Streaming_over_HTTP.

[69] A. K. Agrawala N. C. Tas, T. Nadeem. Making wireless networks moral. In *INFOCOM*, 2011.

[70] OpenCV. http://opencv.org/. Available Online, 2014.

[71] OPNET. https://www.riverbed.com/products/steelcentral/opnet.html?redirect=opnet.

[72] OPNET. System-in-the-loop (SITL) functionality. https://support.riverbed.com/bin/support/static/l22aqi15ovr9jrkucioi3hpols/html/lp17rh21cfml414c4v0a1hqfna/modeler/wwhelp/wwhimpl/js/html/wwhelp.htm#href=Tutorials/sitl_tut_1.html.

[73] J. Ott, S. Wenger, N. Sato, C. Burmeister, and J. Rey. RFC4585, extended RTP profile for real-time transport control protocol (RTCP)-based feedback (RTP/AVPF), 2006.

[74] D. A. Owens and K. Wolf-Kelly. ""near work, visual fatigue, and variations of oculomotor tonus". Investigative ophthalmology & visual science, 1987.

[75] Jitendra Padhye, Victor Firoiu, Don Towsley, and Jim Kurose. Modeling tcp throughput: A simple model and its empirical validation. *ACM SIGCOMM Computer Communication Review*, 28(4):303–314, 1998.

[76] Nadim Parvez, Anirban Mahanti, and Carey Williamson. An analytic throughput model for tcp newreno. *IEEE/ACM Transactions on Networking (TON)*, 18(2):448–461, 2010.

[77] Sumit Rangwala, Apoorva Jindal, Ki-Young Jang, Konstantinos Psounis, and Ramesh Govindan. Neighborhood-centric congestion control for multihop wireless mesh networks. *IEEE/ACM Transactions on Networking (TON)*, 19(6):1797–1810, 2011.

[78] Y. A. Reznik. "user-adaptive mobile video streaming using mpeg-dash". volume 8856 of *Proceedings of SPIE Optical Engineering and Applications*, page 88560J, 2013.

[79] T. Henderson etc S. Floyd, ICSI. In *RFC3782, The NewReno Modification to TCP's Fast Recovery Algorithm*, April 2004.

[80] M. Shemer S. Holmer and M. Paniconi. "Handling packet loss in WebRTC". IEEE ICIP, 2013.

[81] G. Pujolle S. Lohier, Y. G. Doudane. Mac-layer adaptation to improve tcp flow performance in 802.11 wireless networks. In *Proc. IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob),Montreal, Canada*, 19-21 June 2006.

[82] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RFC3550, RTP: A transport protocol for real-time applications, 2003.

[83] Amir Shadmand and Mohammad Shikh-Bahaei. Tcp dynamics and adaptive mac retry-limit aware link-layer adaptation over ieee 802.11 wlan. In *Communication Networks and Services Research Conference, 2009. CNSR'09. Seventh Annual*, pages 193–200. IEEE, 2009.

[84] Jatinder Pal Singh, Yan Li, and Nicholas Bambos. Channel state awareness based transmission power adaptation for efficient tcp dynamics in wireless networks. In *IEEE International Conference on Communications, 2005. ICC 2005. 2005*, volume 5, pages 3553–3559. IEEE, 2005.

[85] Jatinder Pal Singh, Yan Li, Nicholas Bambos, Ahmad Bahai, Bangnan Xu, and Gerd Zimmermann. Tcp performance dynamics and link-layer adaptation based optimization methods for wireless networks. *IEEE Transactions on Wireless Communications*, 6(5):1864–1879, 2007.

[86] H. Snellen. "probebuchstaben zur bestimmung der sehschrfe (sample letters to determine the visual acuity)". 1873.

[87] Kun Tan, Feng Jiang, Qian Zhang, and Xuemin Shen. Congestion control in multihop wireless networks. *IEEE Transactions on Vehicular Technology*, 56(2):863–873, 2007.

[88] R. Vanam and Y. Reznik. "improving the efficiency of video coding by using perceptual preprocessing filter". IEEE Data Compression Conference, 2013.

[89] Rahul Vanam, Louis Kerofsky, and Yuriy A. Reznik. Perceptual pre-processing filter for adaptive video on demand content delivery. In *IEEE International Conference on Image Processing (ICIP)*, pages 2537–2541, 2014.

[90] Rahul Vanam and Yuriy A. Reznik. Perceptual pre-processing filter for user-adaptive coding and delivery of visual information. In *Picture Coding Symposium (PCS)*, pages 426–429, 2013.

[91] NOAO W. Stevens. In *RFC2001, TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms*, January 1997.

[92] WebRTC. http://www.webrtc.org/. Available Online, 2014.

[93] Nan Song Wu Wei Cai, Liang Huang. Class e power amplifier for wireless medical sensor network. *International Journal of Enhanced Research in Science, Technology & Engineering*, 2016.

[94] NanSong Wu Wei Cai, Jie Gong. 2.4ghz class f power amplifier for wireless medical sensor network. *Proceedings of the 2 nd World Congress on New Technologies (NewTech'16) : Fundamentals and Applications*, 2016.

[95] Wikipedia. "snellen's chart", http://en.wikipedia.org/wiki/Snellen_chart.

[96] Starsky H.Y. Wong, Hao Yang, Songwu Lu, and Vaduvur Bharghavan. Robust rate adaptation for 802.11 wireless networks. In *Proceedings of the 12th annual international conference on Mobile computing and networking (MobiCom)*, pages 146–157, Los Angeles, Sep. 2006.

[97] Hongjiang Xiao, Yuping Wei, Qionghai Dai, and Wenli Xu. Congestion-adaptive and queue reservation scheme for delay-constrained video streaming over ieee 802.11 networks. In *Visual Information Engineering, 2008. VIE 2008. 5th International Conference on*, pages 420–425. IET, 2008.

[98] Yao Xiao, Yang Guan, Wei Chen, Chien-Chung Shen, and Leonard J Cimini. Location-aware cooperative routing in multihop wireless networks. In *2011 IEEE Wireless Communications and Networking Conference*, pages 761–766. IEEE, 2011.

[99] L. Xu and J. Helzer. "media streaming via tfrc: An analytical study of the impact of tfrc on user-perceived media quality". Proc. IEEE Infocom, 2006.

[100] J. Xue and C. W. Chen. "a study on perception of mobile video with surrounding contextual influences". Fourth International Workshop on Quality of Multimedia Experience, vol. 5, no. 7, pp. 248-253, Jul. 2012.

[101] J. E. Hue Y. Bababekova, M. Rosenfield and R. R. Huang. "font size and viewing distance of handheld smart phones". Optometry and Vision Science, vol. 88, no. 7, pp. 795-797, Jul. 2011.

[102] Maya Yajnik, Sue Moon, Jim Kurose, and Don Towsley. Measurement and modelling of the temporal dependence in packet loss. In *INFOCOM'99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 1, pages 345–352. IEEE, 1999.

[103] K.S. Wu Y.J. Chen and Q. Zhang. ”From QoS to QoE: A Tutorial on Video Quality Assessment”. Communications Surveys & Tutorials, IEEE, pages 1126–1165, 2015.

[104] Hong Zhou, Doan Hoang, P Nhan, and Vinod Mirchandani. Introducing feedback congestion control to a network with ieee 802.11 wireless lan. In *Wireless Telecommunications Symposium, 2004*, pages 61–66. IEEE, 2004.