CROSS-LAYER PERFORMANCE ANALYSIS AND ADAPTATION FOR REAL-TIME WIRELESS VIDEO STREAMING

BY MICHAEL T. LOIACONO

A dissertation submitted to the

Graduate School—New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

Graduate Program in Electrical and Computer Engineering

Written under the direction of

Dr. Wade Trappe

and approved by

_____

_____

_____

_____

_____

New Brunswick, New Jersey

January, 2011

ABSTRACT OF THE DISSERTATION

Cross-Layer Performance Analysis and Adaptation for Real-Time Wireless Video Streaming

by Michael T. Loiacono

Dissertation Director: Dr. Wade Trappe

The proliferation of wireless technology, mobile computing, and increasingly sophisticated video codecs have fueled the sharp increase in demand for user and machine-centric applications (such as IPTV, telemedicine, cyber-physical control, and surveillance) involving wireless video streaming. Unfortunately, today's state of the art local wireless technologies, such as IEEE 802.11, can be easily demonstrated to fail when subjected to the conditions associated with many of these real-world applications. This is because wireless video streaming involves complex relationships between the the video codec, the wireless PHY and MAC, and the application's (or user's) sensitivity to distortions in the video signal, and these relationships are not well understood or exploited by today's video streaming systems.

In this dissertation, we reveal and analyze several key obstacles to wireless video streaming, and propose a set of adaptive real-time cross-layer approaches to deal with them.

The dissertation begins with an overview of the major hurdles to wireless video streaming. A particular focus is on the instability of current link adaptation algorithms in CSMA/CA based wireless systems under the congested scenarios associated with wireless transmission of multiple simultaneous uplink video streams. Additionally, we focus on congestion control with respect to the fairness policies employed by the IEEE

802.11 channel access mechanism, and demonstrate that airtime fairness is preferred to throughput fairness for transmission of multiple simultaneous uplink video streams in a multi-rate environment. Several other obstacles are presented, including: the need for cross-layer approaches in wireless video streaming; problems with the use of MSE and PSNR in existing cross-layer approaches in today's literature; and the difficulty with perceptual, task-based video quality assessment in the context of an adaptive real-time cross-layer video streaming system.

After that, we propose several adaptive real-time cross-layer solutions to deal with these obstacles. They include: (i) airtime fair distributed cross-layer congestion control in multi-rate wireless environments; (ii) cross-layer link adaptation for wireless-video; (iii) video quality assessment in adaptive real-time cross-layer video streaming systems; and (iv) joint link adaptation and congestion control driven by user/task-centric resource allocation. We support the proposed algorithms through simulations, theory, and experiments with real wireless devices on which we have implemented our algorithms.

Acknowledgements

Firstly, I'd like to acknowledge and thank my advisor, Prof. Wade Trappe for his guidance throughout my graduate career, and for his flexibility in dealing with my status as a full-time worker in industry.

I am grateful to Dr. Justinian Rosca who has been so patient with me throughout my career at Siemens, and who has helped me see problems from new perspectives. His passion for scientific research is undeniable, and he cares deeply for his employees.

I wish to acknowledge Dr. Visvanathan Ramesh for always challenging us in his department at SCR to challenge and criticize our own work, and incessantly strive to push our work to new levels. I am also appreciative of his encouragement in reflecting on career paths, personal goals, and exploring new domains.

I would also like to acknowledge Prof. David Pheanis, who taught his CSE-421 class far more than "'Microprocessor System Design I"'. To this day, no single person has ever taught me more about hard work, discipline, self-respect, and respect for peers than Dr. Pheanis.

The professors who have served on my M.S. and Ph.D. committees at Rutgers (Professors Wade Trappe, Marco Gruteser, Yanyong Zhang, Kristin Dana, Dipankar Raychaudhuri, and Dr. Justinian Rosca) have provided me with valuable insights and suggestions, and for that I am thankful.

Finally, I am grateful to my friends and family, who have been a constant source of inspiration, encouragement, and healthy distraction.

Dedication

To my grandparents, who incessantly reminded me of the importance of an education. To my family, who provided me with the infrastructure for obtaining an education. To my wife, Christina, for her endless support.

Table of Contents

Chapter 1

Introduction

The proliferation of wireless technology, mobile computing, and increasingly sophisticated video codecs have fueled the sharp increase in demand for user and machine-centric applications (such as IPTV, telemedicine, cyber-physical control, and surveillance) involving wireless video streaming. Unfortunately, today's state of the art local wireless technologies, such as IEEE 802.11, can be easily demonstrated to fail when subjected to the conditions associated with many of these real-world applications. This is because wireless video streaming involves complex relationships between the the video codec, the wireless PHY and MAC, and the application's (or user's) sensitivity to distortions in the video signal, and these relationships are not well understood or exploited by today's video streaming systems.

In this dissertation, we reveal and analyze several key obstacles to wireless video streaming, and propose a set of adaptive real-time cross-layer approaches to deal with them.

The dissertation begins with an overview of the major hurdles to wireless video streaming. A particular focus is on the instability of current link adaptation algorithms in CSMA/CA based wireless systems under the congested scenarios associated with wireless transmission of multiple simultaneous uplink video streams. Additionally, we focus on congestion control with respect to the fairness policies employed by the IEEE 802.11 channel access mechanism, and demonstrate that airtime fairness is preferred to throughput fairness for transmission of multiple simultaneous uplink video streams in a multi-rate environment. Several other obstacles are presented, including: the need for cross-layer approaches in wireless video streaming; problems with the use of MSE and PSNR in existing cross-layer approaches in today's literature; and the difficulty with

perceptual, task-based video quality assessment in the context of an adaptive real-time cross-layer video streaming system.

After that, we propose several adaptive real-time cross-layer solutions to deal with these obstacles. They include: (i) airtime fair distributed cross-layer congestion control in multi-rate wireless environments; (ii) cross-layer link adaptation for wireless-video; (iii) video quality assessment in adaptive real-time cross-layer video streaming systems; and (iv) joint link adaptation and congestion control driven by user/task-centric resource allocation. We support the proposed algorithms through simulations, theory, and experiments with real wireless devices on which we have implemented our algorithms.

In Chapter 2, we give an overview of several key obstacles to wireless video streaming. First, we introduce the performance anomaly of current link adaptation algorithms in CSMA/CA-based wireless systems under the congested scenarios associated with wireless transmission of multiple simultaneous uplink video streams. This obstacle is so critical to wireless video streaming that we devote an entire chapter to studying it (Chapter 3), and an entire chapter to solving it (Chapter 5). After that, we focus on congestion control with respect to the fairness policies employed by wireless CSMA/CA channel access mechanisms. We demonstrate the dangers of using throughput-fairness-based channel access (such as the one found in the IEEE 802.11 DCF) for video streaming scenarios, and show that it is preferable to apply an airtime-fair channel sharing policy with a cross-layer approach. Third, we motivate the need for cross-layer approaches to wireless video streaming. After that, we discuss the misuse of MSE and PSNR in today's cross-layer approaches. Finally, we discuss the difficulty with true perceptual video quality assessment in the context of an adaptive real-time cross-layer video streaming system.

In Chapter 3, we show that current link adaptation schemes for CSMA/CA based wireless systems such as IEEE 802.11 exhibit sudden and severe drops in throughput when subjected to the congested conditions associated with wireless video streaming. This chapter provides an analysis of the factors that lead to the poor performance of current link adaptation schemes in real multi-rate environments. We show that current

link adaptation schemes fail because they do not differentiate between poor channel conditions and collisions as the source of transmission failures, and consequently invoke improper responses that cascade to dramatic throughput degradation. We support the analysis through experimentation with real data from a wireless video surveillance application, and provide recommendations for the next generation of wireless CSMA/CA link adaptation schemes (which are discussed in Chapter 6).

In Chapter 4, we propose a distributed cross-layer congestion control algorithm that provides enhanced quality of service and reliable operation for real-time uplink wireless video applications. Such applications are characterized by many wireless devices transmitting video at various physical (PHY) rates over a relatively congested channel. Unfortunately, todays off-the shelf IEEE 802.11 equipment can be easily demonstrated to suffer catastrophic failure when subject to these conditions let alone provide acceptable perceptual quality to the user. We show that in order to remedy these issues, it is preferable to apply an airtime-fair channel sharing policy with a cross-layer approach. The idea is to use a fast frame-by-frame control loop in the carrier sense multiple access/collision avoidance (CSMA/CA)-based medium access control (MAC) layer while simultaneously exploiting the powerful control loop gain attainable by performing source-rate adaptation in the application layer. We support the proposed algorithm through both simulation and experimentation with various channel and PHY rate scenarios using real wireless cameras.

In Chapter 5, we propose a new class of link adaptation algorithms designed to accommodate the unique requirements of wireless video transmission. This can be viewed as an instance from a class of cross-layer adaptive algorithms which use a perceptual video quality metric as the decision statistic. Current link adaptation algorithms for IEEE 802.11 WLANs exhibit behavior which makes them unsuitable for transmission of multiple simultaneous real-time uplink video streams. First, these algorithms do not consider the properties of the video codec, and hence, are unaware of the impact of PHY rate selection on the perceptual quality of the received video. Second, they do not differentiate between channel errors and collisions, and hence severely malfunction when the collision probability is non-negligible. We propose a link adaptation strategy

that not only optimizes the perceptual quality of the received video, but also maintains network stability by preventing catastrophic failure due to collisions. We show that switching to a lower PHY rate improves the SNR/BER performance, but increases channel contention (and hence the collision probability). Then, we use this information plus knowledge of the video codec and network transport protocol to estimate the received perceptual video quality at the current and adjacent PHY rates. The PHY rate that yields the best perceptual quality is chosen for each Group of Pictures (GOP). We support the proposed algorithm through both simulations and experiments with real wireless devices on which we have implemented our algorithm.

In Chapter 6, we explore video quality assessment in the context of real-time adaptive video streaming systems. With the evolution towards user- and machine-centric wireless video streaming applications underway, it is becoming progressively more important to ensure that distortions in the received video signal have minimal impact on the end-use application or task. This is in contrast to traditional QoS, which aims to optimize system and packet-level performance, but not necessarily perceptual or task-based video quality. In order to drive towards this goal, one of the most important evolutions to modern video streaming systems will be the adaptive allocation of both wireless and video coding resources at the transmitter according to a cost function designed to optimize some task- or user-centric measure of the received video quality. Today's cross-layer systems are based on mathematically simple, but perceptually inaccurate distortion metrics like MSE/PSNR. Perhaps this is not surprising, since aside from the great difficulty in designing perceptually accurate video quality metrics themselves, there are a number of other challenges in using such a metric in a real-time adaptive system. They include: (1) estimating the quality metric at the encoder with no direct access to the received video signal; (2) predicting how the perceptual quality would change if some adjustment is made to the instantiation of the parameter space either in the video codec or in the wireless PHY/MAC; and (3) how to balance complexity with accuracy for suitability in a real-time system with potentially limited computation and memory resources. We begin with an overview of distortions found

in digital videos. After that, we present the state of the art in video quality assessment, and investigate how well perceptually-oriented video quality metrics can predict the performance of task-specific applications with non-human observers, such as object tracking. Finally, we propose solutions to issues (1) and (2) listed above (item 3 (balancing complexity with accuracy) is covered implicitly).

In Chapter 7, we propose a resource control strategy which utilizes a synergy between the material in Chapters 4-6 in order to simultaneously perform link adaptation and congestion control according to a task-oriented objective function. The CLLA algorithm presented in Chapter 5 forms the basis for the resource control strategy proposed in this chapter. The first extension to CLLA is to allocate resources based on user-centric and task-centric criteria, rather than MSE. The second extension to CLLA is to consider scenarios where the client performs bitrate adaptation (for example, in response to a change in the PHY rate). By leveraging the CLC algorithm proposed in Chapter 4, we can not only consider the case where bitrate adaptation is performed, but the case where bitrate adaptation is performed in such a way that congestion is controlled via an airtime fairness policy. The end result is a joint link adaptation and congestion control strategy driven by task-centric resource allocation. We support the proposed algorithm through simulations and experiments with real wireless cameras on which we have implemented our algorithm.

Chapter 2

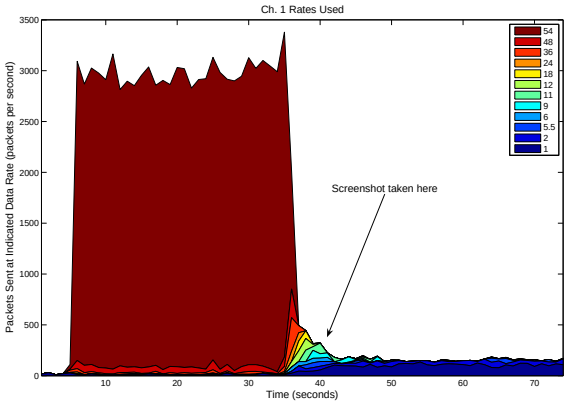
Obstacles to Wireless Video Streaming

2.1 Introduction

In this Chapter, we give an overview of several key obstacles to wireless video streaming. This chapter serves as an introduction and motivation for the solutions we propose later in the dissertation. First, we introduce the performance anomaly of current link adaptation algorithms in CSMA/CA-based wireless systems under the congested scenarios associated with wireless transmission of multiple simultaneous uplink video streams. This obstacle is so critical to wireless video streaming that we devote an entire chapter to studying it (Chapter 3), and portions of Chapters 4, 5, and 7 to solving it. After that, we focus on congestion control with respect to the fairness policies employed by wireless CSMA/CA channel access mechanisms. We demonstrate the dangers of using throughput-fairness-based channel access (such as the one found in the IEEE 802.11 DCF) for video streaming scenarios, and show that it is preferable to apply an airtime-fair channel sharing policy with a cross-layer approach. Third, we motivate the need for cross-layer approaches to wireless video streaming. After that, we discuss the misuse of MSE and PSNR in today's cross-layer approaches. Finally, we discuss the difficulty with true perceptual video quality assessment in the context of an adaptive real-time cross-layer video streaming system.

2.2 Performance Anomaly of Packet-Based Link Adaptation in Congested Scenarios

One notable flaw of current local area wireless technologies is their use of simplistic link adaptation mechanisms, which can cause a wireless application to experience sudden,



(a)

(b)

Figure 2.1: Video streaming application. (a) Prior to t=35, the system accommodates $\approx$ 3000 packets per second, with nearly all packets being transmitted at 54Mbps. At t=35, the system begins to crash. Note the selection of decreasing PHY rates, and also the steep drop in the number of packets per second. (b) Screen-shot of normal (left) and degraded (right) video streams.

severe and often irrecoverable drops in throughput. Figure 2.1 shows a real-world example of catastrophic failure where multiple cameras are streaming uplink video, and the system suddenly suffers a sharp decrease in throughput associated with a rapid succession of down-rates in the link adaptation mechanism . Figure 2.1(a) shows the rapid rate at which the network crashes with respect to the decreasing PHY rates selected by the rate adaptation mechanisms. The total system packets per second indicates that system throughput plummets as lower PHY rates are selected, and hence, video quality is degraded.

This effect, which throughout this dissertation we shall call the "Snowball Effect" (due to its characteristic behavior witnessed in laboratory experiments), is a serious hurdle for the deployment and adoption of local wireless technologies for a broad range

of applications and, consequently, a thorough understanding of this effect and the performance of current 802.11 link adaptation schemes is necessary in order to remedy this issue. We detail this problem in Chapter 3.

2.3 Channel Access and Congestion Control Issues: Airtime vs. Throughput Fairness

It is well known that per-station throughput in a WLAN multirate basic service set (BSS) tends to move quickly toward the rate of the lowest rate station in the BSS [1]. Channel access among competing stations is managed such that stations divide the available throughput according to CSMA/CA, which implements a throughput fairness policy. We illustrate this in Figure 2.2, which shows the effect of 1 out of 10 stations in a BSS dropping its PHY rate from 11 to 2 Mb/s. Throughput fairness results in decreased aggregate system throughput (from about 5.5 to 4 Mbps in Figure 2.2), no throughput guarantees above a minimum level, and a possibility of congestion with the added danger of completely dropping some video flows. Consequently, if a station switches its PHY rate, then we want to adapt that station's channel share and video source rate accordingly in order to protect the system from catastrophic behavior. We need a balanced allocation of airtime such that a slow station does not consume more airtime than a fast station. We refer to this principle as airtime fairness, and propose a cross-layer solution to this obstacle in Chapter 4.

2.4 The Problem of Video Quality Assessment in Cross-Layer Approaches

2.4.1 The Need for Cross-Layer Approaches

Modern video codecs such as H.264 offer a wide variety of tunable parameters both on the encoder and the decoder (see [2] for an overview). The values of these parameters greatly influence how a perturbation in the received video signal (e.g. due to packet loss in the wireless channel) will manifest itself to the end-user, and they can even influence the performance of the wireless network itself (for example, if the video bitrate is set

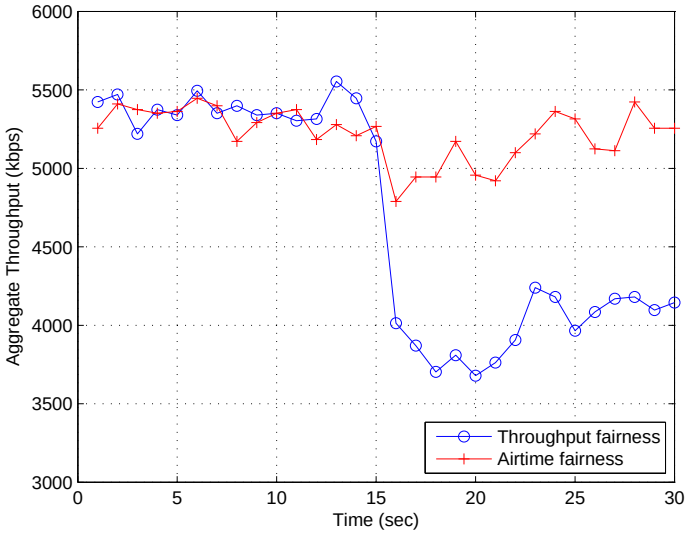


Figure 2.2: Throughput vs. Airtime Fairness. The blue line shows the aggregate system throughput under a throughput fairness regime, and the orange line is aggregate system throughput under an airtime fairness regime. Initially, all 10 stations are at 11 Mb/s. At t=15, 1 of 10 stations switches from 11 to 2 Mb/s. The aggregate system throughput is reduced in the case of throughput fairness.

higher than the channel capacity of the network, etc.). Similarly, the PHY and MAC layers of many wireless systems are highly configurable, and the choice of parameter settings in this space can greatly influence the reliability, stability, throughput, and overall performance of the wireless system (link adaptation is one clear example of this). Furthermore, the neurophysiology and psychophysics of the Human Visual System (HVS) determine how the end-user will perceive the received video signal. Therefore, common sense dictates that HVS modeling, video quality assessment, video coding, and wireless networking/communication should be jointly considered in the engineering of wireless video streaming systems, and thus a cross-layer approach is needed.

2.4.2 The Problem with MSE and PSNR

In the past decade, there has been a surge of cross-layer approaches in the literature. This literature has generally been contributed by researchers from three different disciplines: (1) networking/communication; (2) video coding; and (3) theorists. Although there has been a fair amount of cross-pollination across these three disciplines, there

unfortunately has been little, if any, cross-pollination with the HVS-related video quality assessment (VQA) literature. Namely, mean squared error (MSE), or equivalently, peak signal to noise ratio (PSNR), is the de facto metric around which today's cross-layer video streaming literature is designed and evaluated. MSE is a spatial distortion metric, and therefore does not capture any of the temporal distortions found in digital videos. Additionally, it is well known, and easily demonstrated that MSE does not correlate well with human perception (c.f. Figure 2.3) [3, 4, 5]. Yet, the literature continues to use MSE as the de facto metric (see [6, 7, 8, 9, 10, 11, 12, 13] for a sampling of the literature). Perhaps this is not surprising since MSE is a mathematically simple metric, while true perceptual VQA is a challenging, open research area. Additionally, even if one has a perceptually accurate VQA metric, there are a number of challenges in making that metric useful in the context of a real-time adaptive system.

2.4.3 Barriers to Perceptual Video Quality Assessment in the Context of an Adaptive Real-Time Cross-Layer System

Assuming one has a perceptually accurate VQA metric, there are a number of challenges in making that metric useful in the context of a real-time adaptive system. First, it is not at all likely that the encoder will have access to the received video signal. This seems to preclude the use of a full-reference VQA metric. Second, even if one could somehow accurately assess the current received video quality at the encoder, there is yet another challenge: predicting how the perceptual quality would change if some adjustment was made to the instantiation of the parameter space in either the video codec, the wireless PHY/MAC, or both. Third, real-time video streaming systems may have potentially limited computation and memory resources, and therefore, the tradeoff between complexity and accuracy must be carefully considered. These issues, as well as their solutions are the topic of Chapter 6.

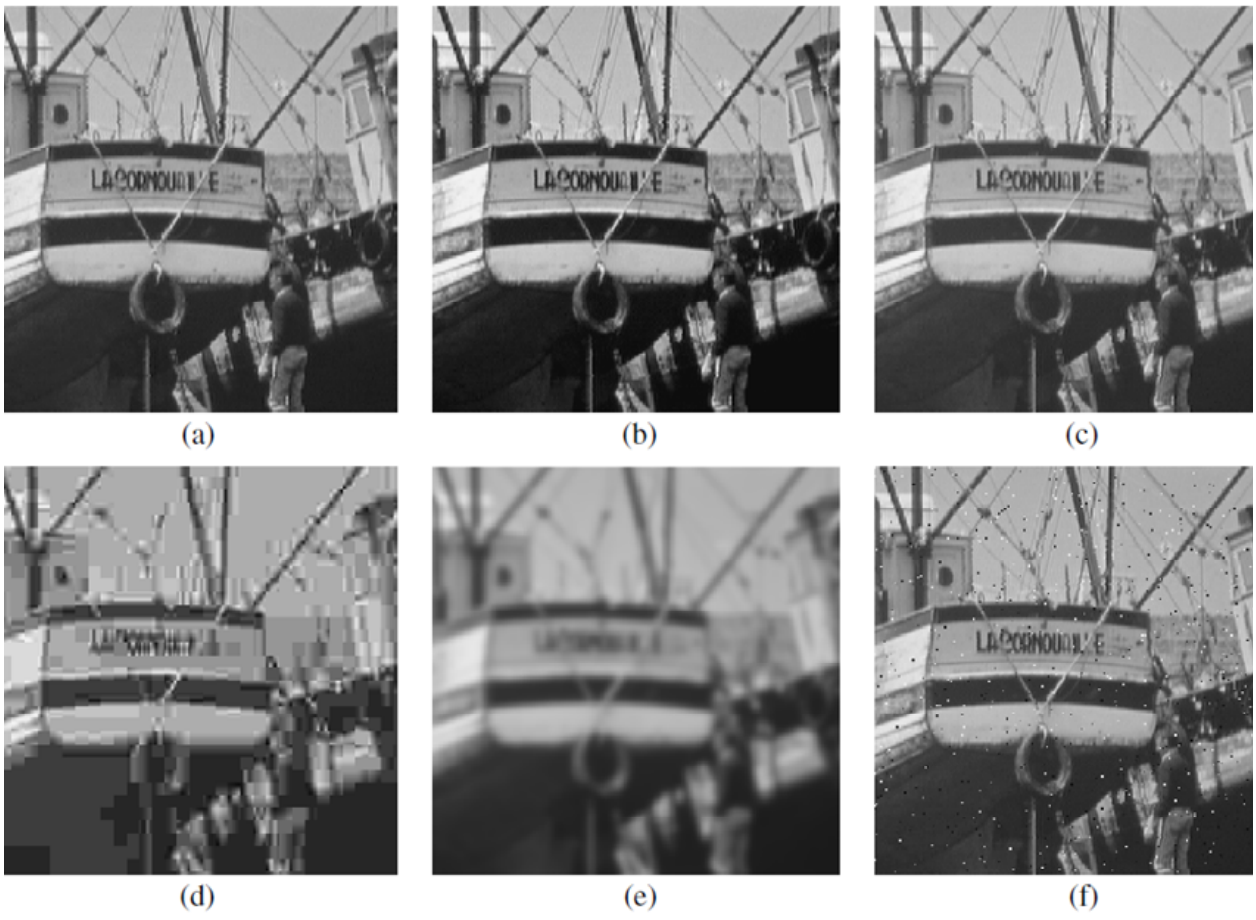


Figure 2.3: Different distortions can yield vastly different perceptual qualities, but identical MSE distortions (here, MSE=210). (a) Original image. (b) Contrast-stretched. (c) Mean-shifted. (d) JPEG compressed. (e) Blurred. (f) Impulsive noise (salt-pepper) [4].

Chapter 3

Performance Anomaly of Packet-Statistic-Based Link Adaptation

3.1 Introduction

Commercial wireless technologies, such as 802.11, have led to significant enhancements in consumer access and connectivity. Unfortunately, the current generation of 802.11 equipment can be easily demonstrated to fail when subjected to adverse conditions associated with many real applications (especially voice/video over WLAN). One notable flaw of current local area wireless technologies is their use of simplistic link adaptation mechanisms, which can cause a wireless application to experience sudden, severe and often irrecoverable drops in throughput. This effect, which throughout this dissertation we shall call the "Snowball Effect" (due to its characteristic behavior witnessed in laboratory experiments), is a serious hurdle for the deployment and adoption of local wireless technologies for a broad range of applications and, consequently, a thorough understanding of this effect and the performance of current 802.11 link adaptation schemes is necessary in order to remedy this issue.

Link (or equivalently, rate) adaptation refers to techniques for dynamically and adaptively choosing modulation schemes according to channel conditions. The modulation schemes that yield high PHY rates (e.g. 54Mbps) are fragile and susceptible to corruption from interference. On the other hand, more resilient modulation schemes can be employed at the expense of lower PHY rates. There is clear need for such techniques in order to use the wireless channels effectively.

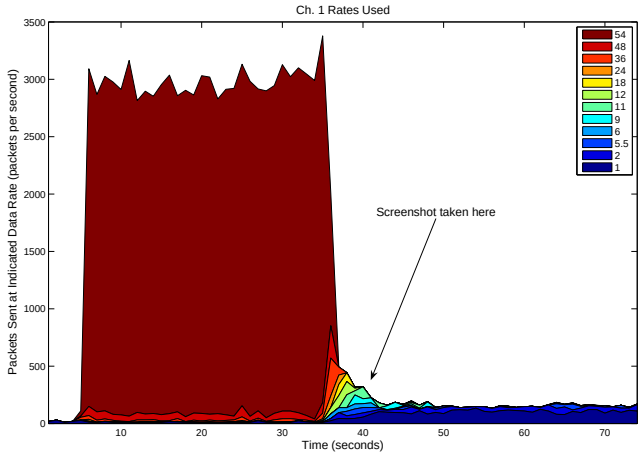
In order to understand the problems associated with rate adaptation schemes, it is important to realize that transmission failures occur for two reasons, collisions and poor channels, and that the current generation of rate adaptation algorithms typically

address one or the other of these issues, and generally not both. A consequence of such a design is that the responses taken may not be suitable for alleviating the actual cause of transmission failure and might, in fact, worsen the problem. In particular, an algorithm that assumes errors are due to poor channel conditions will severely malfunction if the errors are actually due to collisions, and vice-versa.

Indeed, the seriousness of this problem is indicated by several recent papers (e.g. [14, 15, 16]) where the authors discuss ways to differentiate between the two types of errors. Although there are several papers that propose such solutions, there are few if any papers that diagnose the cause of the problem itself. We believe that in order for rate adaptation schemes to overcome this problem, it is necessary to thoroughly understand the details of the problem itself. In this chapter, we set out to thoroughly understand these details, and ultimately arrive at the principal requirements of next generation rate adaptation schemes.

Our analysis is based on both simulation studies and experimental evidence. Throughout our analysis, we have sought to explore this phenomena under real application conditions, and thus we have setup a multi-camera video surveillance application in our laboratory for the purpose of diagnosing the Snowball Effect. Through our analysis, we have found that the underlying cause of the Snowball Effect is the fact that the current generation of rate adaptation schemes do not distinguish between the causes for transmission errors. In particular, when an adaptive algorithm assumes that all transmission errors are due to poor channel conditions when the true cause is collisions, the algorithm will invoke an improper response that leads to a chain reaction of events that ultimately causes the system throughput to drop into a downward spiral. Under certain conditions, the system may recover from the downward spiral on its own, and additionally, the failure may be predicted using a statistic that we derived.

The rest of this chapter is organized as follows: Section 3.2 defines the Snowball Effect through a complete real-world example. After that, Section 3.3 gives a formal analysis of the rate control problem. In Section 3.4, both simulations and real experiments are used to support the claims made in the Formal Analysis, and also to provide additional perspectives of the Snowball Effect. Then, Section 3.5 discusses related work.



(a)

(b)

Figure 3.1: Video streaming application. (a) Prior to t=35, the system accommodates $\approx$ 3000 packets per second, with nearly all packets being transmitted at 54Mbps. At t=35, the system begins to crash. Note the selection of decreasing PHY rates, and also the steep drop in the number of packets per second. (b) Screen-shot of normal (left) and Snowball-degraded (right) video streams.

Finally, we conclude the paper in Section 3.6.

Although consequences of the Snowball Effect are clear (severe and sustained drops in system throughput, possibly unrecoverable system failure, and loss of functionality for the end user), the dynamics of this anomaly are not so clear.

3.2 Complete Example - Video Streaming in a Multi-Camera Environment

In this section, we define and illustrate the Snowball Effect in a real-world setting. We have used hardware from different vendors for transmission of voice, video, and best

effort data traffic over wireless LAN, and in all cases, the voice and video applications suffered seriously because of manifestations of the problem. For clarity, we only focus on the video streaming example.

3.2.1 Experimental Setup

Our real-world experimental setup is based on a wireless, real-time, surveillance/security system. We consider the following scenario: Six to ten Axis 207w[1] cameras stream video (3Mbps Constant Bitrate MPEG4/RTP) wirelessly (802.11g) to the central security center (a Dell OptiPlex GX280) via a wireless access point (Linksys WRT54G). The GX280 processes the video in real-time to perform actions such as: face detection, person tracking, forbidden zone detection, etc. Such systems are commonly used to secure airports, subway systems, and other civil infrastructure.

In our lab, the wireless signal is exposed to regular interference from nearby APs, lights, microwaves, etc., but overall, the channel is good enough for 54Mbps transmission. Since there are so many contending wireless stations, we expect the main cause of transmission failures to be collisions, not poor channel conditions. Given this setup, our system is at high risk for the Snowball Effect (since the rate adaptation mechanisms will react assuming that the failed transmissions are due to a bad channel while they are really due to collisions).

We monitored a plethora of system statistics in real-time using AiroPeek SE [17] - a packet sniffing application which we customized to track features of interest using its SDK. The statistics were available in real time, but they were also archived for later use. Additionally, we recorded the received live video streams on the OptiPlex GX280.

The experimental procedure itself was as follows: We started 6 cameras, and simultaneously used AiroPeek to sniff the ether while the screen capture utility recorded the live video. We let the system run until the Snowball Effect occurred. We ran the experiment several times, varying several camera parameters like the video source rate.

[1]Interestingly, the Axis cameras' release notes caution of a potential downfall when trying to simultaneously use a large number of cameras, but no remedy is suggested. The problems we describe manifest with many other hardware components and configurations, and are not specific to this particular setup or vendor.

Table 3.1: Network Configuration

Layer	Parameter	Value
Application	Codec	MPEG4
	Traffic Direction	One-way (Uplink)
	Source Rate	3 Mbps
	Total Packet Size (bytes)	1536
Transport	Protocol	IP/UDP/RTP
	Header (bytes)	20+8+12
Link/MAC	Protocol	802.11g
	Header (bytes)	8+24 (link+MAC)
	Link Adaptation	Depends on experiment
	Retry Limit	3
	Queue Length	1000 packets + expiration logic
	Fragmentation	Disabled
	RTS/CTS	Disabled
Physical	Protocol	802.11g
	Header (bytes)	8 byte preamble
	Max Rate (bps)	54 Mbps
	Base Rate (bps)	1 Mbps

Table 3.2: 802.11g PHY/MAC Parameters

Parameter	Value	Comments
Slot time	9 μs	Idle slot time (σ)
SIFS	10 μs	Single Inter-Frame Spacing time
DIFS	28 μs	$SIFS + 2 \cdot \sigma$
CW_{min}	16	Minimum contention window
m	8	Backoff stages
CW_{max}	1024	$2^m CW_{min}$
ACK Packet Size	14 bytes	Size of an ACK

For the sake of brevity, we only present results from the experimental configuration detailed in Tables 3.1 and 3.2.

Additionally, unless otherwise noted, we endorse the following facts/assumptions:

1. Prob[there is at least one packet in a station's queue] is near 1, but not exactly 1.
2. A wireless packet transmission can fail for two reasons: collision or bad channel.
3. There are no hidden/exposed nodes.

3.2.2 The Snowball Effect - Experimental Observations

When we operate the network as described in the previous section, it suffers a catastrophic failure after a short amount of time. This catastrophic failure is characterized by:

- Lower 802.11 PHY rates (Figure 3.1)
- Sudden, steep drop in throughput/goodput (Figure 3.2)

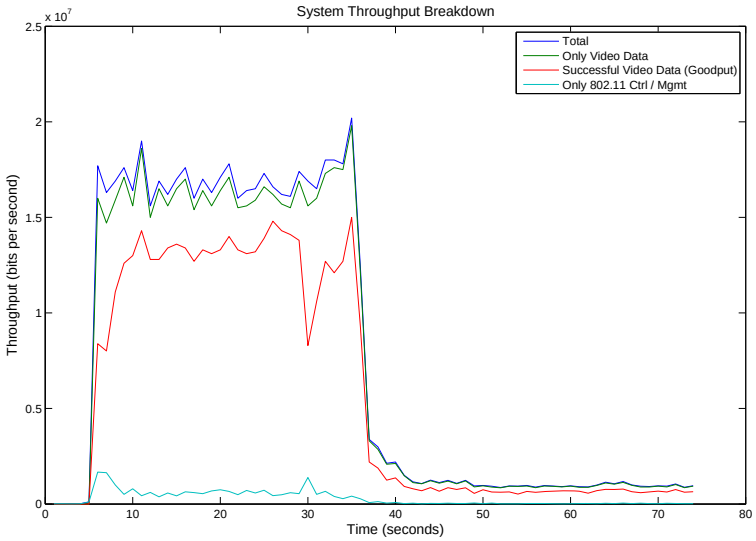


Figure 3.2: Prior to t=35, the system obtains nearly 18Mbps total throughput. After t=35, the system crashes and throughput is driven nearly to zero. In addition to throughput, the figure shows the breakdown of throughput into CTRL/MGMT traffic and video traffic, and it also shows goodput.

- Long interruption of some of the video streams or voice calls
- Network may not recover on its own.

To the end user, this translates to severely degraded video quality (Figure 3.1(b)), dropped calls (in the case of voice), and loss of application functionality. We call this the Snowball Effect, and we formally analyze it next.

3.3 Formal Analysis

The previous figures show that the system suddenly experiences serious performance degradation - goodput and PHY rates drop abruptly, video quality becomes very bad. What causes this sudden change?

Throughout the formal analysis presented below, we assume that rate adaptation algorithms trigger to lower PHY rates based on some combination of conditions on packet statistics such as increased delay, decreased goodput, increased interval between successfully transmitted packets, or increased number of retries. Of course, there are many types of rate adaptation algorithms, but they all fundamentally rely in some way

on these "trigger statistics" [16].

We will show that when a station in the BSS lowers its PHY rate, the global probability of a collision, $P[collision]$, increases, which in turn, causes the aforementioned rate adaptation "trigger" criteria to be satisfied. Therefore the rate adaptation mechanism lowers the PHY rate further, even though the channel conditions remain adequate for 54Mbps transport.

3.3.1 Normal Operation

Consider an initial set of parameters that characterizes each of N stations in a basic service set (BSS) under normal operation:

- λ_i (arrival rate for station i): Assume λ_i is deterministic and identical for all i ($\lambda_i = \lambda, \forall\ i \in N$) due to constant bitrate video
- μ_i (service rate for station i): General Distribution

For this generic queue structure (i.e. D/G/1/1000), these two parameters yield the probability that a newly arriving packet in station i finds the queue empty, call it q_i, $0 < q_i < 1$, $i \in N$. For now, let all stations transmit at the highest rate (i.e. 54Mbps), so $\mu_i = \mu$, $\forall\ i \in N$ Furthermore, denote one of these N stations as station j, so $\mu_j = \mu_i = \mu$, and $q_j = q_i = q$.

3.3.2 Initial Entry into the Downward Spiral

Now, assume that an event happens (e.g. burst of interference, etc.) which forces one of the N stations (say station j) to move to a lower PHY rate, say 36Mbps. We now have a new set of parameters for the low-rate station:

- $\lambda'_j = \lambda_i = \lambda, \forall\ i \in N$ (arrival rate is unchanged)
- μ'_j (new service rate for slow station), $\mu'_j < \mu_j$

The arrival rate does not change, but the decreased PHY rate causes the slow station's service rate to decrease to $\mu'_j < \mu_j$, and consequently, there is a reduced

probability, $q'_j < q_j$, that a newly arriving packet (in the slow station) will find the queue empty.

3.3.3 Global Effect of One Station Switching to a Lower PHY Rate

It is tempting to assume that this lower μ'_j corresponds only to the station operating at a reduced rate. However, in [1], it was shown that fast stations transmitting at 54Mbps obtain the same throughput as slow stations at 36Mbps [2] due to the throughput-fairness property of the 802.11 DCF. Therefore, this new μ'_j applies globally to *all* of the stations. Consequently, the reduced q'_j also applies to all stations. Thus, we have:

- $\lambda_i = \lambda, \forall\ i \in N$ (unchanged)
- $\mu_i = \mu'_j, \forall\ i \in N$ (global service rate reduction)

Theorem 1 *Decreasing the global service rate, μ increases $P[collision]$.*

Proof 1 *We will use some of the results in [18], which is an extension of Bianchi's results [19] for the case of unsaturated networks. Denote $\tilde{C}$ as the number of* contending *stations (i.e. the number of stations with at least one packet in their queue) in a discrete time step. Initially, assume all stations transmit at their maximum rate (54Mbps). Furthermore, denote $\tilde{C}_{avg}$ as the* average *number of contending stations in each time step:*

$$\tilde{C}_{avg} = N - \sum_{i=1}^{N} q_i \tag{3.1}$$

Now, let one of the N stations (station j) switch to a lower rate. Then, by our previous arguments, there is a global reduction in service rate (i.e. μ_i is reduced to μ'_j, $\forall i$, and hence, q_i is reduced to q'_j, $\forall i$). Then, by Equation 3.1, this means that $\tilde{C}_{avg}$ (and hence $\tilde{C}$) is likely to be larger now than it was when all stations were transmitting at 54Mbps and obtaining q_i.

[2] This effect will be amplified in 802.11n, where there is a wider range of PHY rates to choose from.

Recall the formula for calculating the collision probability under non-saturated conditions [18], $p(\tilde{C}) = 1 - [1 - \tau(\tilde{C})]^{\tilde{C}-1}$, where $\tau(\tilde{C})$ is the probability that one of the contending stations transmits in a given time step. By inspection of this equation, we see that the collision probability grows with $\tilde{C}$. But we just showed that $\tilde{C}$ grows with the number of low-rate stations in the BSS, and hence with decreasing service rate.

When the collision probability grows, we argue that there is an increased likelihood that the "trigger statistics" satisfy their criteria. For one example, consider the "retry" "trigger statistic". We just showed that lowering the PHY rate increases $P[collision]$. But increasing $P[collision]$ implicitly implies that there are more failed transmissions, on average. When transmissions fail, they are retransmitted until they either succeed or expire [20]. Thus, increasing $P[collision]$ also increases the average number of retries.

As a second example, consider the "delay" "trigger statistic". The delay for a station in an unsaturated 802.11g BSS is comprised of the following components: $D = D_{IFQ} + D_{MAC} + D_{TX}$, where D_{IFQ} is time spent waiting in the interface queue, D_{MAC} is time spent in the MAC (delay due to DCF contention, backoff, etc.), and D_{TX} is transmission time. Lowering the PHY rate obviously increases D_{TX}. Furthermore, as we just explained, the corresponding increase in $P[collision]$ implies an increase in the average number of retries. When a station suffers a failed transmission attempt, that station must increment its backoff stage before the transmission may be reattempted [20]. Thus, as the number of retries increases on average, we see a corresponding shift to higher backoff stages being used more frequently. Consequently, a packet spends more time doing backoff, thus increasing D_{MAC}. In fact, lowering the PHY rate of a station in the BSS hurts doubly bad since it will also cause the medium to be sensed busy more frequently by other stations, hence causing their backoff counters to take longer to reach zero (and increasing D_{MAC} even further). The sharp increases in D_{TX} and D_{MAC} mean that it takes longer for a packet to leave the transmission stage. Since a packet cannot leave the IFQ for the transmission stage until the previous packet leaves the transmission stage, this means that D_{IFQ} also increases.

Similar arguments, omitted for brevity, reveal that increasing $P[collision]$ causes goodput to decrease, and the interval between successfully transmitted packets to increase.

3.3.4 Completing the Circle

According to the fundamental fact stated in the beginning of this section, rate adaptation algorithms trigger to lower PHY rates based on some combination of conditions on the trigger statistics. We just showed that when a station lowers its PHY rate needlessly, it increases $P[collision]$, and hence, it promotes the trigger statistics to satisfy their criteria, hence causing stations' rate adaptation mechanisms to lower PHY rates even further. Therefore, we are back where we started, but at a lower PHY rate. This cycle continues until nearly all stations are using their lowest rate. At this point, catastrophic failure is imminent.

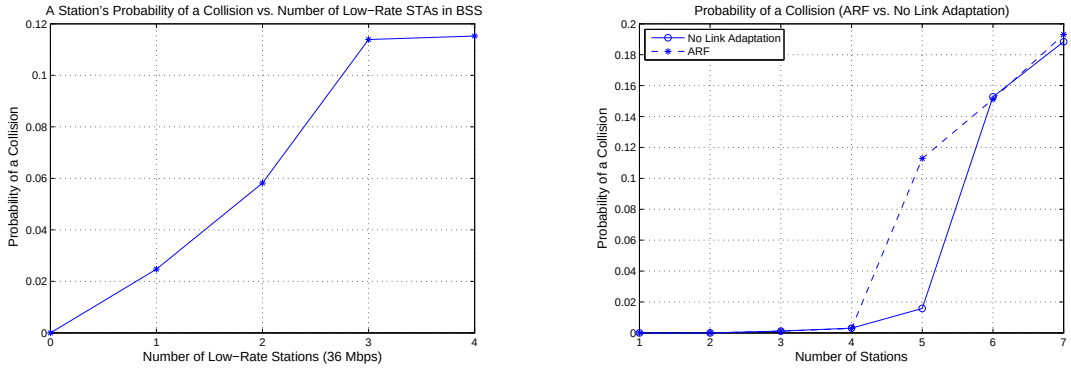
3.4 Experimental Analysis

In this section, both simulations and real experiments are used to support the claims made in the previous section. Furthermore, we provide additional perspectives of the Snowball Effect by using results from real experiments.

3.4.1 ns2 Simulation

Our ns2 [21] simulation environment is designed to replicate the parameters of our real experiments (as shown in Table 3.1). There are no interferers, and the SNR is perfect.

The goal of our first ns2 experiment is to validate the aforementioned theorem by studying the relationship between $P[collision]$ and the number of low-rate stations. We expect $P[collision]$ to increase with the number of low-rate stations. Indeed, Figure 3.3(a) shows $P[collision]$ for a station in the BSS vs. the number of low-rate (36Mbps) stations for a fixed number (5) of total stations in the BSS. We increased the number of low-rate stations from 0 to 4, and reported $P[collision]$ for the station that remained at a high rate (54Mbps). It is clear from the figure that increasing the number of low



(a) (b)

Figure 3.3: (a) Probability of collision versus number of low-rate stations; (b) Probability of collision for ARF link adaptation vs. no link adaptation.

rate stations increases $P[collision]$, thus validating our hypothesis.

The goal of our second experiment is also to validate the aforementioned hypothesis, but in a way that better resembles real-life rate adaptation. Namely, we set out to study the probability of a collision for a station in a BSS that uses ARF [22], and in a BSS that does not use any rate adaptation. The results, Figure 3.3(b), clearly show that for a given number of stations, when ARF is used, $P[collision]$ is higher than if ARF is turned off and all stations transmit at a fixed rate of 54Mbps. Since our simulation was configured with no interferers and a perfect SNR, the increase in $P[collision]$ is entirely due to ARF being fooled: it treats collisions as channel errors, and lowers the PHY rate needlessly - confirming our previous claims.

3.4.2 Snowball effect in real-lab WLAN video streaming

Multiple real-world experiments with video/voice over WLAN confirm the "Snowball" behavior. We first describe in more detail the results already shown in Figures 3.1 and 3.2. After that, we examine the time needed to recover from the "crashed" regime when one or more clients are intentionally shut down so as to free up bandwidth and ease contention.

Figure 3.1 shows a real-world example of catastrophic failure in a video over WLAN experiment. Figure 3.1(a) shows the rapid rate at which the network crashes with respect to the decreasing PHY rates selected by the rate adaptation mechanisms. The

total system packets per second indicates that system throughput plummets as lower PHY rates are selected, and hence, video quality is degraded. Figure 3.2 shows the relationship between throughput (total number of bits that are physically put on the air), broken down into video data and 802.11 Ctrl/Mgmt packets, and the corresponding goodput (subset of video data packets that are successfully acknowledged by the intended receiver). It is clear that after the network crashes (at t=35), the goodput is relegated to less than 1Mbps.

Recovering from Catastrophic Failure

Once the network suffers a catastrophic failure, what does it take to get the system back at a high rate? To investigate this question, we performed the following experiment:

1. Configure 6 Axis cameras according to Table 3.1.
2. Start all 6 cameras simultaneously.
3. When the network crashes, stop all but 1 of the cameras.
4. Start a timer and see how long it takes for the network (i.e. the 1 remaining camera) to recover back to its original PHY rate of 54Mbps, and source throughput of 3Mbps.

Then, we repeat this exact experiment, but instead of stopping all but one camera, we stop all but two cameras, etc. Figure 3.4 details our results. The captions under each figure show the number of active cameras after the crash and the average recovery times for each experiment. As expected, the recovery time for each experiment increases as we try to recover more and more cameras. In fact, apparently, we cannot recover four or more cameras (c.f. Figure 3.4(d)).

Predicting when the regime of operation is transitioning into a dangerous state

It is important to predict when the system state approaches catastrophic failure. In this subsection, we present one feature that highlights the evolution of the system state,

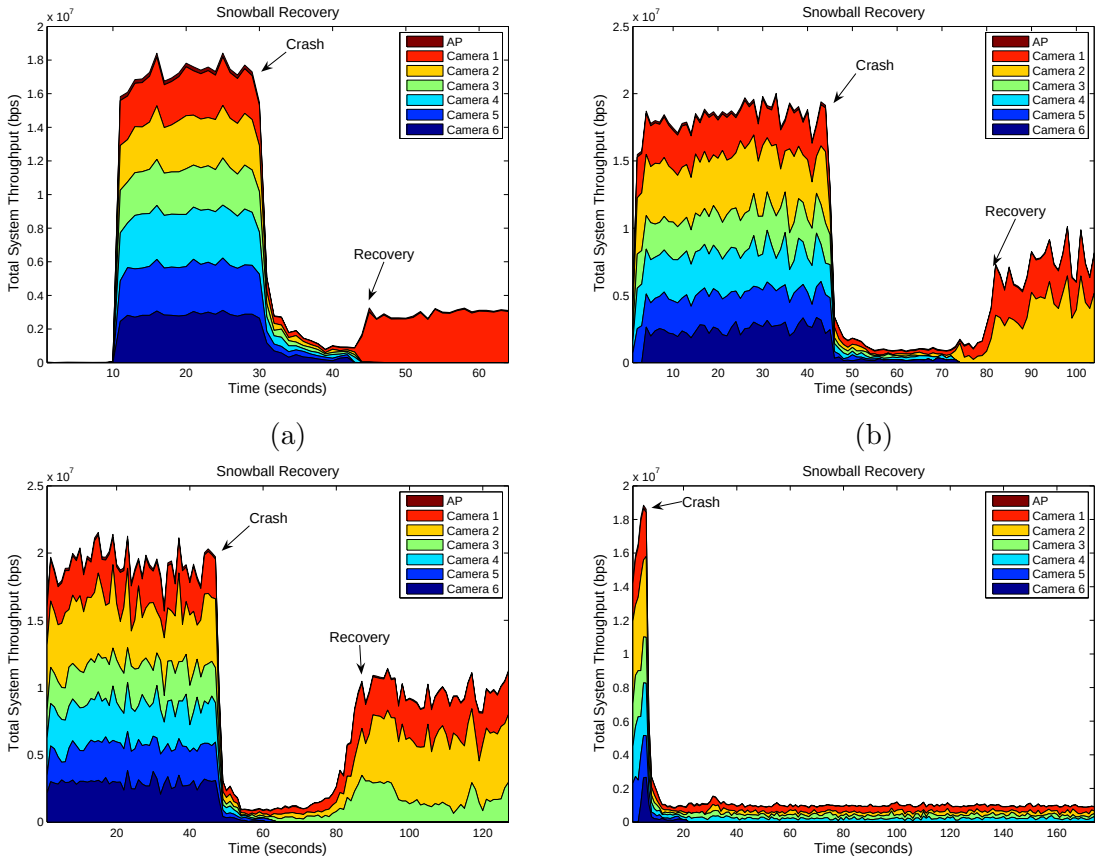


Figure 3.4: Recovery from the Snowball Effect by intentionally stopping a subset cameras. (Recovery time is: (a) 12 sec. when *5 of 6* cameras are stopped after the crash; (b) 24 sec. when *4 of 6* cameras are stopped after the crash; (c) 54 sec. when *3 of 6* cameras are stopped after the crash; (d) System never recovers when *2 of 6* cameras are stopped after the crash.

and which furthermore, could be used to predict the imminent state.

We expect drops in goodput to indicate potential problems. Therefore, we propose the following feature, $0 < gpGap_i(t) = goodput/video\ data\ throughput < 1, \forall i \in N$. This feature is measured each second, and it is an indicator of how efficiently a station is using the medium. *Master gpGap* is defined as the same ratio, but it uses total system goodput and video data throughput instead of each stations' goodput/throughput.

In most experiments, one can visually notice a dip in the *Master gpGap* feature around the time of the crash. For example, Figure 3.5(a) shows such a dip 10 seconds before the network crashes. In some cases, the dip is narrow and deep, but in other cases, it is wide and shallow. We track the correlation between the *gpGap* statistics for three cameras at a time. The origin in this space indicates when *gpGap* is jointly zero for all 3 stations, so normal operation should ideally be as far as possible from the origin. We are interested in the minimum distance point to the origin, representing in 3-space, the lowest point in *gpGap* for all 3 cameras. Similarly, we look at the maximum derivative points of the *gpGap* statistic. We expect there to be two maximum derivative points - one highly negative point (formation of the dip) and one highly positive point (termination of the dip).

We show that *joint gpGap* statistics clearly indicate the evolution of the system state. In Figure 3.5(b) and (c), we present the *joint gpGap* statistics for cameras 2, 3, and 4, and their derivatives. The color bar to the right of each plot indicates the time dimension. In this experiment, we add a new station to the network every 10 seconds, so $0 < t < 10$ corresponds to one active camera, $10 < t < 20$ corresponds to two active cameras, etc.

Figure 3.5(b) shows three distinct clouds of operation: normal, prediction zone (a crash may be imminent), and crashed. The crashed state corresponds to the red points after the sixth camera becomes active, and the Snowball Effect takes its toll. In Figure 3.5, there are a series of points on both the left and right sides of the color bar. On the left side, we have marked points in time that are within $\epsilon = 0.075$ of the minimum distance point. Notice that the black minimum distance points are distributed most densely near the "prediction cloud" region marked on the color bar. On the right-hand

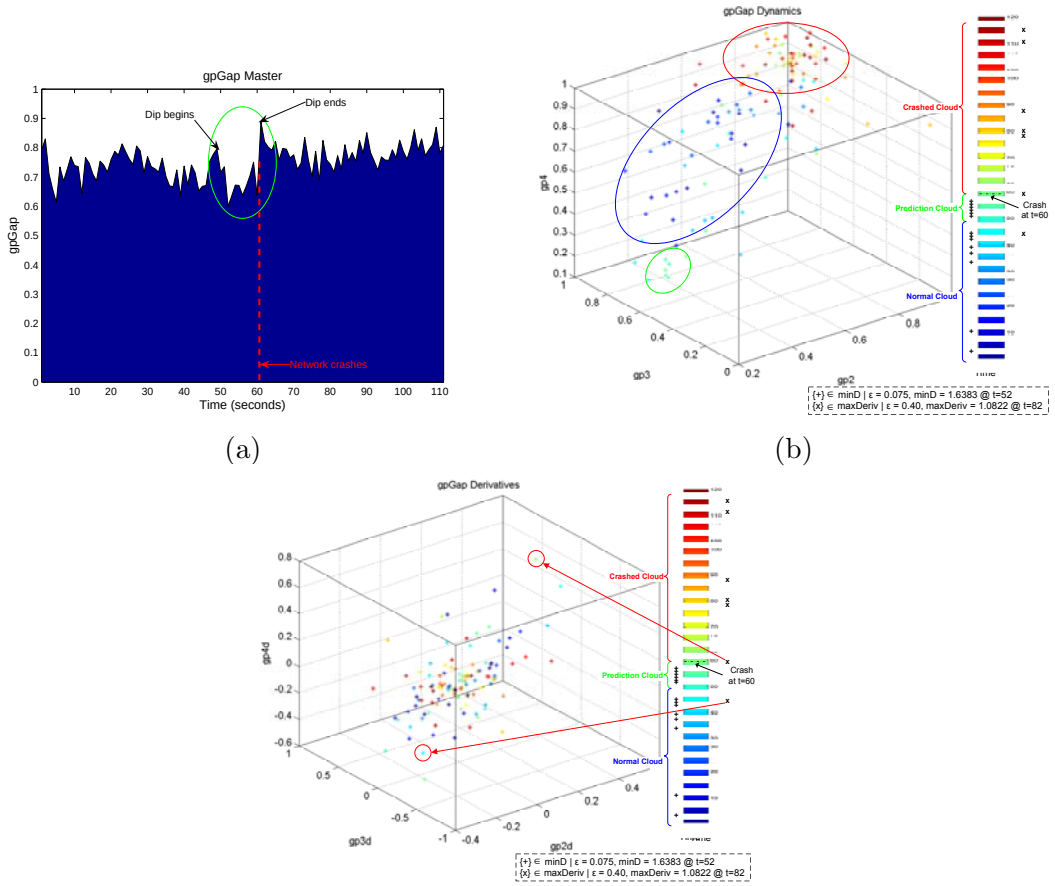


Figure 3.5: The gpGap Statistic (a) may be a useful statistic for predicting catastrophic failure. Its time evolution (b), and derivatives (c) are shown in the plot.

side of the color bar, we have marked the points that are within $\epsilon = 0.40$ of the maximum derivative. One highly negative derivative point appears early in the prediction phase, indicating the formation of a dip. A highly positive maximum derivative point appears at the end of the prediction phase, completing the formation of a dip in the *gpGap* plot. Overall, the transition of the *jointgpGap* statistic in the scatter plot indicates if the system is operating in a healthy regime.

3.5 Related Work

Although the IEEE 802.11 standard allows flexibility regarding the selection of transmission parameters, such as PHY rate, it does not explicitly specify when and how to use specific settings. Early link adaptation techniques like [23] select the corresponding

PHY rate according to a goodput maximization criteria. Alternatively, the Auto Rate Fallback (ARF) algorithm [22], and subsequent approaches [24] adapt rates depending on the frequency of contiguous transmission failures or successes.

Several authors have recently observed flaws in the rate adaptation mechanisms implemented in present hardware ([14, 15]). If a rate adaptation algorithm assumes that failures are due to a bad channel when they are really due to collisions, then it might unnecessarily lower the PHY rate, which causes diminished throughput, among other serious problems (as in ARF, [22]). Since detecting *channel problems* (e.g. path-loss, interference) is difficult in practice, [14] proposes a mechanism to detect *collisions*. In this mechanism, stations exchange transmission time information (piggybacked onto data packets) when transmission failures occur. However, the detection delay is too large to make this solution practical.

Another scheme, [15], uses RTS/CTS messages to differentiate between frame collisions and frame failures due to channel errors. In this paper, the authors recognize that an RTS frame transmission is not at all likely to fail due to a bad channel (because of its small size and robust transmission modulation), and hence use it as an indicator of collision vs. bad channel. Simulations showed throughput improvement, but the technique suffers from overheads, and the algorithm affects the fairness of the DCF mechanism itself.

A third adaptation scheme is SNR-based rate adaptation, where the algorithm estimates the SNR and adjusts the modulation scheme accordingly. Although this approach is impressive in theory, it faces serious practical shortcomings like long estimation delays and difficult exchange of signal information. Delays become particularly critical in the case of real-time voice/video streaming applications, where the link conditions rapidly fluctuate due to mobility or interference. To address this problem, [16] proposes a hybrid SNR-based rate adaptation mechanism. However, this approach is limited by the very accuracy of the estimation possible in a closed-loop, low delay system - especially in the absence of standardized channel condition measurements (e.g. 802.11k).

3.6 Conclusion

The Snowball Effect is the result of an incorrect assumption made in current rate adaptation mechanisms. We showed that current rate adaptation schemes fail because they do not differentiate between poor channel conditions and collisions as the source of transmission failures, and consequently invoke improper responses that cascade to dramatic throughput degradation.

Based on our analysis, we recommend that a next generation rate adaptation algorithm should have the following requirements: (1) It should consider perceptual video quality; (2) It should not react to *all* collisions or *specific* statistics at a fine granularity - this is overly complex and not needed for video streaming; (3) The algorithm should track how close the system is to saturation, and adapt the rate according to the state the system is currently operating at; (4) The algorithm should balance collisions with channel errors.

We propose cross-layer, perceptual-quality-aware link adaptation algorithms in Chapters 5 and 7.

Chapter 4

Airtime Fair Distributed Cross-Layer Congestion Control for Real-Time Video Over WLAN

4.1 Introduction

The proliferation of wireless technology has encouraged a variety of multimedia services to become available on portable devices. For example, IPTV streaming (downlink), real-time gaming, video surveillance (uplink), and conferencing (bi-directional) over the wireless medium have all become alternatives to conventional multimedia content delivery. Unfortunately, the wireless-enabling technology, IEEE 802.11 wireless local area network (WLAN), can be easily demonstrated to fail when subjected to the congested conditions associated with many of these real applications. These effects are a serious hurdle for the deployment and advancement of wireless technologies, and therefore, a congestion-control remedy is needed.

Multimedia has several properties that affect the way we design systems to control congestion and optimize delivery. First, frequent quality changes within a short period can be very annoying to the users, but sparse quality problems can often be concealed by today's video codecs. Therefore, error-free communication over the link is not critical if robust codecs are used. Second, in real deployments, link conditions vary rapidly over time and space, so PHY rates and modulation schemes vary from camera to camera. Furthermore, in multi-camera wireless environments, the risk for congestion increases with each additional client. At the same time, users of multimedia systems expect instant content delivery with good quality.

If we translate the above properties into network parameters, this means that users prefer a system with low latency, low packet loss, high throughput, and robustness to failure. However, jointly addressing these issues is difficult in a layered IP architecture.

Some of the hurdles are:

1. Performance Anomaly: Conventional use of the 802.11 Distributed Coordination Function (DCF) results in a system that achieves throughput fairness, but at the cost of decreased aggregate system throughput and increased congestion [1].

2. Imperfect Link Adaptation (LA): At the MAC level, devices implement LA intelligence [22] to select modulation and channel coding schemes according to the estimated channel conditions. However, today's LA algorithms can be misled by collision errors - thus lowering the PHY rate even though the SNR is good, which in turn leads to an even higher collision probability. This "Snowball Effect" [25] quickly leads the system into catastrophic failure.

3. Available Bandwidth Estimation: It is often desirable for congestion control algorithms to estimate the available bandwidth. At the application (APP) layer, systems suffer from slow and inaccurate capabilities to estimate end-to-end available bandwidth. A measure of the available bandwidth can not be very accurate in the absence of MAC layer information.

Note that the plan of adopting hybrid coordinator function (HCF) controlled channel access (HCCA) has been put on hold by WiFi Alliance due to lack of commercial interests [26]. Therefore CSMA/CA based channel access will still play the major role of mainstream products in the near future. It is our belief that improving quality of service (QoS) under contention based channel access is a valuable and interesting topic.

Given these difficulties of a single-layer solution, several cross-layer adaptation systems for video transmission have been proposed. In [27], the authors provide adaptive QoS by utilizing real-time MAC parameter adaptation on the retry limit plus priority queuing. In [28], background traffic is also considered in the adaptation process. The cross-layer signaling scheme is used to pass link quality information to the video coder based on signal strength and estimated throughput, but the performance anomaly is still an issue. Moreover, Khan et al. [29] use PSNR as the quality metric through pre-estimated distortion, but this is not suitable for real-time video applications. Setton et al. [30] estimate link capacity then jointly allocate it with video flows to maximize

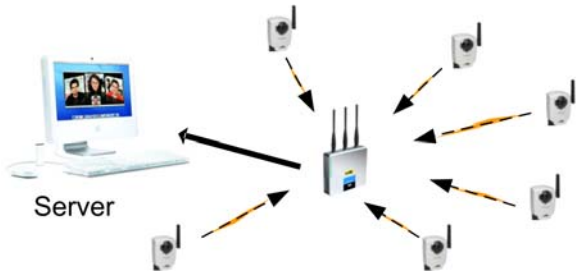


Figure 4.1: Wireless video transmission topology. Videos are streaming from wireless cameras to the server through an access point.

link utilization. This work further shows the feasibility of active interaction across all layers in the protocol stack through information exchanging. In our previous work [31], a practical cross layer framework utilizing airtime fairness is shown to be effective on certain controlled situations. We believe in the potential of this track, but much work needs to be done considering aforementioned highly dynamic problems.

We propose a solution across layers that adaptively ensures wireless video transmission of acceptable quality in an uplink, multi-camera topology as shown in Fig. 4.1 where videos are streaming from wireless cameras to a server through an access point. Suitable applications include wireless home entertainment, video surveillance, and real-time monitoring for search/rescue. Our central idea is that airtime fairness is preferred to throughput fairness, and such a policy should be achieved using an interplay of control loops across layers. We propose a fast frame-by-frame control loop in the MAC layer while simultaneously exploiting the powerful control loop gain attainable by performing source-rate adaptation in the APP layer at a slower timescale. Furthermore, we show that the use of airtime fairness combined with video source rate adaptation increases aggregate throughput, decreases the packet loss rate (PLR), and improves robustness. All of these features are implemented according to a distributed architecture. Thus, our solution manifests as a module (which runs within each camera) that has two tasks: (1) induce airtime fairness, and (2) perform video source rate adaptation. Implementing these tasks involves:

1. Scaling the video encoder's target bit rate according to the camera's observed packet loss rate. We call this "SRA" (source rate adaptation).

2. Upper-bounding the video encoder's target bit rate according to a formula we call "FATE" (fair airtime throughput estimation), which computes the maximum throughput that the camera is allowed to use in order to guarantee system-wide airtime fairness.

3. Adapting the 802.11 MAC parameters, in particular the contention window, to control access to the channel in a way that guarantees system-wide airtime fairness at a quick timescale. We call this "CWA" (contention window adaptation).

Overall, an implementation of the above concepts represents a distributed cross-layer architecture. We detail how to exploit and harmonize these properties in order to create a distributed congestion control system that has robustness on a frame-by-frame scale. Our analysis and validation of the algorithm is based on both simulation and experimental implementation of the algorithm in programmable wireless cameras. Results from real experiments match those from simulation, and they demonstrate significantly improved performance in video over wireless local area network (WLAN) in terms of both throughput and packet loss. When applying more practical and severe conditions such as suboptimal link adaptation, or dynamically adding cameras, the control scheme can sustain a very good PSNR.

The rest of this chapter is organized as follows: In Section 4.2 we give a theoretical treatment of airtime fairness and control, and ultimately arrive at a conceptual description of our solution. After that, we discuss the implementation of our solution in Section 4.3. A performance analysis follows in Section 4.4. Finally, we conclude the paper in Section 4.5.

4.2 Theory of Airtime Fairness and Control

Systematic approaches to cross-layer design of joint congestion control and scheduling has been studied in recent publications [32]. Researchers extend the utility maximization frame work for a general cross-layer design methodology that duality theory leads natural layer decomposition into separate designs of different layers interacting with each other [33]. Related works include [34][35] on joint source traffic and MAC

scheduling. Based on the layering concept, proposed heuristic framework fills up the gap between system theory and real-world implementation. Specifically, the airtime fairness-centric design highlights contention nature of CSMA/CA as well as video application properties. We explain the critical interplay next.

4.2.1 Why We Need Airtime Fairness

It is well known that per-station throughput in a WLAN multirate basic service set (BSS) tends to move quickly toward the rate of the lowest rate station in the BSS [1]. Channel access among competing stations is managed such that stations divide the available throughput according to CSMA/CA, which implements a throughput fairness policy. We illustrate this in Fig. 4.2, which shows the effect of 1 out of 10 stations in a BSS dropping its PHY rate from 11 to 2 Mb/s. Throughput fairness results in decreased aggregate system throughput (from about 5.5 to 4 Mbps in Fig. 4.2), no throughput guarantees above a minimum level, and a possibility of congestion with the added danger of completely dropping some video flows. Consequently, if a station switches its PHY rate, then we want to adapt that station's channel share and video source rate accordingly in order to protect the system from catastrophic behavior. We need a balanced allocation of airtime such that a slow station does not consume more airtime than a fast station. We refer to this principle as airtime fairness.

The fairness issue in IEEE 802.11 WLAN has received considerable attention in the recent literature. Self-clocked fair Queuing (SCFQ) [36] has been adopted by the Distributed Fair Scheduling (DFS) protocol [37] for weighted fairness. They apply the system virtual clock in a fully distributed fashion so as to emulate centralized fairness. Other algorithms [38, 39, 40] model the Distributed Coordination Function (DCF) or the Enhanced Distributed Channel Access (EDCA) based on Markovian structure [19] and then suggest tuning CSMA/CA parameters such as contention window size, frame size, retry limit, and AIFS to achieve desired fairness related to airtime. Proportional fairness has been recommended as the optimized objective of resource allocation in multi-rate wireless networks [41]. If multiple stations contend for a shared resource, the system shall guarantee individual throughput proportional to its capacity. Jiang et al.

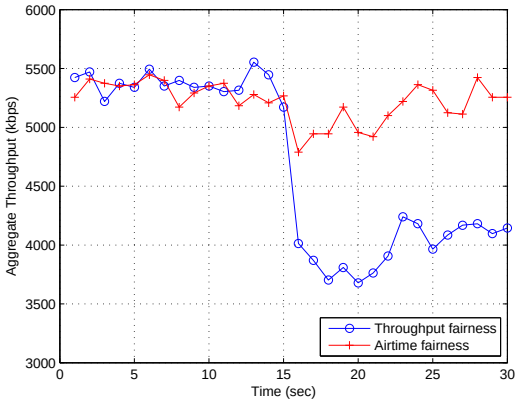


Figure 4.2: Throughput vs. Airtime Fairness. The blue line shows the aggregate system throughput under a throughput fairness regime, and the orange line is aggregate system throughput under an airtime fairness regime. Initially, all 10 stations are at 11 Mb/s. At t=15, 1 of 10 stations switches from 11 to 2 Mb/s. The aggregate system throughput is reduced in the case of throughput fairness.

[42] have further demonstrated the equivalence of proportional and airtime fairness for WLAN under which every station consumes same amount of airtime. It implies that the airtime fairness is a natural result of the more fundamental proportional fairness. Accordingly, multi-rate stations can contend in a fair manner by incorporating control techniques with proper objectives.

We consider two ways to control access to the medium, and therefore allocation of airtime:

1. MAC Layer Approach: Adapt the contention window. If two stations are contending for the channel, the one with a smaller initial contention window size (CW_{min}) is the one that is more likely to win the contention.

2. APP Layer Approach: Adapt the video bit rate. A station that transmits video encoded at a 400kb/s bit rate will not contend as aggressively for the channel as a station that transmits video encoded at a 800kb/s bit rate.

Although we can control airtime from either the APP or the MAC, we have found that the best solution is to use both methods simultaneously. The two individual methods each have drawbacks; however, controlling airtime simultaneously from the MAC and the APP enables a joint solution.

4.2.2 MAC-Layer Airtime Control

We now introduce the concept of MAC-layer airtime control and its advantages and disadvantages. It has been shown that the initial contention window size CW_{min} is an effective parameter for controlling airtime share over time [39]. The intuition for contention window adaptation (CWA) is as follows.

Assume a set of stations $S = \{1, \cdots, N\}$. In a time interval, an arbitrary station, i, under static channel conditions consumes data airtime A_i in n_i transmissions given by:

$$A_i = \frac{1}{r_i} \cdot \sum_{k=1}^{n_i} f_k = n_i \cdot \frac{\bar{f}}{r_i} \tag{4.1}$$

where f_k is the MAC service data unit size (MSDU) size for the kth frame transmission, $\bar{f}$ is the average size, and r_i is the PHY rate applied.

From the CSMA/CA process, CW_i defines the allowable range of contention windows size, i.e., the backoff time is decided by the random number uniformly drawn within the interval of $[0, CW_i - 1]$. Therefore, the expected value of backoff time, B, can be related to adapted CW^i_{min} as

$$E\left[\sum_{k=1}^{n_i} B_k\right] = n_i \cdot E[B] \approx n_i \cdot \frac{CW^i_{min}}{2} \cdot SlotTime \tag{4.2}$$

assuming collisions and wireless errors are both low due to proper CWA and link adaptation. Since all stations perform backoff simultaneously, we can approximate that, for stations i and j, $\sum_{k=1}^{n_i} B^i_k \approx \sum_{k=1}^{n_j} B^j_k$, if and only if

$$\frac{n_i \cdot CW^i_{min}}{n_j \cdot CW^j_{min}} = 1. \tag{4.3}$$

Airtime fairness is achieved if $A_i = A_j$ [c.f. (4.1)]. If this equality is to hold, then, since $\bar{f}$ is constant, we have airtime fairness if $n_i/n_j = r_i/r_j$. Plugging this back into (4.3), we have airtime fairness criteria in a multirate environment as in [43, 44] if

$$r_i \cdot CW^i_{min} = r_j \cdot CW^j_{min}. \tag{4.4}$$

Thus, by tuning the CW_{min} ratio, relative fair airtime share can be realized.

We verified the CWA control scheme using NS2 configured with saturated traffic and six stations operating at three different PHY rates. Table 4.1 shows the fairness

Table 4.1: Airtime fairness under approximated control equation and saturated traffic. High PHY rate stations gain 5 to 6 % more airtime share than the ideal share at 0.1667.

Camera #	1	2	3	4	5	6
PHY Rate (Mpbs)	11	11	5.5	5.5	2	2
PHY Rate Ratio	5.5	5.5	2.75	2.75	1	1
CW_{min}	32	32	64	64	176	176
Airtime Utility	0.17316	0.177852	0.162435	0.171038	0.152995	0.162519
Throughput (kbps)	1234.213	1267.657	578.8859	609.5421	198.2704	210.6125
Throughput Ratio	5.860113	6.018904	2.748582	2.89414	0.941399	1

results among stations when we set CW_{min} according to (4.4). There are ratios of two measurements, PHY rate and throughput, in the table. They indicate the proportion of each measurement on corresponding station to station at the lowest 2 Mb/s PHY rate (station 6) from the test. Airtime utility is evaluated as the share of data transmission time for each station. If we take the average airtime utility of two high rate stations and compare to the ideal share at 0.1667, we obtain 5.3% of error margin. Consequently, by tuning the CW_{min} ratio, relative fair airtime share can be realized.

Note that MAC-layer adaptation has the advantage of fast frame-by-frame timescale reaction. Changing CW_{min} instantaneously modifies the station's channel access properties. However, such a change may also lead to high packet loss at the interface queue unless additional action is taken. We now illustrate this downside with an example. Consider two stations: Camera A and Camera B, both transmitting MPEG4/RTP video encoded at a bit rate of 800 kb/s. Camera A uses an 11 Mb/s PHY rate and Camera B uses a 2 Mb/s PHY rate. Under airtime fairness, the CWA policy penalizes (lowers) Camera B's channel access probability by a factor proportional to its reduced PHY rate. This reduction in access probability combined with the lower PHY rate puts a theoretical limit on the maximum throughput that Camera B can attain. If this limit is less than 800 kb/s (approximately the minimum throughput needed to sustain a video encoded at 800 kb/s), then the transmitter's queue will saturate and begin dropping packets. Consequently, the received video quality will drop due to increased packet loss.

4.2.3 APP-Layer Airtime Control

The upper bound on throughput imposed by CWA is precisely what guarantees system-wide airtime fairness. Is there another way to impose such an upper-bound without using CWA? Suppose we design an algorithm that somehow "matches" the encoder's target bit rate with the throughput constraint that's implicitly imposed by the CWA algorithm. For example, in the previous case, suppose the upper bound was 500 kb/s. If we deactivate the CWA policy and simply tell Camera B's video encoder to lower its target bit rate from 800 to 500 kb/s, then we will have implicitly induced airtime fairness without using CWA.

Since we're not actually using CWA in this APP-Layer approach, we need a formula that predicts the upper bound on throughput that CWA would have imposed given a set of current network conditions. In our solution, the module responsible for this task is called cross-layer fair airtime throughput estimation (FATE).

To implement the concept of airtime fairness, we apply a technique whereby SRA cooperates with FATE. The intuition of SRA is that reducing the target bit rate of the video reduces packet loss. On the other hand, when packet loss is low, SRA can increase the target bit rate. When we do this target bitrate adaptation within the upper bound set by FATE, we call it "Smart SRA". When we ignore FATE, it is possible to set the target bit rate higher than the theoretical maximum allowed for airtime fairness, and we call this "Blind SRA." In reality, we always use Smart SRA; Blind SRA is only used for comparison.

The advantage of controlling airtime using source bit rate adaptation is that it solves the problem of high packet loss that plagues the CWA solution. In the previous example, CWA put an upper bound on Camera B's throughput (say, 500 kb/s), but the camera still tried transmitting video encoded at a 800 kb/s target bit rate, resulting in high packet loss. However, when we use the APP-layer approach, we physically change the encoder's target bit rate from 800 to 500 kb/s, thus greatly reducing packet loss compared to CWA. However, it operates on a slow timescale since video encoders take several seconds to fully reach a newly set target bit rate. Furthermore, as we will soon

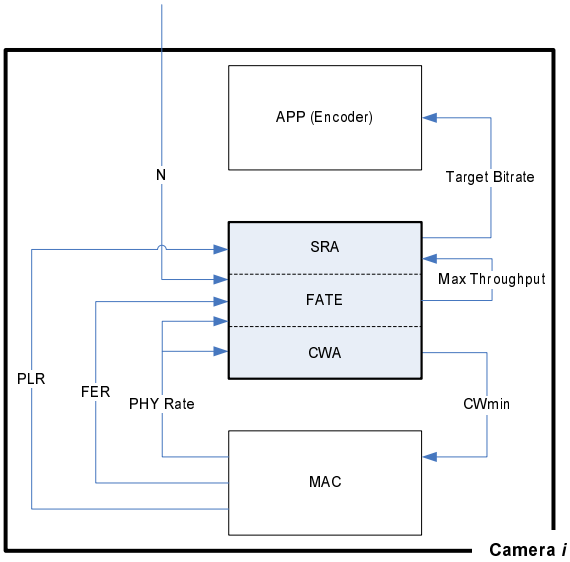


Figure 4.3: CLC System Diagram. The SRA module constantly interacts with the MAC to track packet loss for source rate. The FATE module takes frame error rate and PHY rate for throughput. CWA is used to instantly provide airtime fairness. N is the number of active cameras.

see, FATE reacts on fixed 100 ms intervals rather than on fast frame-by-frame time scale like CWA.

After exploring CWA, FATE, and SRA, it is now conceptually clear that simultaneously doing APP and MAC layer airtime control yields the best combination of low packet loss (due to APP) and quick reactivity/stability (due to MAC).

4.3 Implementing Distributed Cross-Layer Congestion Control

In the last section, we introduced the theory of cross-layer airtime control. In this section, we explain how the solution is actually implemented in a distributed manner.

The cross-layer congestion control (CLC) system is shown in Figure 4.3 for a single camera. Since the system is distributed, each camera has its own instantiation. All instantiations share the number N of active cameras in the BSS. As we will see, N can be changed dynamically.

The CLC module operates as follows on each camera: The SRA module constantly interacts with the MAC to track packet loss. If the PLR is too high, then it incrementally reduces the target bitrate. If the PLR stabilizes at a low value, then SRA increases

the target bitrate stepwise as long as the new target bitrate does not exceed the maximum set by the FATE module. The FATE module takes frame error rate (FER) as well as PHY rate information from the MAC, and N from the application server to compute the maximum throughput that a station can attain to maintain system-wide airtime fairness. During every adjustment to the target bitrate, CWA is used to instantly provide airtime fairness while the relatively slow encoder "catches up" to the new target bitrate. The specific way that each component uses its inputs and calculates its outputs is detailed in the next three subsections.

4.3.1 Cross-Layer Fair Airtime Throughput Estimation

There are several ways to evaluate per-station and system throughput in IEEE 802.11 networks. Based on framing and timing details in standards, theoretical maximum throughput can be calculated assuming no channel or collision errors [45]. With errors, the Markovian model can be applied again to model the throughput in a multirate deployment, like the one we are interested in [39]. Given information such as wireless error rate, transmission rate, and backoff parameters for all stations, one can solve nonlinear equations for transmission probabilities in order to compute estimates of system and per-station throughput.

However, in a real multi-camera setting, we do not have the distributed access of all this information, nor do we have the computation power for accurate modeling and equation solving. Therefore, we propose a cost efficient scheme for per-station throughput evaluation. Any device that is capable of counting its frame errors and has information about the total number of stations in its BSS can implement the proposed formula.

Assume G^i_{max} is the theoretical maximum throughput as seen by station i, computed according to [45] given $\bar{f}$ (average MSDU size) and r_i (STA i's PHY rate):

$$\begin{aligned} G^i_{max} &= \frac{MSDU\ Size}{Delay\ per\ MSDU} \\ &= \frac{\bar{f}}{T_{DIFS} + T_{SIFS} + T_{BO} + T_{ACK} + T_{DATA}} \end{aligned} \tag{4.5}$$

with delay components explained in Table 4.2.

Table 4.2: MAC Delay Components

Parameter	**Comments**
T_{SIFS}	Single Inter-Frame Spacing time
T_{DIFS}	$T_{SIFS} + 2 \cdot \sigma$
T_{BO}	Backoff time
T_{ACK}	Ack frame time
T_{PHY}	Preamble + PLCP time
h_{MAC}	MAC header size
T_{DATA}	$T_{PHY} + \left(h_{MAC} + \bar{f}\right)/r_i$

We now exploit the throughput version of WLAN airtime (proportional) fairness. Define G_o^i as actual throughput of station i in an error free channel. The portion for airtime consumed can be represented as (G_o^i/G_{MAX}^i). Then apply airtime fair criteria:

$$\begin{gathered} \sum_{i=1}^{N} \frac{G_o^i}{G_{MAX}^i} = 1 \\ \frac{G_o^i}{G_{MAX}^i} = \frac{G_o^j}{G_{MAX}^j}, \ \forall i, j \end{gathered} \tag{4.6}$$

we have

$$G_o^i = \frac{1}{N} \cdot G_{MAX}^i \tag{4.7}$$

when airtime fairness is achieved. An interesting property that, given a fixed number of stations, the throughput of one station is independent of the data rates used by other stations [42] is confirmed. The next step is to compensate for the effect of errors due to the FER at each station, p_f^i. The idea is to estimate the portion of frames successfully transmitted in a given amount of airtime share. Since collision rate is low, so we discount the throughput estimate by p_f^i:

$$G_e^i \approx G_o^i \cdot \left(1 - p_f^i\right) = \frac{1}{N} \cdot G_{MAX}^i \cdot (1 - p_f^i) \tag{4.8}$$

The necessary observations, $\bar{f}$, r_i, and p_f^i, are evaluated on a fixed interval (i.e. every 100 ms) and used to re-compute G_e^i. The result is then passed to the SRA module, as indicated by the arrow labeled "Max Throughput" in Figure 4.3 and the condition on G_e shown in the SRA pseudo code (see Fig. 4.4). SRA module updates video encoder with target bitrate every ϵ time. The rate goes down if PLR is larger than threshold η_1; goes up if lower than η_2 for t consecutive time. Average estimated throughput, $\overline{G_e^i}$, is caluated from G_e^i samples over ϵ time and applied after adjusted by tolerable error

```
1: repeat
2:    update PLR from MAC
3:    update G_e^i from FATE module samples
4:    if PLR > η1 and Bitrate−δ ≥ V_min then
5:       Bitrate ← Bitrate−δ
6:    else if PLR < η2 for t consecutive time and
      Bitrate+δ ≤ min{ G_e^i · 1/(1−η1), V_max } then
7:       Bitrate ← Bitrate+δ
8:    else
9:       do nothing
10:   end if
11:   send Bitrate to video encoder
12:   sleep ε time
13: until end of video session
```

Figure 4.4: Pseudo code of Source Rate Adaptation (SRA) module. SRA updates video encoder with (target) Bitrate every ϵ time. PLR thresholds η_1 and η_2 used to trigger rate changes. FATE influences the system via SRA as shown by the condition on G_e.

η_1. Encoder capability in terms of maximum rate V_{max}, minimum rate V_{min}, and step size δ are also considered.

Consequently, FATE enables airtime fairness via the SRA process and is a sufficient solution under a static channel (without PHY rate changes). As mentioned earlier, the 5-10 s delay required by the encoder to "catch up" to the new target bitrate is too long. If there is a change in the PHY rate, the system could easily suffer catastrophic failure during the transition period. In the next subsection, we show how CWA is used to provide stability during these transition periods.

4.3.2 Contention Window Adaptation

The CWA module computes the appropriate CW_{min} for each data frame immediately before the frame enters the backoff process. The computation uses the PHY rate as an input, and computes its output by using an abstraction of Equation 4.4. Namely, let the right hand side of (4.4) be equal to a constant, $r_{max} \cdot CW^o_{min}$, where r_{max} and CW^o_{min} are the maximum PHY rate and initial CW_{min} value, respectively. Then, the left hand side is divided by r_i to get target CW_{min} of the i^{th} camera. Thus, we have

$$CW^i_{min} = \min\left\{\lfloor c_i \cdot CW^o_{min} \rfloor, CW_{max}\right\} \tag{4.9}$$

where $c_i = r_{max}/r_i$.

CW_{min} is set inversely proportional to the PHY rate. Thus, stations using low channel rates due to degraded signal quality have a lower probability to transmit than high-rate stations. Statistically, equal data airtime share among stations is maintained under this adaptation. This implementation guarantees QoS according to a set of requirements and also releases congestion on channel utilization time, and thus it addresses the hurdles introduced in Section 4.1.

In the next section, we use both simulations and real experiments to confirm that the cross-layer control concept presented above is reasonably accurate for video applications.

4.4 Performance Evaluation

Next we evaluate both in simulation and experimentally a family of congestion control algorithms based on their improvement in PLR, throughput, and reliability, and ultimately show that CLC exhibits the best performance.

4.4.1 Simulation and Experimental Setups

Our real-world experimental setup is based on a wireless, real-time, surveillance/security system. We consider the following scenario: Six to seven Axis 207w cameras stream video (100kbps-800kbps Constant Bitrate MPEG4/RTP) wirelessly (802.11b) to the central security server (a Dell OptiPlex GX280) via a wireless access point (Siemens AP2630). The GX280 processes the video in real-time to perform actions such as: face detection, person tracking, forbidden zone detection, etc. We monitored system statistics in real-time using custom plugins for AiroPeek SE [17]. The distributed cross-layer control algorithms were implemented on the cameras using shell scripts written for the Linux kernel running on each camera.

Test scenarios are described by the number of low rate stations in the BSS (see Table 4.3). The various algorithms we implemented are shown in Table 4.4. We set the number of cameras to $N \in \{6, 7\}$, so there are a total of $2 \times 3 = 6$ multirate

Table 4.3: Multirate scenarios (Mbps). The scenarios are described by the number of low rate stations in the BSS. There are three scenarios each for $N = 6$ and $N = 7$, giving a total of 6 scenarios.

Camera #	1	2	3	4	5	6	(7)
All High,N=6,(7)	11	11	11	11	11	11	(11)
One Low,N=6,(7)	11	11	11	11	11	2	(11)
Three Rates N=6,(7)	11	11	5.5	5.5	2	2	(11)

Table 4.4: Solutions under test. Cameras are programmed corresponding to the following cases.

Name	**Description**
Raw CBR	Constant bitrate that never changes (off-the-shelf)
Blind SRA	Source Rate Adaptation (SRA) without input from FATE (i.e. no limit on how high station can make its bitrate)
Smart SRA	Source Rate Adaptation (SRA) plus FATE
CLC	SRA plus FATE plus Contention Window Adaptation (CWA)

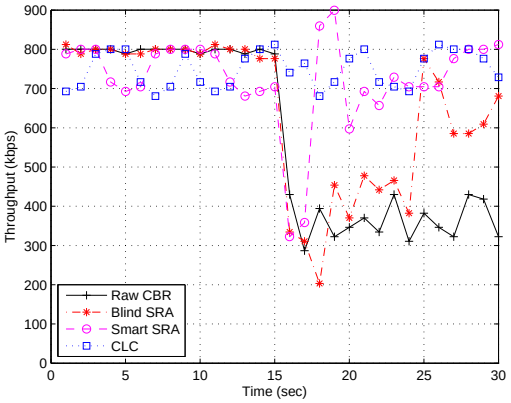
scenarios. The values $N \in \{6, 7\}$ were chosen empirically so that N=6 brings the off-the-shelf system close to saturation and N=7 brings the system just over saturation. Observations also show that: 0 (*ideal channel*) $< p_e < 0.2$.

The experimental procedure was as follows: First, we configured each camera to use MPEG4 as shown in Table 4.5 with initial constant target bitrate 800kb/s. Then, we programmed each camera with one of the four algorithms listed in Table 4.4, and we selected one of the six multirate scenarios listed in Table 4.3, and ran it for 30 seconds while the cameras streamed video to the server. Additional experiments were conducted where the first 15 seconds used one multirate scenario, and the last 15 s used a different multirate scenario (to study PHY rate transitions in real-time).

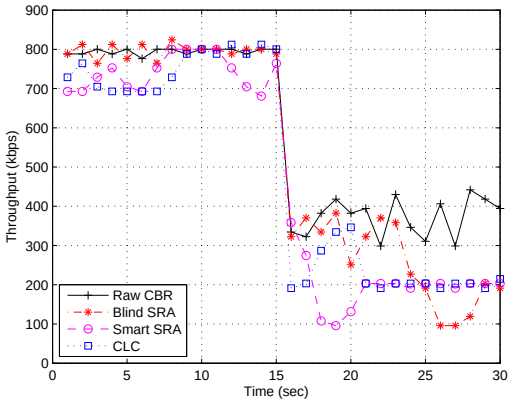
We also utilized ns2 simulations, which were designed to match the real scenarios just described. Our theories were first validated with ns2 tests, and then we used the real experiments to test feasibility of implementation and performance under real conditions.

Table 4.5: Network Configuration

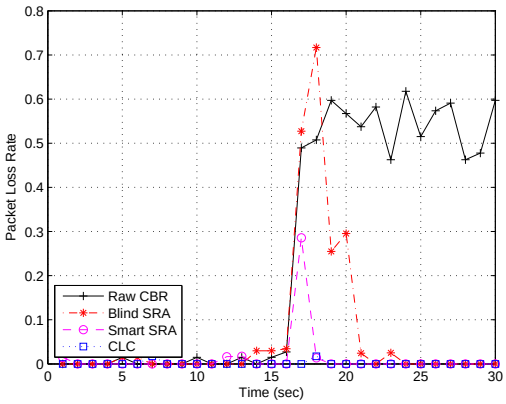
Layer	**Parameter**	**Value**
Application	Codec	MPEG4, CBR 100kbps-800kbps, 100kbps per step, GOV=30, Resolution=640x480,
	Traffic Direction	One-way (Uplink video)
	Total Packet Size (bytes)	1500
Transport	Protocol	IP/UDP/RTP
	Header (bytes)	20+8+12
Link/MAC	Protocol	802.11e
	Retry Limit	3
	Queue	Type: pfifo_fast Length=1000 packets
	Fragmentation	Disabled
	RTS/CTS	Disabled
Physical	Protocol	802.11b
	Header (bytes)	8 byte preamble
	Max Rate (bps)	11 Mbps
	Base Rate (bps)	1 Mbps



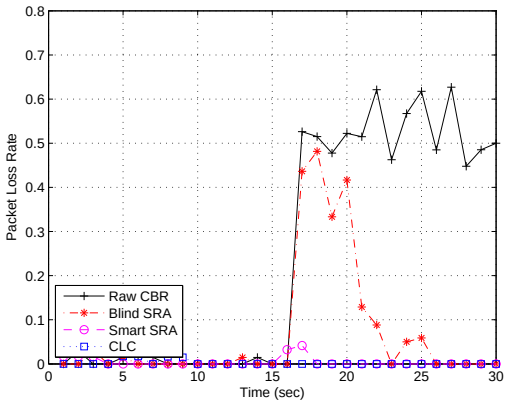
(a) 11 Mbps station throughput



(b) 2 Mbps station throughput



(c) 11 Mbps station PLR



(d) 2 Mbps station PLR

Figure 4.5: Simulation results for throughput and PLR. The scenario is $N = 6$, All High for the first 15 seconds and then transition to Three Rates for the last 15 seconds. The throughput and PLR are given for all four different control algorithms for an 11Mbps station and a 2Mbps station.

4.4.2 Basic Comparison of Solutions Under Test

In this subsection, we overview the performance of all four control algorithms listed in Table 4.4 vis-a-vis the following experiment: At t=0, N=6 stations are at the "All High" scenario, then at t=15 sec., there is a transition to the "Three Rates" scenario. Typical values of the SRA parameters are $\eta_1 = 5\%$, $\eta_2 = 3\%$, δ =100kb/s, t =3 sec., $\epsilon = 1$ sec. For clarity, we initially show only simulation results, and further, we only report results from a randomly selected fast (11 Mbps) station and a randomly selected slow (2 Mbps) station.

Fig. 4.5 summarizes the results of our comparison in terms of throughput and packet

loss at the two selected stations. As expected, the CBR approach performs the worst since it is simply the out-of-the-box approach to uplink video without any adaptation. It yields the highest PLR and lowest throughput in both the 11 and 2 Mb/s stations.

The Blind SRA algorithm shows mild improvement in terms of PLR in both stations. Notice the sharp decrease in throughput at the high rate stations [Fig. 4.5(a)] immediately after the transition to "Three Rates". This is because Blind SRA operates using throughput fairness, so the presence of a low-rate station causes the throughput of every station in the BSS to converge toward that of the lowest rate station. If this throughput is less than the minimum needed to sustain the video (800 kb/s), then the station will suffer high packet loss. Indeed, in this experiment the throughput converges to about 350 kb/s (c.f. Table 4.6), and consequently, we see a corresponding jump in PLR in both 11 Mb/s [Fig. 4.5(c)] and 2 Mb/s [Fig. 4.5(d)] stations. The spike will go away as the SRA algorithm reacts to the high packet loss by reducing the target bitrate of the video. However, as PLR transiently improves, Blind SRA will increase the target bitrate without regard for the current PHY rate or other stations in the BSS. Eventually, the algorithm will increase the target bitrate beyond what the channel can accommodate (possibly even all the way back to the maximum target of 800 kb/s), and thus, the undesirable characteristic of Blind SRA is oscillations in PLR and Throughput.

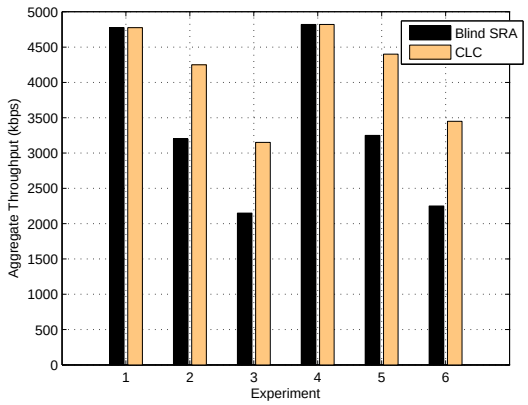
The Smart SRA algorithm demonstrates sound improvement in both PLR and throughput. It enhances Blind SRA by upper-bounding the throughput in SRA via FATE, thus eliminating oscillations and promoting airtime fairness. The drawback of Smart SRA is that it operates at a slower time scale, so it is unstable during PHY rate transitions. Indeed, in Fig. 4.5(a), we see a temporary drop in throughput (and a corresponding spike in PLR [Fig. 4.5(c)] immediately after the transition to the "Three Rates" scenario.

The CLC algorithm (SRA + FATE + CWA), demonstrates the best overall performance. It inherits all of the benefits of Smart SRA, plus it eliminates the instability during transition periods by doing CW adaptation (which has a quick response).

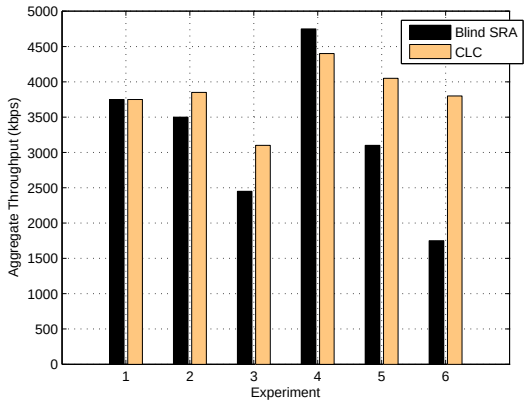
We now detail the throughput performance of CLC versus Blind SRA. Table 4.6

Table 4.6: Simulation Average Throughput (kbps): $N = 6$, $p_e = 0.2$, Initial Video Target Bitrate = 800kbps

Camera #	1	2	3	4	5	6	Total
One Low, Blind SRA	496	559	324	571	561	592	3105
One Low, CLC	726	730	740	737	747	189	3871
Three Rates, Blind SRA	441	514	275	377	416	205	2231
Three Rates, CLC	745	747	498	488	178	186	2844

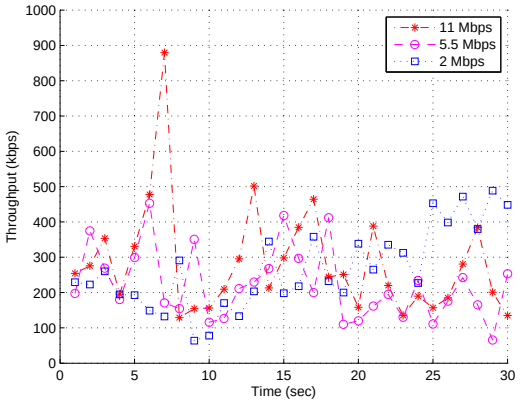


(a) Simulation

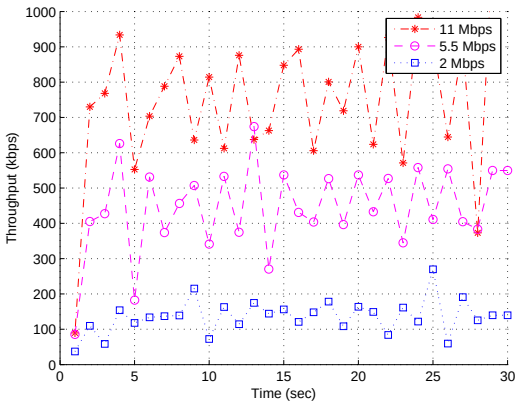


(b) Real

Figure 4.6: Aggregate system throughput for Blind SRA vs. CLC. Experiments: 1) $N = 6$ All High, 2) $N = 6$ One Low, 3) $N = 6$ Three Rates, 4) $N = 7$ All High, 5) $N = 7$ One Low, 6) $N = 7$ Three Rates

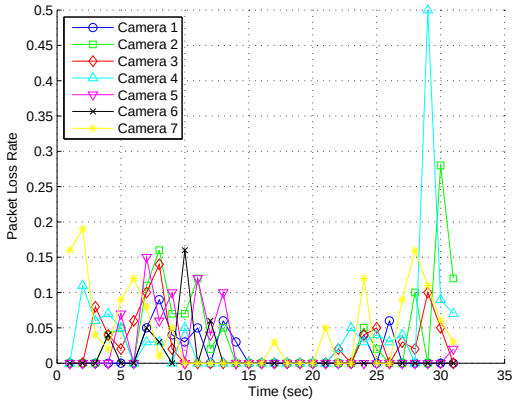


(a) Blind SRA Throughput

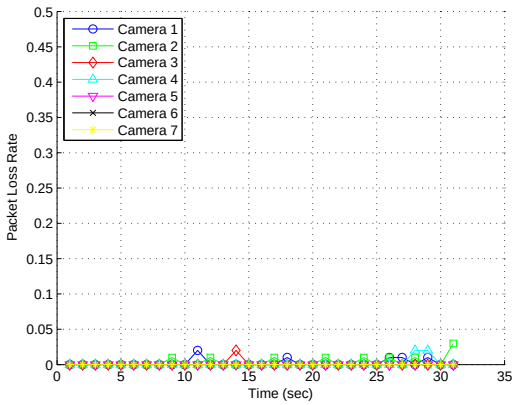


(b) CLC Throughput

Figure 4.7: Representative throughput for Blind SRA vs. CLC (Data is from real experiments). The scenario is $N = 7$, Three Rates. The use of CLC induces airtime fairness. This is the reason that the per-station throughput is layered proportional to the station's PHY rate. As evidenced in the aggregate throughput plots, this layering ultimately results in increased aggregate throughput.



(a) Blind SRA Packet Loss



(b) CLC Packet Loss

Figure 4.8: Packet Loss Rate for Blind SRA vs. CLC (Data is from real experiments). The scenario is $N = 7$, Three Rates. The use of SRA ensures that every station sets its target bitrate in a way that minimizes packet loss, and the use of FATE in conjunction with SRA ensures that airtime fairness is maintained. Under this regime, packet loss is drastically reduced.

compares the average throughput of Blind SRA and CLC for three scenarios in simulation. We see that CLC yields a higher aggregate and per-station throughput than Blind SRA because it uses FATE plus CWA to ensure airtime fairness, and hence the presence of low-rate stations does not penalize the performance of high-rate stations.

To validate our claims and results, we use data from real experiments. Fig. 4.6 compares the aggregate throughput for various scenarios under Blind SRA and CLC for both simulation and real experiments. The figure shows a reasonable correspondence between results obtained from simulation and from real experiments. Additionally, as the channel moves closer to its capacity limits (increasing N), and as the variation in PHY rates among the stations increases, the relative performance of CLC over Blind SRA grows. In fact, for the most challenging scenario ($N = 7$, Three Rates), the aggregate throughput observed in the real experiment is nearly doubled (from 1.75 to 3.35 Mb/s), thus expanding the effective capacity of the system for higher bitrates and/or more cameras. This is clear evidence that airtime fairness is preferred to throughput fairness in multirate wireless video transport systems.

To rectify the exact reason why airtime fairness results in more efficient use of the bandwidth than throughput fairness, we examine the per-station throughput from a real experiment during the "Three Rates" scenario using Blind SRA and CLC for N=7. For clarity, we report results from a randomly selected ("representative") station at each PHY rate. The layered throughput seen in Fig. 4.7(b), is due to CLC's airtime fairness policy, which forces per-station throughput to be proportional to the station's maximum attainable throughput as determined by FATE. The effect of this layering is that slow stations never try to use more of the channel than they can actually handle given their PHY rate, thus providing more airtime to fast stations, and ultimately increasing the aggregate throughput.

Packet loss rate (PLR) is another critical factor of video quality. It is worthwhile looking at its raw value across time and space. We have selected the experiment that we used in the throughput study: $N = 7$, "Three Rates". The results are shown in Fig. 4.8.

Indeed, we see that PLR is drastically reduced while collision probability (available

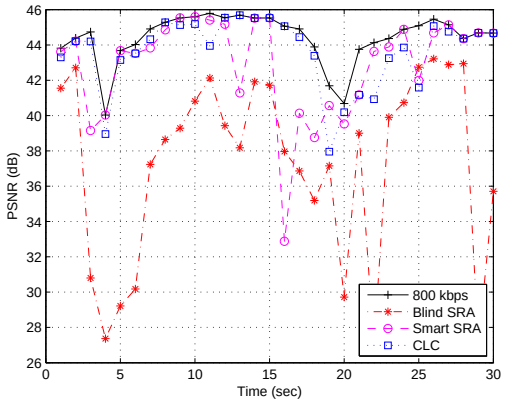


Figure 4.9: Representative PSNR at 11 Mb/s station for testing the case where stations join the BSS at 15 second. During the transition period (15 to 20 s), there is a significant drop of more than 10 dB in PSNR under Smart SRA, while CLC stays within 4 dB of the ideal PSNR curve.

in simulations only) is also observed low at 3% in CLC. In fact, for some cameras (5, 6, and 7), packet loss is completely eliminated for the duration of the experiment. In contrast, Blind SRA suffers from both jagged behavior as well as sustained periods of packet loss. It is thus clear that CLC offers significant improvement in PLR performance, and hence, quality.

4.4.3 Varying the Number of Wireless Stations

Changing the wireless camera deployment (e.g. new cameras joining the BSS) may lead to sudden congestion. We again compare performance under various adaptation schemes. The scenario is setup according to our previous results where $N = 6, 7$ is the maximum limit at which the system can sustain high quality video at the 11 Mb/s stations. The test begins with $N = 6$, and after 15 s, a new 11 Mb/s PHY station joins the BSS. We focus our observations on both the average quality during first 15 s and also on the behavior of each scheme around the transition point (the 15th s). For advanced quality comparison, video clips from real indoor surveillance are used. We carefully selected scenes with employees walking by (i.e. proper amount of motion) to avoid unexpected high quality due to static scenes. Then, we evaluated the objective video quality using PSNR as a performance metric.

The four curves shown in Fig. 4.9 are representative curves randomly selected from one of the 11 Mb/s stations. The black line is a baseline showing the best possible quality for 800 kb/s video under perfect conditions. Blind SRA, Smart SRA, and CLC are shown in red, magenta, and blue respectively.

Prior to the joining of a new station, both Smart SRA and CLC are able to provide good quality close to the ideal case. Blind SRA is the only scheme that suffers low video quality due to lower throughput and higher PLR. Around the transition period (15-20 s), Smart SRA suffers a significant drop of more than 10 dB in PSNR, while CLC stays within 4 dB of the ideal PSNR curve.

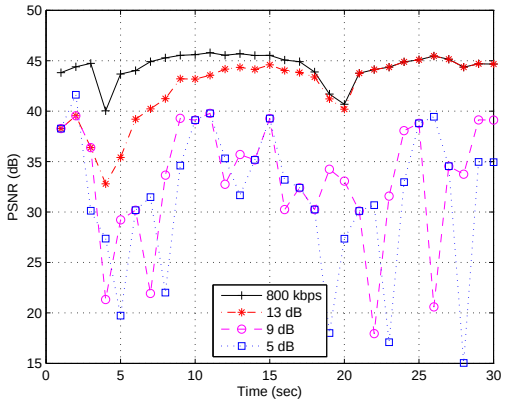
Thus, we have observed the effectiveness of the proposed CLC framework when new stations join the BSS, but we have also illustrated that only source rate adaptation is not enough; CWA is needed during transition regions.

4.4.4 Robustness Under Link Adaptation

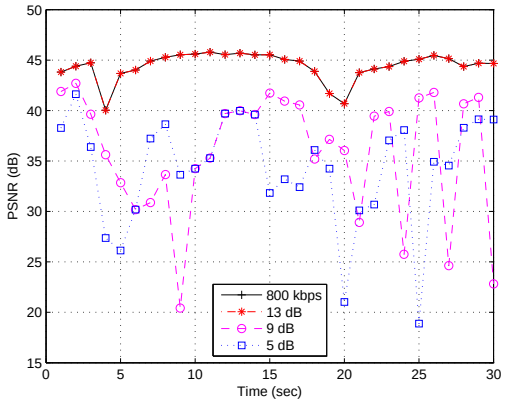
In this final test, we examine the system further with the practical scenarios of varying channel quality and dynamic PHY rate selection by suboptimal rate adaptation schemes found in today's off the shelf equipment.

This part of the examination is only performed in verified NS2 implementation so we could fully control the channel quality and link adaptation scheme for better analysis. The link quality is simulated by the received SNR at the access point. The received bit error rate (BER) can then be obtained according to reference chip specifications [46]. With respect to the link adaptation scheme, we implement the classic adaptive auto rate fallback (AARF) [47] method instead of testing the proprietary scheme embedded in our wireless cameras.

Video quality is compared under various channel qualities (SNR=13, 9, 5 dB shown in red, magenta, and blue, respectively) against a baseline 800 kb/s video (black curve) with perfect link condition. Fig. 4.10 demonstrates the results for the two best schemes (Smart SRA and CLC). In general, CLC provides better than average PSNR, and Smart SRA suffers more frequent dips in PSNR. This indicates that reacting to frequent PHY rate transitions solely on a slower time scale at the APP layer is not sufficient - even



(a) Smart SRA



(b) CLC

Figure 4.10: PSNR results for two schemes: Smart SRA (left) and CLC (right). CLC provides better average PSNR, and Smart SRA suffers more frequent and sharp drops in quality.

with FATE bandwidth bounds. On the other hand, as shown in Fig. 4.10(b), CLC (CWA plus Smart SRA)'s PSNR curve tracks the ideal 800 kb/s curve very closely. Furthermore, there is more quality gain among the lowest tiers of link quality (5 dB and 9 dB SNR) when using CLC. As expected, we have observed that CWA combined with Smart SRA has the power to overcome the aforementioned obstacles associated with real-time uplink video over WLAN.

4.5 Conclusions

We propose a distributed cross-layer control (CLC) architecture that facilitates healthy operation of real-time uplink video over multirate WLAN applications. The use of multiple PHY rates in 802.11 induces serious performance issues in real application scenarios. The CLC implementation uses a fast frame-by-frame control loop in the MAC layer while simultaneously exploiting the powerful control-loop gain attainable by source-rate adaptation in the APP layer. CLC improves aggregate and per-station throughput, PLR, reliability, and ultimately results in a better user experience than APP layer adaptation schemes. We demonstrate these results both in simulation and experimentation in a real wireless video surveillance application.

Chapter 5

Cross-Layer Link Adaptation for Wireless Video

5.1 Introduction

Streaming video is becoming an increasingly important application for wireless networks. Unfortunately, link adaptation mechanisms used in commodity wireless LANs, such as IEEE 802.11, react improperly to perceived packet losses. First, they do not consider that in congested networks, many packets fail due to collisions rather than channel errors, and consequently, they improperly switch to lower PHY rates, thus increasing the collision probability even further. Second, they do not consider the properties of the video codec, and hence, are unaware of the impact of PHY rate selection on the perceptual quality of the received video. In order to provide video services, it is necessary to overcome these limitations.

Link adaptation refers to techniques for dynamically and adaptively choosing modulation schemes according to channel conditions. The modulation schemes that yield high PHY rates (e.g. 54Mbps) are fragile and susceptible to corruption from interference. On the other hand, more resilient modulation schemes can be employed at the expense of lower PHY rates. There is a clear need for such techniques in order to use a wireless channel effectively.

One problem with today's link adaptation mechanisms is that they rely on MAC-level statistics to make their rate selection decisions. However, such MAC-level statistics do not necessarily correlate with the perceived quality of the received video. Indeed, in modern video codecs such as MPEG4 Part 2 and H.264, the received video quality depends on:

1. Error Concealment and Resilience: Modern video codecs offer a variety of techniques (e.g. Previous Frame Copy, Flexible Macroblock Ordering, etc.) for concealing or being resilient to errors due to lost packets [48, 13].

2. Quantization: If the video bitrate must be scaled (e.g. by changing the quantization parameter) to accommodate a different PHY rate, then video distortion will be impacted according to the codec's Rate-Distortion model.

3. GOV Structure: The GOV length (defined as I+P if I:P is the I-to-P frame ratio) plays a central role in determining video quality. One reason is that errors in I-frames may propagate to the other P-frames in the GOV.

4. Packetization and Decoding: If a frame's macroblocks are fragmented into several packets, and one packet is lost, then some decoders may discard the entire frame, while other decoders may try to conceal the missing macroblocks and display the frame as-is.

A second problem with today's link adaptation mechanisms is that they do not differentiate between packet errors caused by collisions and those caused by a bad channel. A consequence of such a design is that the responses taken may not be suitable for alleviating the actual cause of transmission failure and might, in fact, worsen the problem. In particular, an algorithm that assumes errors are due to poor channel conditions will severely malfunction if the errors are actually due to collisions, and vice-versa [25]. For video configurations typically used in surveillance applications, it can be shown that simultaneous wireless video streaming involving more than one camera has a non-negligible collision probability [25, 31, 49].

Indeed the seriousness of these difficulties is indicated by several recent papers. In [6], the authors propose a link adaptation strategy based on perceived distortion at the receiver. However, they only consider the case of a single wireless video source, and therefore do not consider collisions/network stability. In [30, 28, 50] the authors motivate the need for cross-layer (e.g. codec-aware) adaptive algorithms in wireless video, and then propose theoretical and system-level architectures. In [25], the authors detail the severe malfunctions that occur when rate adaptation algorithms assume packets are

failing due to a bad channel when they are really failing due to collisions. In [15, 14, 16] the authors propose new link adaptation algorithms that aim to differentiate between packet failures due to collisions and those due to a bad channel.

In this chapter, we address the aforementioned difficulties by proposing a new class of link adaptation algorithms, called CLLA (Cross Layer Link Adaptation), designed for real-time wireless video transmission. This new class of link adaptation algorithms incorporates knowledge of the codec (e.g. concealment, quantization, GOV structure, packetization), the collision probability, and the physical channel conditions. For each GOP (a GOP, or Group of Pictures, is defined as a single I-frame followed by zero or more P-frames as determined by the I:P frame ratio), this information is used to estimate a video quality metric for the current, and adjacent PHY rates. The algorithm chooses the PHY rate associated with the best perceptual quality.

The rest of this chapter is organized as follows: After giving an overview of this new class of algorithms in Section 5.2, we detail a particular instance from this class in Section 5.3 (another instance based on perceptual VQA is proposed as future work to complete the dissertation, and will be based on the results of our endeavors in Chapter 5). A performance analysis follows in Section 5.4. Finally, we conclude the paper in Section 5.5.

5.2 Cross Layer Link Adaptation

Figure 5.1 shows the CLLA system for a single video source, such as a wireless camera (in the rest of this paper, we assume the video sources are wireless cameras). The system is distributed, so each camera runs the algorithm independently. The only information shared between cameras is N, the number of cameras in the 802.11 Basic Service Set (BSS).

The CLLA module operates as follows on each camera: First, a subset of codec-related parameters is chosen based on availability and tradeoffs between optimality and practicality. As shown in Figure 5.1, this subset is chosen from the four inputs to the *Codec* selector (these inputs correspond to the items in the bulleted list in the

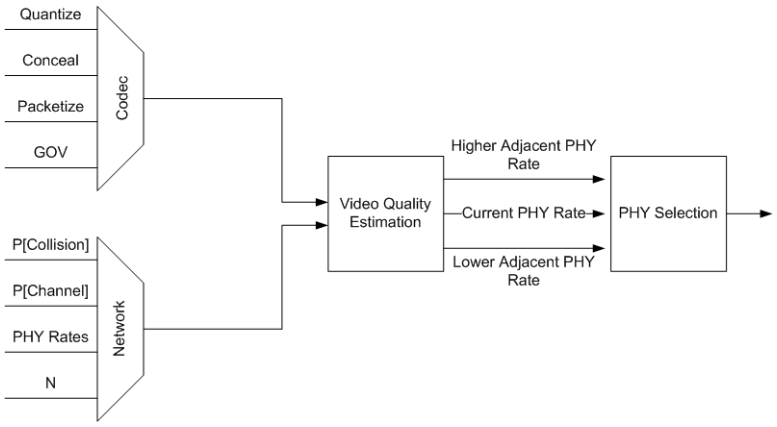


Figure 5.1: CLLA System Diagram.

Introduction). At the same time, a subset of network-related parameters is chosen based on similar tradeoffs. The set of possible network-related parameters is shown in Figure 5.1 as the four inputs to the *Network* selector ($P[Collision]$ is the packet collision probability, $P[Channel]$ is the channel error probability, *PHY Rates* are the currently selected and adjacent PHY rates, and N is the number of cameras in the BSS). The chosen subset of codec and network parameters represents cross-layer information about the system.

This cross-layer information is input to a module that estimates a video quality metric, which is chosen according to the application. Some examples of quality metrics are: Peak-to-Peak Signal-to-Noise Ratio (PSNR), Mean Absolute Difference (MSAD), Structural Similarity Index (SSIM) [4], VQM [51], and Mean Squared Error (MSE). The quality metric is estimated at the current and adjacent PHY rates, and the algorithm selects the PHY rate that yields the best quality score.

Since a camera does not have access to the actual received video frames, the quality metric must be computed in a no-reference mode (i.e. any comparisons between transmitted and received video frames must be estimated blindly). One way to estimate information about the received video is to observe the packet loss rate, and then use knowledge about the codec to estimate how the received video will be affected. The details of this estimation process depend greatly on which quality metric is chosen, and also on the contents of the codec and network parameter subsets. In the rest of this chapter, we describe in detail one particular instance of CLLA (where an instance

is defined by the choice of quality metric, and the contents of the codec and network subsets).

5.3 MSE-Based Instance of CLLA

The instance of CLLA described in the rest of this chapter uses MSE as the quality metric (though, a perceptual VQA extension is also proposed based on our endeavors in Chapter 5 which are currently underway). We use MPEG4 as the video codec, and 802.11b as the wireless technology. We have experimented with other codecs, but for simplicity of discussion we do not describe them in this paper.

For each GOP, the algorithm estimates the MSE at the current and adjacent PHY rates, and chooses the PHY rate that minimizes the MSE over the GOP. Although the idea itself is simple, we make two major contributions: (1) developing a no-reference estimate for the received MSE at the current and adjacent PHY rates; (2) predicting the packet loss rate at the adjacent PHY rates. These two steps are detailed in the next two subsections.

5.3.1 Estimating the MSE

We can approximate the MSE for a GOP as the linear decomposition $MSE = MSE_{PLR} + MSE_Q + \theta$, where MSE_{PLR} is the MSE due to packet loss, MSE_Q is the MSE due to quantization, and θ is a correction factor for any error resilience or concealment. For simplicity, we ignore θ in this instance.

Traditional MPEG4 video streams are either CBR (Constant Bit Rate) or VBR (Variable Bit Rate), though more sophisticated bitrate scaling mechanisms are also possible [31, 49]. In any case, MSE_Q accounts for the distortion due to quantization, and can be obtained from the Rate-Distortion model (which can either be computed or measured empirically) of the video source as a function of the bitrate. For simplicity, we assume CBR traffic in this instance, and therefore, MSE_Q is computed once, and does not change.

MSE_{PLR} must be estimated blindly (due to the no-reference constraint mentioned

earlier) for the current and adjacent PHY rates. Therefore, we must make assumptions about the values of pixels impacted by packet loss. If we ignore any error resilience or correction techniques, then an impacted pixel will take on a random value between 0 and 255 (for 8-bit depth). Therefore, one reasonable assumption is that an impacted pixel's value is centrally biased, and has an average error of half the maximum range of pixel values. For example, with 8-bit color depth, pixel values range from 0 to 255, and on average, a lost pixel will have an error of 255/2. Therefore, the MSE due to packet loss for a GOP is estimated as [52]

$$MSE_{PLR} = \frac{N_i}{P} \cdot \left(\frac{V}{2}\right)^2 \tag{5.1}$$

where N_i is the number of lost/impacted pixels in the GOP, V is the maximum pixel range (255 for 8-bit color depth), and P is the total number of pixels in the GOP (i.e. the number of pixels per picture (e.g. 640x480) multiplied by the GOV length (i.e. the number of pictures in the GOP)).

The remaining task is to estimate N_i at the current and adjacent PHY rates. Assuming the packet loss rate is known, then in a GOP, the number of I-Packets (packets carrying data belonging to an I-Frame) lost is $L_i = \eta(r) \cdot n_{p,i}$, where $\eta(r)$ is the estimated packet loss rate (PLR) at PHY rate r, and $n_{p,i}$ is the number of packets required to transmit a single I-Frame [1]. Similarly, in a GOP, the number of P-Packets (packets carrying data belonging to a P-Frame) lost is $L_p = \eta(r) \cdot (n_{p,p} \cdot (GOV - 1))$, where $n_{p,p}$ is the number of packets required to transmit a single P-Frame.

The numbers $n_{p,i}$ and $n_{p,p}$ vary with the amount of motion in the scene, the resolution, bitrate, and framerate, but their approximate values can be obtained either empirically or theoretically.

For each I-Packet lost, the number of affected pixels in the GOP is

$$N_{i,i} = \frac{1}{n_{p,i}} \cdot (M + M \cdot (GOV - 1)) \tag{5.2}$$

where M is the number of pixels per frame (e.g. 640x480). The first term in the sum

[1] For wireless networks, $n_{p,i}$ will almost certainly be greater than 1 since the maximum MTU in 802.11 is about 1536 bytes - much less than the size of an I-Frame.

says that each I-Packet contains $1/n_{p,i}$ of the M pixels in the I-Frame. The second term in the sum accounts for propagation errors. Since each P-Frame predicts from the preceding I-Frame, loosing $1/n_{p,i}$ of the pixels in that I-Frame implies loosing $1/n_{p,i}$ of the pixels in all the P-frames in that GOP. If the codec employs a technique to eliminate such propagation errors, then the second term can be omitted.

For each P-Packet lost, the number of affected pixels in the GOP is

$$N_{i,p} = \frac{1}{n_{p,p}} \cdot M. \tag{5.3}$$

Therefore, N_i can be expressed as

$$N_i = (L_i \cdot N_{i,i}) + (L_p \cdot N_{i,p}) \tag{5.4}$$

$$= \eta(r) \cdot (M(1 + 2(GOV - 1))) \tag{5.5}$$

and we can plug it into Equation 5.1 to solve for MSE_{PLR}.

The process for estimating MSE_{PLR} is the same for the current and adjacent PHY rates. The only difference is the value of $\eta(r)$, which we show how to estimate next.

5.3.2 Estimating the Packet Loss Rate

In 802.11 WLAN, the PLR is defined as the frame error rate (FER) raised to the power of the retry limit [20]. Frames fail due to collisions (p_c) or channel errors (p_e), but not both ($FER = p_c + p_e - p_c p_e$). For a BSS (802.11 Basic Service Set) with more than 1 station (wireless device), both p_e and p_c depend on the PHY rate, so $\eta(r) = (p_e(r) + p_c(r) - p_e(r)p_c(r))^R$, where R is the retry limit. At the current PHY rate, $\eta(r)$ can be measured by the wireless driver, and therefore, is known to each camera. To find $p_e(r)$ at the current PHY rate, we assume the channel is symmetric, and lookup (using empirical or theoretical BER/SNR curves) the corresponding bit error rate (BER) for that PHY rate at the currently measured SNR. Thus, at the current PHY rate, $p_e(r) = 1 - ((1 - BER)^L)$, where L is the packet length in bits. Consequently, at the current PHY rate,

$$p_c(r) = \frac{\eta(r)^{1/R} - p_e(r)}{1 - p_e(r)}. \tag{5.6}$$

Our goal is to estimate $\eta(r)$ at the adjacent higher and lower PHY rates given $\eta(r)$ at the current PHY rate. At the adjacent PHY rates, $p_e(r)$ can be obtained from the BER/SNR curves, as it was for the current PHY rate. It remains to find $p_c(r)$.

The Markov model of the 802.11 DCF first introduced by Giuseppe Bianchi [19] leads to formulas that give p_c as a function of $\hat{N}$ (the number of contending stations in a saturated BSS), and a set of several other parameters, Ω: $p_c = f(\Omega, \hat{N})$ for saturated (the probability that any station's queue is empty is zero) systems. However, we cannot directly use these formulas because: (1) they represent a fixed point formulation that requires numerical techniques to solve for the collision probability solely as a function of $\hat{N}$; and (2) a multi-camera wireless network is not at all likely to be saturated.

Obstacle (1) above can be overcome by solving a fixed point formulation *a priori*. The result is a function $p_c = f(\hat{N})$, which happens to behave logarithmically, and thus is invertible over our domain of interest, $\hat{N} > 0$. Obstacle (2) above can be overcome by converting the current system from an unsaturated one with N stations to a saturated one with $\hat{N}$ stations by solving the inverse function $f^{-1}(p_c) = \hat{N}$, where p_c is known at the current PHY rate.

This gives us $\hat{N}$ for the current PHY rate ($\hat{N_c}$). If we can somehow get $\hat{N}$ for the adjacent PHY rates ($\hat{N_l}$ and $\hat{N_h}$), then we can solve $p_c = f(\hat{N_{l|h}})$ to get the collision probability at the adjacent rates.

To get $\hat{N_l}$ and $\hat{N_h}$, we begin with the ratio $\rho = \hat{N}/N$. One interpretation of this ratio is that it indicates how frequently an arbitrarily selected station's queue is nonempty (in the unsaturated system). Fundamental queuing theory tells us that the number of packets in a queue is determined entirely by the arrival rate to the queue and the service rate of the queue. If we assume that the video bitrate does not change with the PHY rate (we have considered this case, but omitted it for clarity), then the only thing that changes at the adjacent PHY rates is the service rate. We want to know how much the global service rate, and hence, ρ, changes if a single station switches to an adjacent PHY rate. The precise way in which the service rate changes with the PHY rate is a complex process, so we must make some approximations.

From the perspective of the station (call it station α) switching its PHY rate, we

approximate $\rho_\alpha = \frac{r_c}{r_s} \cdot \rho$, where r_c is the current PHY rate and r_s is the adjacent PHY rate being considered. From the perspective of the $N-1$ stations (call them β) not switching their PHY rate, they do not know the PHY rates of any other stations in the BSS, so we err on the side of caution and assume that they are all at the maximum PHY rate, r_{max}. Then, at the current PHY rate, the normalized percentage of service time of one of the $N-1$ max-rate stations is

$$a_c = \frac{T(r_{max})}{(N-1) \cdot T(r_{max}) + T(r_c)} \tag{5.7}$$

where for 802.11b, r_{max} is 11 (Mbps); $T(r) = T_{DAT} + T_{OV} + T_{BACK}$; $T_{DAT} = L/r$ (where L is the packet length in bits); $T_{OV} = DIFS + 2 \cdot t_{preamble} + SIFS + t_{ACK}$; $T_{BACK} = \frac{CW_{min}}{2} \cdot SLOT$; $t_{ACK} = (14 \cdot 8)/r$ (an ACK is 14 bytes); and $t_{preamble}$ is 96 microseconds for $r = \{11, 5.5, 2\}$, and 192 microseconds for $r = \{1\}$.

If an adjacent PHY rate is selected, the percentage of service time, a_s, seen by one of the $N-1$ max-rate stations is the same as a_c except we use r_s in the denominator of Equation 5.7 instead of r_c. The ratio $\gamma = a_s/a_c$ gives the relative change in service time-share between the current and adjacent PHY rates for the $N-1$ stations not changing their PHY rate, so for these stations, $\rho_\beta = \rho + \rho \cdot (1-\gamma)$.

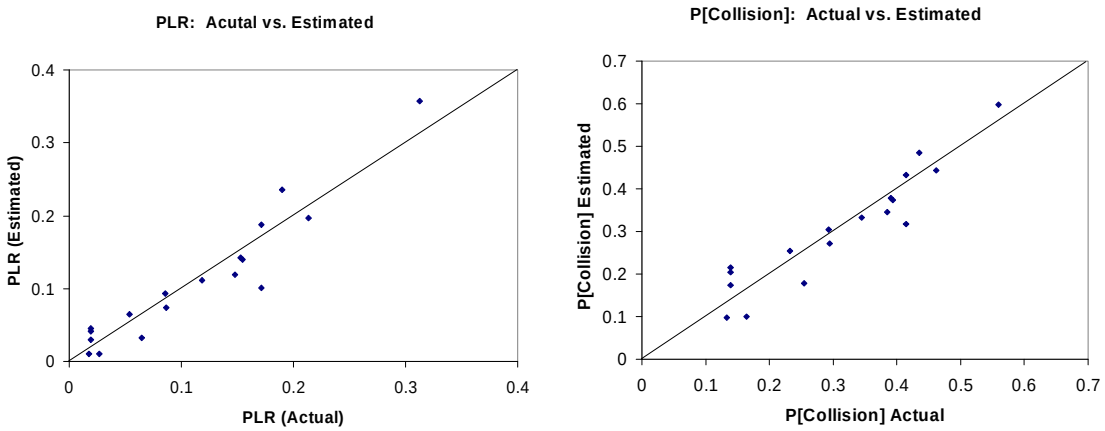
Overall, at the adjacent PHY rate, ρ changes to ρ' according to a weighted average of ρ_α and ρ_β

$$\rho' = \frac{(N-1) \cdot \rho_\beta + \rho_\alpha}{N}. \tag{5.8}$$

Then, $\hat{N}$ at the adjacent PHY rate (l or h) is $\hat{N_{l|h}} = \rho' \cdot N$. The collision probability at the adjacent PHY rate is obtained by plugging $\hat{N_{l|h}}$ into the function $p_c = f(\hat{N_{l|h}})$. It follows that the total PLR at the adjacent rate is $\eta(r) = (p_e(r) + p_c(r) - p_e(r)p_c(r))^R$. We experimentally validated the performance of our estimate, and as shown in Figure 5.2, we found that our estimate performs relatively well.

5.4 Performance Analysis

Next, we evaluate experimentally the performance of CLLA based on the improvement in MSE, PLR, throughput, and reliability, and ultimately show that CLLA performs



(a) (b)

Figure 5.2: Scatter plots of actual vs. estimated (a) packet loss rate, and (b) collision probability reveal that our estimate performs relatively well.

better than today's most common off-the-shelf link adaptation algorithm, Auto Rate Fallback (ARF) [22].

5.4.1 Experimental Setup

We implemented CLLA on Axis 207w wireless IP cameras from Axis Communications. The Axis 207w cameras run a lightweight embedded version of Linux, and we implemented the CLLA algorithm as a shell script, which is approximately 1000 lines of code. By default, Axis 207w cameras use ARF as the link adaptation algorithm.

Our setup is based on a typical surveillance/monitoring application: Six cameras, scattered spatially, stream video (MPEG4 constant bitrate 600 Kbps, GOV=15, Resolution=640x480, Framerate=15), wirelessly (IEEE 802.11b, Marvell 88w8385-BDK1 Chipset, cf8385 driver) to a security center (Dell OptiPlex GX280) via a single access point (Siemens AP2630). The OptiPlex is wired to the AP, so there is only one wireless hop (from the cameras to the AP). The cameras are looking at scenes recorded from inside a major metropolitan subway station, and on average, the amount of motion in all scenes is the same for all cameras. We monitored system statistics in real-time using custom plugins we developed for WildPackets AiroPeek [17], and on each camera, our shell script dumped statistics computed from inside the CLLA algorithm. Of the six cameras, five remained stationary at locations with very high SNRs, and one

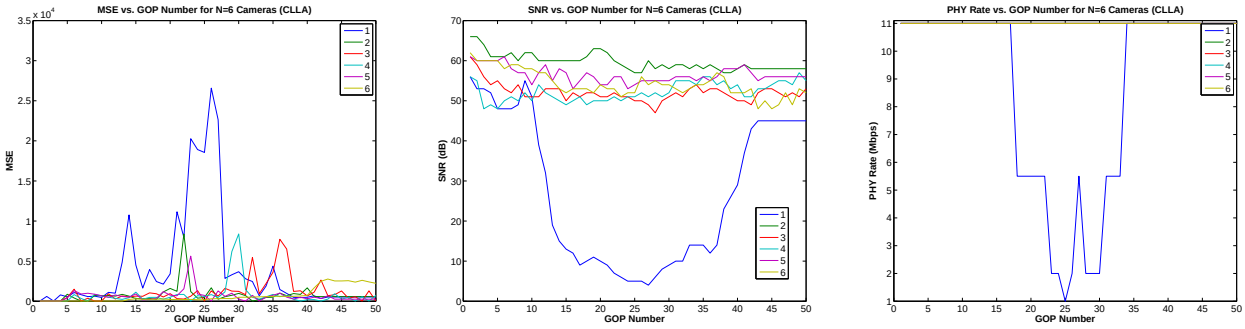
camera was mobile and visited stops along a path that covered a variety of SNRs. The experiments were conducted on a clean, isolated wireless channel.

5.4.2 Results

Figure 5.3 shows the MSE, SNR, and PHY rates at each GOP for each camera, with $N = 6$ cameras in the BSS, with each camera running CLLA. Camera 1 is mobile and follows a path on which the SNR varies from very high to very low, and the other cameras are stationary at locations with a high SNR. Figure 5.4 shows the same thing, but each camera is running the off-the-shelf link adaptation algorithm, ARF (no CLLA). Comparing the two figures, we see that even with essentially the same channel conditions (c.f. Figures 5.3(b) and 5.4(b)), the MSE for each camera is dramatically lower for CLLA than it is for ARF (c.f. Figures 5.3(a) and 5.4(a)). This is best explained by looking at the PHY rate selection plots (5.3(c) and 5.4(c)). In both cases, the five stationary cameras have a perfect SNR, and therefore, no reason to use a PHY rate lower than r_{max} (11 Mbps for 802.11b). Indeed, in CLLA, the five stationary cameras stay at 11 Mbps, and the one mobile camera switches its PHY rate according to a balance between p_e (due to the SNR) and p_c (due to channel contention from the other cameras). In ARF, however, all of the stations continuously switch PHY rates, and never find a stable state. This is because ARF cannot distinguish between packet loss due to collisions (p_c) and loss due to a bad channel (p_e), and therefore, when it observes high packet loss (which for the $N - 1$ stationary cameras, is entirely due to collisions since their SNR is perfect),

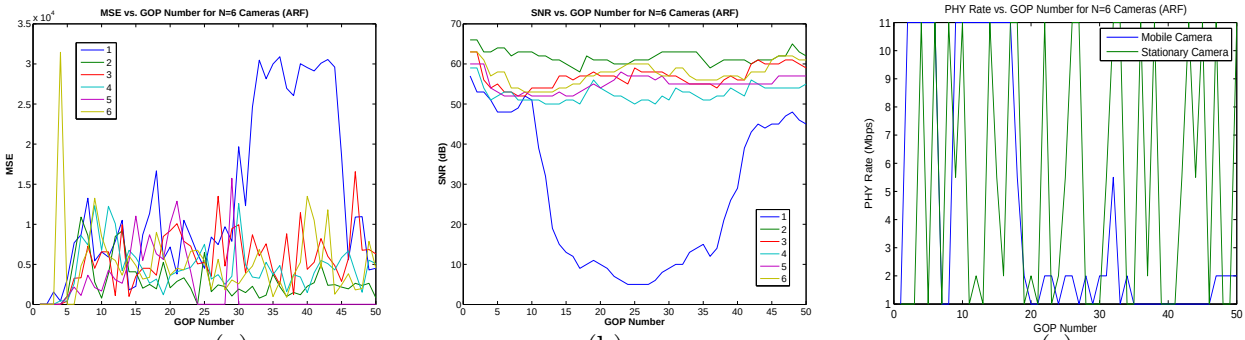
it assumes the loss is due to the channel, and therefore switches to a lower PHY rate, thus increasing the collision probability even further. This behavior is characteristic of the Snowball Effect described in [25].

Figure 5.5 shows the MSE, SNR, and PHY rates at each GOP for the mobile camera in a scenario where it is the only camera in the BSS ($N = 1$). In this scenario, all packet loss is due to the channel since collisions are impossible with only one station in the BSS. The SNR curve for the mobile camera (camera 1) is essentially the same in Figures 5.5(b) and 5.3(b), but comparing Figures 5.5(c) and 5.3(c), we see that camera 1 is more



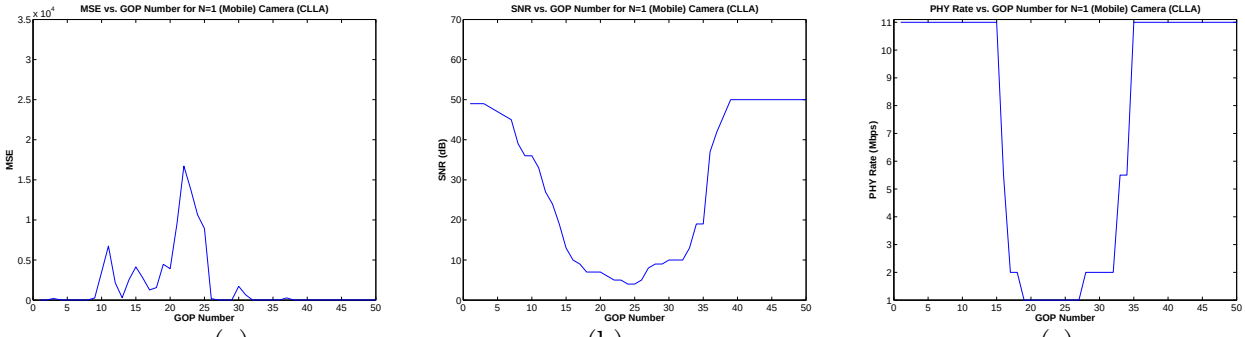
(a) (b) (c)

Figure 5.3: MSE, SNR, and PHY results at each GOP for a BSS with 6 cameras using CLLA link adaptation. Camera 1 is mobile and follows a path with varying SNR. The other five cameras are stationary at a high SNR.



(a) (b) (c)

Figure 5.4: MSE, SNR, and PHY results at each GOP for a BSS with 6 cameras using ARF link adaptation. Camera 1 is mobile and follows a path with varying SNR. The other five cameras are stationary at a high SNR.



(a) (b) (c)

Figure 5.5: MSE, SNR, and PHY results at each GOP for a BSS with only 1 camera which is using CLLA link adaptation. The camera is mobile and follows a path on which the SNR varies from very high to very low.

hesitant to use low PHY rates with $N = 6$ than it is with $N = 1$. Indeed, with $N = 1$ cameras, the collision probability is zero, but with $N = 6$ cameras, it is significant, so the PHY rates selected in Figure 5.5(c) are chosen entirely according to the SNR (p_e), while in Figure 5.3(c), the PHY rates are chosen to balance p_e with p_c. Since p_c increases when a station switches to a lower PHY rate, the net result is that under CLLA, camera 1 tends to use higher PHY rates with $N = 6$ than it does in isolation ($N = 1$).

Figure 5.6 shows the aggregate system throughput for the above scenarios (6 Cameras CLLA, 6 Cameras ARF). Under CLLA, the average aggregate throughput is approximately 3.3 Mbps. Indeed, for six cameras, each of which is sending video at 600 Kbps constant bitrate, we expect the aggregate throughput to be about $600Kbps \cdot 6 = 3.6Mbps$. Under ARF, the system only achieves an average aggregate throughput of about 1.2 Mbps. This is because under ARF, the system suffers very high packet loss rates due to collisions and the Snowball Effect, while under CLLA, the PLR is very low since the algorithm considers the impact of PHY selection on the collision probability.

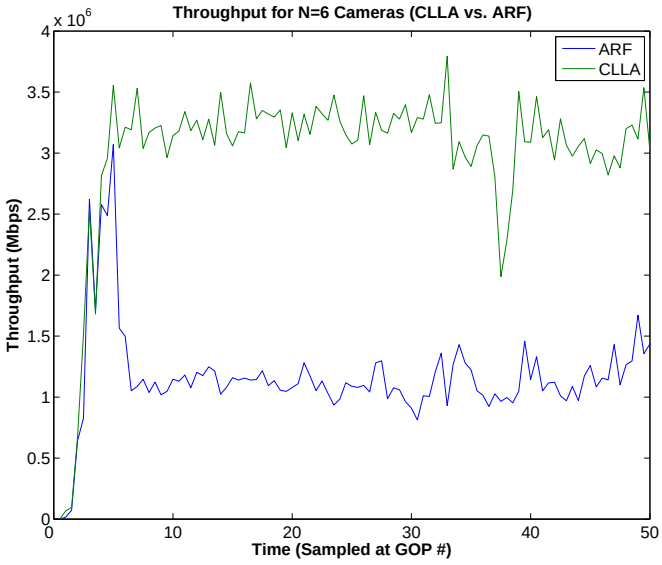


Figure 5.6: Aggregate throughput for 6 cameras under CLLA and ARF. CLLA more than doubles the steady-state aggregate throughput achieved by ARF.

Figure 5.7 shows the MSE averaged over all GOPs for each camera for the above scenarios (6 Cameras CCLA, 6 Cameras ARF). CLLA clearly outperforms ARF in

terms of MSE. In fact, as indicated by the last bar (which is the MSE averaged over all GOPs and over all cameras), the total average MSE for CLLA is over four times lower than it is for ARF (5555.06 for ARF vs. 1309.85 for CLLA).

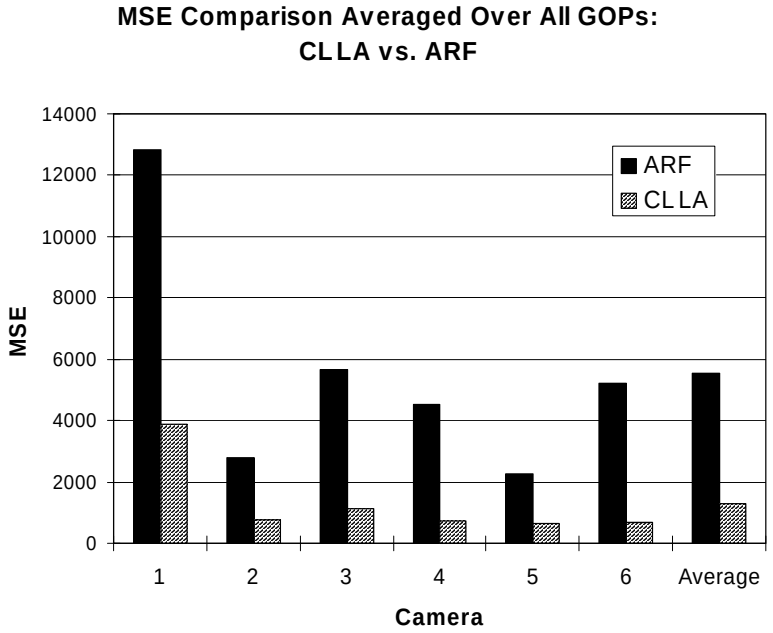


Figure 5.7: MSE averaged over all GOPs for 6 cameras under CLLA and ARF. The last bar is an average over all GOPs and over all cameras, and shows that CLLA results in more than a 4x reduction in MSE compared to ARF.

It is thus clear that CLLA offers significant improvement over ARF in terms of MSE, PLR, throughput, and reliability.

5.5 Conclusion

We proposed a new class of link adaptation algorithms designed for transmission of video over wireless LAN. Current link adaptation schemes for IEEE 802.11 WLANs exhibit behavior which makes them unsuitable for transmission of multiple simultaneous real-time uplink video streams. First, they do not consider properties of the video codec, and thus are unaware of the interplay between PHY rate selection and error resilience, quantization, GOV structure, and packetization. Second, they do not consider that in congested networks (such as those with multiple wireless cameras transmitting simultaneously), many packet failures may be due to collisions, and not a bad channel. Our new class of algorithms, called CLLA (Cross Layer Link Adaptation) accounts for

both of these issues, and selects PHY rates based on a perceptual quality metric. We described in detail one instance of an algorithm from the CLLA family which used MSE as the perceptual quality metric. We implemented CLLA on real cameras and gave a detailed performance analysis which showed that CLLA resulted in more than a 4x reduction in MSE compared to ARF, and more than doubled the steady-state aggregate throughput achieved by ARF.

Chapter 6

Video Quality Assessment in an Adaptive Real-Time Cross-Layer Wireless Video Streaming System

In this chapter, we explore video quality assessment in the context of real-time adaptive video streaming systems. With the evolution towards user- and machine-centric wireless video streaming applications underway, it is becoming progressively more important to ensure that distortions in the received video signal have minimal impact on the end-use application or task. This is in contrast to traditional QoS, which aims to optimize system and packet-level performance, but not necessarily perceptual or task-based video quality. In order to drive towards this goal, one of the most important evolutions to modern video streaming systems will be the adaptive allocation of both wireless and video coding resources at the transmitter according to a cost function designed to optimize some task- or user-centric measure of the received video quality. Today's cross-layer systems are based on mathematically simple, but perceptually inaccurate distortion metrics like MSE/PSNR. Perhaps this is not surprising, since aside from the great difficulty in designing perceptually accurate video quality metrics themselves, there are a number of other challenges in using such a metric in a real-time adaptive system. They include: (1) estimating the quality metric at the encoder with no direct access to the received video signal; (2) predicting how the perceptual quality would change if some adjustment is made to the instantiation of the parameter space either in the video codec or in the wireless PHY/MAC; and (3) how to balance complexity with accuracy for suitability in a real-time system with potentially limited computation and memory resources. It is our hope that by solving these issues, we can not only improve the performance of existing cross-layer approaches, but open the door for new approaches to real-time adaptive cross-layer video streaming systems. Indeed,

in Chapter 7, we show how we can use the solutions to these problems to perform joint link adaptation and congestion control driven by user/task-centric resource allocation.

A generic cross-layer adaptive real-time video streaming system which leverages the solutions to these problems might look something like Figure 6.1. We'll refer to this generic figure periodically in the rest of the dissertation to help illustrate certain concepts.

We begin with an overview of distortions found in digital videos. After that, we present the state of the art in video quality assessment, and investigate how well perceptually-oriented video quality metrics can predict the performance of task-specific applications with non-human observers, such as object tracking. Finally, we propose solutions to issues (1) and (2) listed above (item 3 (balancing complexity with accuracy) is covered implicitly).

6.1 Distortions in Digital Videos

In general, distortions found in digital videos can be classified either as spatial or temporal. In this section, we review the most common types of distortions found in natural digital videos [53].

6.1.1 Spatial

Blur

Blur appears to the viewer as a reduction in the sharpness of edges in areas of moderate to high spatial frequency (e.g. edges or highly textured areas). It is primarily caused by the DCT quantization process, which attenuates the higher order DCT coefficients (e.g. the high frequencies).

Ringing

Fundamentally associated with Gibb's phenomenon, ringing appears as an outward shimmering along sharp contrast edges positioned against relatively smoothly textured backgrounds. It is caused by attenuation of the higher-order AC coefficients, but not to

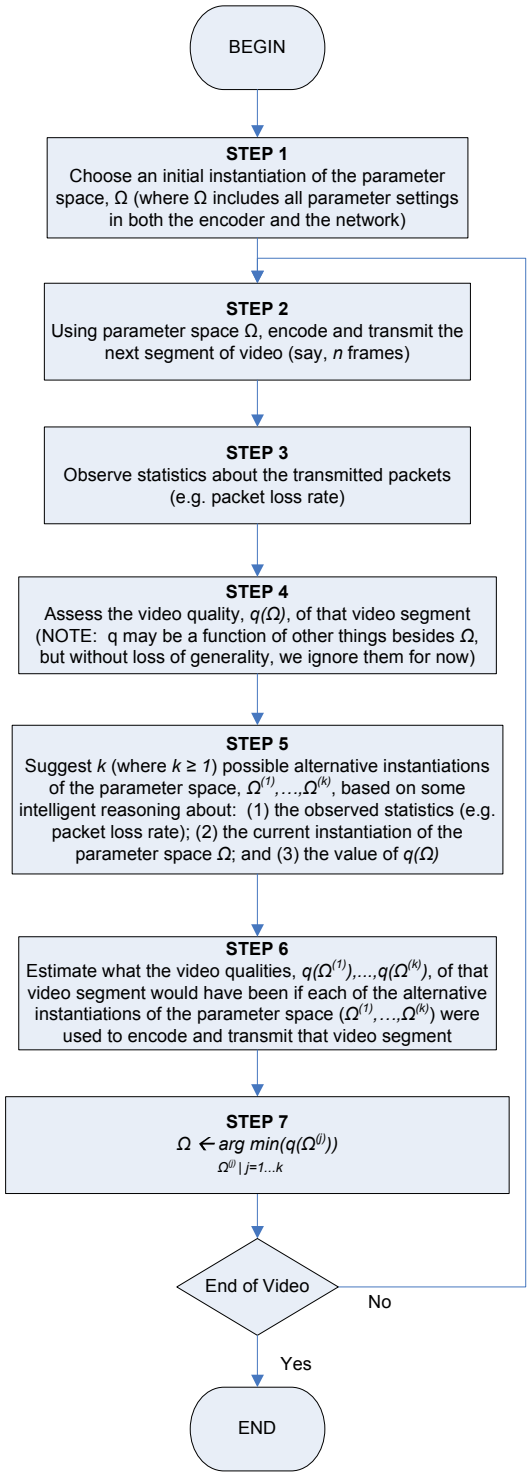


Figure 6.1: A generic vision of what a perceptual quality-based cross-layer adaptive video streaming system might look like.

the extent of blurring (which is often associated with total nullification of the higher-order coefficients).

Blocking

Blocking is found most frequently in DCT-based compression methods. It appears as a discontinuity along adjacent macroblock boundaries (which are block-shaped), hence giving the name "'blocking"'. It is cause by the fact that macroblocks are coded independently, and therefore, the quantization process is not at all likely to result in the same visible distortions in any two macroblocks. Intuitively, it is easy to understand why this would be most visible across macroblock boundaries. Worth noting is that H.264 features a built-in deblocking filter.

Mosaic

The mosaic effect is caused by the same thing as blocking, but it is only visible among adjacent macroblocks that span a spatial region of varying texture, edges, or contrast (e.g. it is not visible in smooth regions). It appears as if an artist created a tile mosaic of the image, but the tiles do not quite fit together smoothly. Blocking is visible even in regions of smooth texture, but mosaic is difficult to spot in those regions. Again, it is caused by discontinuities in macroblock errors due to quantization.

False Contouring

False contouring occurs in regions where the source image has smoothly and slowly increasing or decreasing pixel values (this frequently occurs in images of the sky, where the brightness varies smoothly from the horizon and upward). If you think of the pixel values "'appearing"' to change continuously, then false contouring can be considered a "'discretizing"' of this process, thus appearing as squares of changing contour. This is a direct result of the quantization process, which has a "'many to one"' mapping effect on the pixel values in that region.

Additive Noise

Additive noise generally appears as a grainy or salt and pepper effect in the image. It is primarily caused by either the video acquisition source or the communication channel itself ("'channel"' here is not restricted to the wireless channel; it could refer to any path that the video signal traverses between the source and your brain (such as the display, for example).).

6.1.2 Temporal

Mosquito Effect

The mosquito effect manifests as temporal fluctuations in luminance (or chrominance), generally along regions of smooth texture surrounding moving objects with high contrast. It is caused by a difference in the coding of iso-spatial macroblocks across time (like a temporal version of blocking or ringing).

Smearing

Smearing is caused by the non-instantaneous nature of exposure on the image acquisition sensor (e.g. the integration time is nonzero). The sensor sees light reflecting off the moving object at different points in time, and hence, at different spatial points, but these are all integrated into a single pixel value. This manifests as a blurring or loss of spatial detail in the direction of motion of the moving object. However, this effect is not likely to be noticed unless the viewer is actively fixated on that moving object.

Jerkiness

Jerkiness occurs when the receiver's jitter buffer is smaller than the interarrival time of packets at the receiver. Just as the name suggest, this effect appears as a jittery/jerky video sequence.

Motion Compensation Mismatch

In predictive-coding, the current macroblock to be coded is predicted from a nearby macroblock. Namely, a full-search of all macroblocks in a local window is performed, and the macroblock from that window which minimizes the MSE between itself and the macroblock to be coded is chosen as the prediction reference. The difference (residual) between the two macroblocks is then computed, coded, and quantized. If the residual is not very large, then MC mismatch is not a big concern. However, if the residual is large (for example, if different objects within the current macroblock have moved at different speeds relative to the prediction source), then the quantization process could yield highly visible distortions in the reconstructed macroblock. Even worse, if the motion vectors are correctly received, but the residual is either nullified (because of quantization) or lost (due to a loss in the wireless channel), then the MC mismatch error will be highly visible to the viewer.

Ghosting

Ghosting is a consequence of the deliberate low-pass temporal filtering that's normally performed to reduce the visibility of sensor noise (e.g. flicker). However, very fast moving objects which are highly uncorrelated with the image are incorrectly averaged out, thus quasi-integrated into the background, and appear to have a ghostly trail behind them.

6.2 State of the Art: Video Quality Assessment

The only true way to assess perceptual video quality is through subjective means. This involves cumbersome, time consuming, and expensive controlled experiments where humans are asked to manually evaluate video quality of a series of test sequences. Because of these constraints, subjective video quality assessment is only useful for benchmarking automated objective prediction of video quality. This has motivated the need for objective video quality assessment.

6.2.1 HVS/Perceptually-Oriented VQA

Full-Reference, Reduced-Reference, and No-Reference VQA

Video quality assessment can be broadly classified into: Full-Reference, No-Reference, and Reduced-Reference. No-Reference VQA is concerned with predicting subjective quality from only a received video signal (i.e. the original reference is not available). The state of the art in NR VQA is the most primitive among the three classes of VQA. Current NR VQA involves searching for the presence of some expected, common distortion types (like blocking and blurring), and quantifying their expected visibility to the viewer. Performance of today's NR VQA metrics is quite poor [3]. Reduced-Reference VQA uses extra information transmitted in a side (or the same) channel. Such information might include the location of edges, shapes, or the embedding of marker bits. Then, this information is used to operate in an "'enhanced"' NR mode. By far, the most popular and accurate technique for VQA is Full-Reference (FR). In this mode, the VQA mechanism operates on both the received video and on the pristine source video. FR metrics are our focus henceforth. Full-Reference VQA can be classified into three broad types, each of which are surveyed below.

HVS-Based Modeling

A large class of VQA methods are based on mathematical models of the neurophysiology and psychophysics of the human visual system. The idea is to make the VQA algorithm "'see"' the video as a human would by simulating the visual pathway of the eye-brain system. Actually, most of the VQA metrics in this class are simple extensions of image quality assessment (IQA) metrics. These IQA metrics basically compute a quality metric in a four-step process, shown in Figure 6.2. First, the reference and test images are pre-processed for (1) scaling and alignment; (2) color space transformation; (3) luminance mapping to account for a particular display device; and (4) low-pass filtering to simulate the point spread function of the eye. Second, the reference and test images are decomposed into a series of channels tuned to spatial frequency and orientation which closely mimics the early stage neurophysiology of the neural responses

in the mammalian's primary visual cortex. Third, the reference and test images are compared, and the error signal is analyzed using models of the CSF (contrast sensitivity function),luminance masking, and contrast masking functionality of the HVS. The output of this stage is an "'error sensitivity"' map for each channel which expresses the anticipated visibility of errors in that channel (and possibly using information from other channels), which is usually expressed in units of just noticeable difference (JND). Finally, errors across channels are pooled and normalized - typically using a Minkowski norm:

$$E(\{e_{i,j}\}) = \left(\sum_i \sum_j |e_{i,j}|^\beta \right)^{\frac{1}{\beta}} \tag{6.1}$$

where $e_{i,j}$ is the normalized error of the j-th coefficient in the i-th channel, and β is a constant exponent typically chosen between 1 and 4.

Many HVS-based VQA metrics are simple extensions of their IQA counterparts, differing only in the introduction of a temporal filter before the channel decomposition block. The role of the temporal filter block is to model temporal mechanisms in the HVS. It is widely believed that two kinds of temporal processing are done in the early stage of the HVS: sustained (lowpass), and transient (sustained). Some of the most well known metrics which use this approach are [54, 55, 56, 57, 58, 59].

There are four main problems with HVS-based VQA. First, they are mathematically complex, and therefore often not suitable in a real-time system. Second, the HVS is still not well understood or modeled properly. Third, these metrics rely on overly simplistic models (for example, CSF models are often based on experiments with white and gray bars, or other synthesized image patters, not natural images). Finally, these metrics are heavily based on IQA, and fail to model many of the key temporal functions of the HVS. [3, 60].

Feature-Based and Top-Down Modeling

An alternative approach to VQA is to extract features from the reference, test, and error signals and use top-down hypotheses about the function of the HVS to pool, normalize,

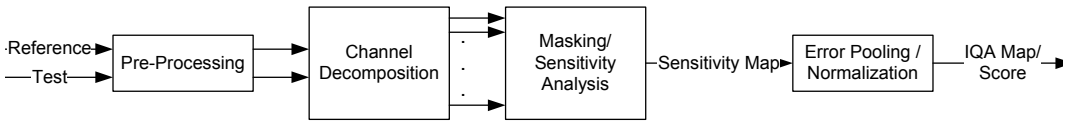


Figure 6.2: Typical system diagram of HVS-based IQA metrics. The inclusion of a temporal filtering block after the pre-processing stage is typical of HVS-based VQA metrics.

and quantify the perceptual impact of these features on the HVS. This is in contrast to the aforementioned bottom-up HVS-based approaches which aimed to mathematically model the neurophysiology and psychophysics of the HVS. However, similar to the HVS-based approach, many of these VQA metrics are simple modifications of IQA metrics, as we shall now discuss.

Feature-Based VQA: One of the most popular feature-based VQA metrics is VQM [51], which computes and pools seven different quality features. VQM has been standardized by ANSI in 2003 and in ITU (ITU-T Rec. J. 144) for digital cable television systems. Other feature-based VQA metrics are described in [61, 62, 63, 64, 65, 66, 3].

There are two main drawbacks to feature-based VQA. First, they can be highly complex, and thus, not suitable for use in a real-time system. Second, they do a much better job at capturing spatial distortions than temporal ones [3, 60].

Top-Down VQA (SSIM-basd): In 2003, the LIVE! lab at the University of Texas (Austin) proposed a new approach to IQA based on the hypothesis that the HVS is highly adapted to extract structural information from the viewing field [4]. They developed a metric called SSIM, which provides a good measure of the structural information in an image. The metric evaluates three terms: luminance, contrast, and structure, and combines them to form an SSIM-map at each pixel location.

$$l(\mathbf{x},\mathbf{y}) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1} \tag{6.2}$$

$$c(\mathbf{x},\mathbf{y}) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2} \tag{6.3}$$

$$s(\mathbf{x},\mathbf{y}) = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3} \tag{6.4}$$

where $\mathbf{x}$ and $\mathbf{y}$ are the reference and test signals, respectively.

These three quantities are computed at every pixel location using an 11x11 circular-symmetric Gaussian window with standard deviation 1.5 samples, and unity-normalized weighting $w = \{w_i, i = 1, 2, ..., N\}$, with $(\sum_{i=1}^{N} w_i = 1)$.

$$\mu_x = \frac{1}{N}\sum_{i=1}^{N} w_i x_i \tag{6.5}$$

$$\mu_y = \frac{1}{N}\sum_{i=1}^{N} w_i y_i \tag{6.6}$$

$$\sigma_x^2 = \frac{1}{N-1}\sum_{i=1}^{N} w_i (x_i - \mu_x)^2 \tag{6.7}$$

$$\sigma_y^2 = \frac{1}{N-1}\sum_{i=1}^{N} w_i (y_i - \mu_y)^2 \tag{6.8}$$

$$\sigma_{xy} = \frac{1}{N-1}\sum_{i=1}^{N} w_i (x_i - \mu_x)(y_i - \mu_y) \tag{6.9}$$

$$\tag{6.10}$$

In other words, the luminance comparison compares the luminance pixel values at each pixel location (filtered through the Gaussian window); the contrast comparison compares the standard deviation of the luminance values, and the structure comparison compares the mean-subtracted, contrast-normalized signals at each pixel location (though, mean-subtraction and contrast-normalization is equivalently expressed here as the correlation coefficient between x and y).

These quantities are then combined to generate the SSIM map:

$$SSIM(\mathbf{x},\mathbf{y}) = \left[l(\mathbf{x},\mathbf{y})^{\alpha} c(\mathbf{x},\mathbf{y})^{\beta} s(\mathbf{x},\mathbf{y})^{\gamma} \right] \tag{6.11}$$

which is then averaged across the two spatial dimensions to create a single quality score for the image.

SSIM has been widely demonstrated to yield very strong correlation with subjective quality scores [4], and therefore, it has been used as a basis for many other IQA and VQA metrics.

One such modification is based on the observation that quality scores depend on viewing distance. This motivated the development of the Multi-Scale SSIM (MS-SSIM) metric [67], which was shown to offer an improvement over SSIM. The basic idea is resample (by half) and resize (by half) the image several times (usually 5) and average the corresponding SSIM maps for each scale to generate the final score.

Yet another modification to SSIM is based on the hypothesis that humans tend to weigh areas of poor perceptual quality more heavily than they weigh areas of good perceptual quality, thus weighting the scores by their rank ordering may produce better results. This modification, called P-SSIM (pooled SSIM), modifies the final weighting of the SSIM map according to simple percentile weighting, with the weights learned heuristically [68].

A third modification to SSIM is based on the hypothesis that humans gaze or fixate on a few points in an image rather than on the the entire image itself. The authors in [68] use a gaze-prediction algorithm called GAFFE [69] to predict fixation points which may correlate well with human fixations, and they weigh the SSIM map more heavily in these regions. The two questions in this approach are (1) how many fixation points should be used; and (2) what should the weighting be? The authors use heuristic methods to answer these questions, and demonstrate reasonable performance increases (though not as much as with P-SSIM). The authors also combined P-SSIM and F-SSIM to form FP-SSIM which jointly considers rank-ordered percentile weighting and fixation weighting, and found that better agreement with subjective scores were produced. However, they found that the improvement was primarily due to the pooling rather than the fixation weighting - possibly because the fixation finding algorithm had suboptimal performance.

All of the above are IQA metrics, but have been modified to VQA metrics simply

by averaging the spatial SSIM map across every frame in the video sequence. Spatial-Temporal averaging of the SSIM map represents a simple way of transforming the SSIM-based IQA metrics to VQA metrics.

An evolution of this direct averaging is to consider motion models of the HVS to modify the temporal weighting of the SSIM map. This approach, called Speed-Weighted SSIM (SW-SSIM), was proposed in [70]. The authors model the HVS as an information communication channel, develop a modified SSIM weighting function as a function of some combination of the information content and perceptual uncertainty of the signal.

Information Content (I): It is hypothesized that the HVS should pay attention to a surprising event, so if prior knowledge of object speed (anticipation) is know, then the "surprise factor" of an observed visual stimulus can be quantified using a classical Bayesian approach. Indeed, recent work by Stocker and Simoncelli [71] suggests that the prior probability distribution about the speed of motion can be fitted to a straight line in the log-log domain (like a very long tailed Gaussian). This leads to a power-law function for the prior distribution, and consequently, we can estimate from any observed motion, the information content associated with it by computing its self-information (surprisal).

Perceptual Uncertainty (U): It was found that for a given stimulus speed, a log-normal distribution provides a good description of the likelihood function of the noisy measurement [71]. Consequently, a natural way to quantify the level of the internal noise, or the perceptual uncertainty, is the entropy of the likelihood function.

The authors then choose to use a simple subtraction operation to combine I with U to get the weight. This is done at every spatial block and at every time instant $w(x, y, t) = I(x, y, t) - U(x, y, t)$.

A number of modifications are then made to this weighting formulation to correct some practical issues such as contrast response saturation at extreme contrast values, and stabilization and consideration of the Weber-Fechner law when the relative and global motion vectors and local contrast are close to zero.

The absolute motion field is computed using Black and Anadan's multi-layer optical flow estimation algorithm with a five-level pyramid decomposition. This is in lieu of

more computationally intensive block matching-based motion estimation.

The weighting map was then applied to several quality maps (MSE, PSNR, and SSIM). All showed an improvement with respect to the differential MOS (DMOS) with subjective scores from the VQEG Phase 1 database.

Open issues and future work with SW-SSIM include:

- Better ways to combine U and I (besides subtraction)
- Computation of contrast and motion estimation could be improved
- How to calibrate the model parameters

The use of HVS-based motion models in the weighting/pooling process versus in the error quantification process itself is what separates approaches like SW-SSIM from the class of VQA metrics in the next section.

Pure Motion-Based VQA

The models discussed so far are primarily designed to capture spatial distortions, and use simple extensions (primarily implemented through error pooling and weighting) to capture limited characteristics about temporal distortion (e.g through speed estimation). The human eye is very sensitive to motion and can accurately judge the velocity and direction of motion of objects in a scene [3, 60]. As previously discussed, two types of temporal filters are often used to augment IQA metrics into VQA metrics: bandpass and lowpass, which model the sustained and transient neurophysiological functions of neurons in the front end of the HVS (this includes the retina, lateral geniculte nucleus (LGN), and Area V1 of the visual cortex [72]). However, Area MT/V5 of the extrastriate cortex is known to play an important role in human perception of motion. While the neurons in Area V1 act more or less as linear filters [60], Area MT/V5 is known to play a role in the *perception* of motion [60, 3].

With this in mind, the authors of [60] have developed a VQA index called MOtion based Video Integrity Evaluation (MOVIE) which measures both spatial and temporal video distortions over multiple scales, and along motion trajectories, while accounting

Table 6.1: Comparison of several VQA metrics using the VQEG FRTV Phase 1 Database. Results were obtained from [60].

VQA Algorithm	SROCC	LCC	OR
PSNR	0.786	0.779	0.678
Proponent P8 (Swisscom)	0.803	0.827	0.578
SW-SSIM	0.812	0.849	0.578
MOVIE	0.833	0.821	0.644

for spatial and temporal perceptual masking effects. They evaluate the MOVIE index using the VQEG FRTV Phase 1 database, and have found that MOVIE delivers VQA scores that correlate quite closely with human subjective judgment.

Performance Analysis of State-of-the-Art VQA Metrics

A sampling of some popular VQA metrics was evaluated in terms of accuracy [60, 3] using the VQEG FRTV Phase 1 Database on the basis of the Spearman Rank-Ordered Correlation Coefficient (SROCC), Linear Correlation Coefficient (LCC) after nonlinear regression, and Outlier Ratio (OR), which is the percentage of the number of predictions outside the range of +/- 2 times the standard deviation. The LCC measures prediction accuracy, the SROCC measures prediction monotonicity, and the OR measures prediction consistency. The results are shown in Table 6.1. Note that Proponent P8 (Swisscomm) was the best performing model tested by VQEG (MOVIE and SW-SSIM were not part of VQEG's evaluation, as they had not yet been invented) [73].

Another evaluation of selected VQA metrics was recently performed [74]. This time, the metrics were compared with respect to both accuracy and complexity (though, it is not clear how the authors defined complexity). The results are shown in Table 6.2 and Figure 6.3. This evaluation is quite interesting for our purposes because it used the LIVE! Wireless Video Database, which tests distortions due to packet loss and H.264 compression in a wireless environment (versus tv-related distortions used by VQEG in the previously mentioned study). Four packet loss rates were considered (0.5%, 2%, 5%, and 17%), and four compression levels (0.5 Mbps, 1 Mbps, 1.5 Mbps, and 2 Mbps). Results are reported in terms of SROCC, LCC, and RMSE. The metrics tested

Table 6.2: Aggregate comparison of several VQA metrics using the LIVE! Wireless Video Database, across all distortion types and compression levels. Results were obtained from [74].

VQA Algorithm	SROCC	LCC	RMSE
PSNR	0.8615	0.8639	8.8997
Frame-SS-SSIM	0.8967	0.8875	8.1448
Frame-MS-SSIM	0.9608	0.9588	5.0196
VQM	0.9721	0.9711	4.2172
SW-SSIM	0.9599	0.9617	4.8450
P-SS-SSIM	0.9628	0.9637	4.7180

were PSNR, Frame-SS-SSIM (averaging of the single-scale SSIM map across frames), Frame-MS-SSIM (same as previous, but using multi-scale SSIM), VQM, SW-SSIM, and P-SS-SSIM (single-scale SSIM with weighted percentile ranking, averaged across all frames).

The evolution of pure motion-based VQA is promising. However, this evolution is still in its infancy (the only known VQM to use this approach is MOVIE), and we must consider the tradeoff between complexity and accuracy. Based on the results shown above, we believe that MOVIE does not offer enough of an accuracy advantage to justify its increased complexity in comparison to the modified pooling strategies of SW-SSIM and P-SS-SSIM, for example. Therefore, we are currently investigating three approaches in order to develop a VQA metric that's suitable for use in a real-time cross-layer video streaming system:

- Further modification of error pooling techniques based on HVS principles related to temporal perception (e.g. extending approaches like P-SS-SSIM and SW-SSIM).
- Investigating methods to reduce the complexity of pure motion-based VQA, like MOVIE.
- Consider using an existing metric without modification. The most likely candidate would be P-SSIM based on its attractive balance of accuracy and complexity.

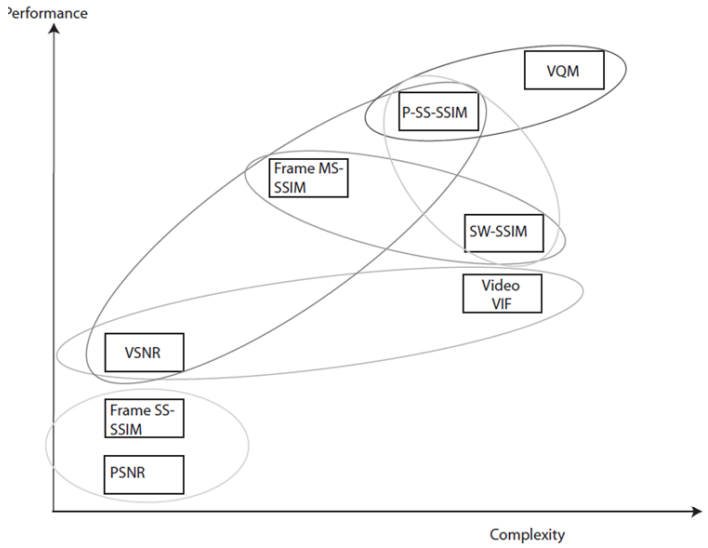


Figure 6.3: Tradeoff between complexity and accuracy. Circles indicate metrics that were statistically indistinguishable. Figure obtained from [74].

6.2.2 VQA with Non-Human Observers

Although a great deal of work has been done on perceptually oriented video quality assessment, not much is known about how well these metrics can predict the performance of applications involving non-human observers (e.g. such as video processing and analytics algorithms). As video processing techniques such as object tracking become more widespread, it is increasingly important to understand the relationship between perceptual VQA and task-based VQA with non-human observers. This topic is explored in the next section.

6.3 Task-Dependent VQA with Non-Human Observers

The proliferation of wireless technology, mobile computing, and increasingly sophisticated video codecs have fueled the sharp increase in demand for applications involving wireless video streaming. Among these applications, there is a huge demand for video analysis and computer vision techniques (such as object tracking). Wireless video streaming systems typically have limited resources, and introduce distortions into the video signals (e.g. due to packet loss and large quantization parameters). Unfortunately, the impact of this low-quality video data on the performance of task-dependent non-human computer vision applications like object tracking is still an under-explored field. The existing video quality assessment research mainly focuses on how humans perceive video quality, and proposes metrics based on the neurophysiology and psychophysics of the Human Visual System (HVS). Evaluating the suitability of a video signal for specific non-human tasks such as object tracking is of tremendous importance in order to overcome the challenges associated with video analysis on distorted video signals. First, such metrics may be used in the design, optimization, and dynamic tuning of parameters of the tracking algorithm. Second, these metrics may be used to drive the adaptive allocation of resources in both the video codec and the wireless video transmission system. Finally, these metrics could be used to benchmark the performance of competing object tracking algorithms.

In this section, we evaluate the performance of object tracking algorithms under

various types of video distortions typically associated with wireless video transmission, and investigate how well today's perceptually-oriented video quality metrics can predict this performance. We use the results of this study to propose guidelines for designing new video quality metrics that can accurately assess the suitability of a distorted video signal for processing by an object tracking algorithm (note that designing new video quality metrics is beyond the scope of this dissertation; this chapter focuses on the concepts associated with making intelligent use of video quality metrics in the context of an adaptive real-time cross-layer wireless video streaming system). The analysis is based on experimentation with a state-of-the-art mean-shift tracking algorithm [75], and with a video database which we have engineered to capture the distortions introduced by present generation encoders and wireless channels.

6.3.1 Object Tracking Algorithm Selection

In our study, we use a variant of the popular mean-shift tracking approach first proposed in [75]. The algorithm has several merits: (1) in the last decade, it has been tested, tweaked, and studied by many researchers in the computer vision field; (2) it has been successfully used in commercial-grade products for object tracking; and (3) the authors who first proposed this algorithm earned the 2010 Longuet-Higgins Prize. For the sake of brevity, we shall forgo a detailed discussion of mean-shift tracking, and refer the reader to the original paper [75] for such details.

6.3.2 Engineering the Video Database

This section explains how we engineered a video database suitable for this study. Four unique video sequences (each 160 frames long, 720x480 resolution, 30 frames per second) form the seeds for the video database. We generated three video sequences from each of the four seeds by selecting three different tracking zones from each seed (actually, two of the three zones are geometrically identical, but for one of them, we enable spatial kernels [76] in the mean-shift tracker). Then, for each of these 12 video sequences, we used the H.264 JM Software [77] to encode each sequence at 15 different bitrates (200 kbps - 1500 kbps, and 2000 kbps). After that we applied seven different levels of packet

loss (obtained from realistic IEEE 802.11 WLAN packet loss patterns generated by a validated ns2 simulator [21]) to each of these 180 video sequences, thus yielding a total of $4 \cdot 3 \cdot 15 \cdot 7 = 1260$ video sequences in our database.

Further details regarding the design methodology are given below:

- Video Selection: The selected database seeds were chosen because they collectively represent a blend in terms of the amount of occlusion, motion, luminance contrast, and chrominance contrast between the tracking target and the background.

- Tracking Region Selection: We selected three different tracking zones: "upper body", "full body", and "full body plus spatial kernels" in order to average-out any dependence on the exact placement of the tracking zone. Additionally, we offset the first "learning frame" from the first I-Frame to the first P-Frame since the initial IDR is overly quantized due to the rate control algorithm used in JM 16 (and most other H.264 implementations).

- Encoding Parameters: The H.264 JM encoder was configured with the settings listed in Table 6.3. These settings were chosen in order to: (1) yield an MTU size that's realistic for transmission over a wireless LAN; and (2) represent commonly used slicing strategies that offer robustness to packet loss, while simultaneously ensuring an appropriate MTU size.

- Decoder Parameters: All decoder parameters were the JM 16 defaults, and the error concealment mode was set to "Motion Copy" (see the discussion later in this chapter for an overview of error concealment in H.264 AVC).

- Distortion Sources: (1) Quantization: Fifteen different bitrates were used (200 kbps - 1500 kbps, plus 2000 kbps) (2) Packet Loss: Seven different packet loss rates were applied (0%, 0.5%, 2%, 5%, 9%, 13%, 17%)

- Tracking Algorithm Parameters: Number of RGB channel bits used = 5

It is our hope to eventually make this database available to the research community so that further studies on task-dependent video quality assessment with non-human observers may be carried out.

Table 6.3: H.264 AVC JM Encoder Parameters

Parameter	**Value**
ProfileIDC	baseline
IntraPeriod	30
IDRPeriod	0
SliceMode	fixed number of MBs in slice
SliceArgument (number of MBs in a slice)	225
NumberOfSliceGroups	3
SliceGroupMapType	Dispersed
RateControl	enabled
RateControlType	original JM RC
RCMinQPPSlice	0
RCMaxQPPSlice	44
RCMinQPISlice	0
RCMaxQPISlice	36
Resolution	720x480
Framerate	30 fps
Duration	160 frames

6.3.3 Results and Analysis

We executed the tracking algorithm on all 1,260 video sequences in the database, and compared using various statistical methods the response of the following video quality metrics with the output of the tracking performance metrics (also shown below):

- Video Quality Metrics
  - Frame-SS-SSIM [4]
  - Frame-MS-SSIM [67]
  - MSE
  - PSNR
- Tracking Performance Metrics
  - Probability of correct tracking
  - MSE of tracking bounding box
  - Number of incorrectly tracked frames

Figures 6.4(a-b) show an example which compares how the video MSE and tracking bounding box MSE change in response to varying quantization levels and packet loss rates across all 1,260 sequences in the database. The contour plot clearly shows that the MSE of the video signal itself is not an accurate predictor (or indicator) of tracking performance. Figures 6.4(c-d) show a similar example, comparing the probability of

correct tracking and Frame-SS-SSIM metrics, and lead to a similar conclusion. Figures 6.4(e-f) show the video MS-SSIM and PSNR, respectively, which can also be contrasted with the dynamics of the bounding box MSE and probability of correct tracking plots.

This analysis ultimately leads to guidelines for how to modify existing perceptual quality metrics so that they can more accurately predict the performance of object tracking algorithms under the distortions typically introduced by wireless video transmission. Many of these guidelines generalize to other means of communicating the video signal to the tracking algorithm (e.g. compressed videos stored on a hard disk, transmission over Ethernet, etc.). Among the most important guidelines to consider are:

- Error sensitivities in mean-shift tracking
  - high-frequency video signals near the tracking zone
  - occlusions (duration, patterns, and quantity)
  - target's luminance and chrominance contrast
  - velocity of the object being tracked
- Impact of different packet loss patterns on tracking performance (e.g. packet loss due to collisions vs. channel errors)
- Impact of ROI-coding and redundant slices near the tracking zone
- Use of color channels in video quality assessment
- Impact of slicing and packetization strategies on tracking performance
- Impact of spatial kernels on tracking performance

6.4 Estimating an End-User's Received Video Signal at the Transmitter Using Cross-Layer Feedback

Once we have a VQA metric, we need to figure out how to compute it in the context of an adaptive real-time video streaming system. In these systems, the most powerful

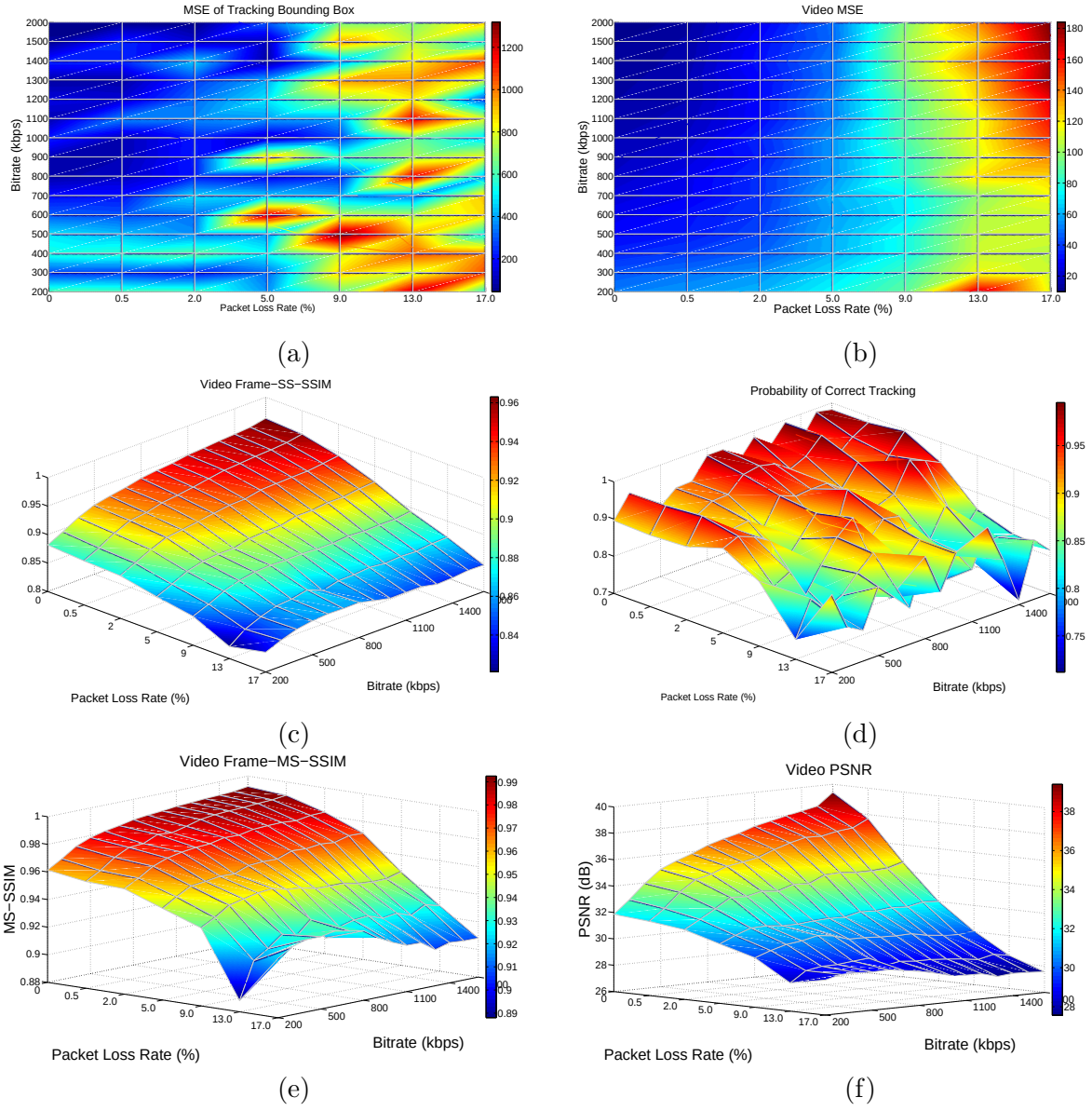


Figure 6.4: How well do current video quality metrics predict tracking performance? Results averaged across 1,260 distorted video sequences indicate that current video quality metrics must be modified if they are to accurately predict the performance of task-specific video processing applications with non-human observers. (a-b) Video MSE vs. MSE of the tracking zone's bounding box; (c-d) Video SSIM (a well-known perceptual quality metric) vs. Probability of Correct Tracking; (e) Video MS-SSIM; (f) Video PSNR

adaptation potential is offered at the encoder/transmitter (due to both the rich space of error resilience features available in the encoder (see [2]), and the adaptability of the wireless transmitter's PHY and MAC), and therefore, it makes sense to put the adaptation logic (which includes the VQA metric as the underlying decision statistic) there. Unfortunately, the encoder does not have direct access to the received video stream. This is a serious obstacle because it precludes the use of a FR metric, and suggests that either a RR or NR metric is needed. Since FR metrics are obviously much more desirable than RR or NR metrics (c.f. the discussion in the previous section), we have investigated and developed a way to overcome this obstacle. This corresponds to steps 3 and 4 in Figure 6.1.

We have developed a simple method for estimating the received video stream at the encoder by using cross-layer feedback from the wireless MAC in single-hop uplink wireless video topologies like the one shown in Figure 6.5. The idea is based on the observation that distortions between the original reference signal and the received (decoded) signal are introduced in three stages (encoding, transmission over a lossy channel, and decoding), and if we can account for all of these distortion types at the encoder, then we can obtain an accurate reconstruction of the received video signal at the encoder. We show that:

- encoding distortions (for example, due to quantization) are already available at the encoder because both the original pristine reference signal and the distorted (encoded) test signal are available there,
- transmission distortions can be made known to the encoder using cross-layer feedback from the wireless MAC, and
- decoding distortion (for example, due to error concealment) can be made available at the encoder by locally running a copy of the receiver's decoder.

Figures 6.6a, 6.6b, and 6.6c illustrate the overall proposed changes to the H.264 Network Abstraction Layer (NAL), Video Coding Layer (VCL), and wireless MAC, respectively, and we will frequently refer to these figures as we discuss how to capture each of the three distortion types at the encoder.

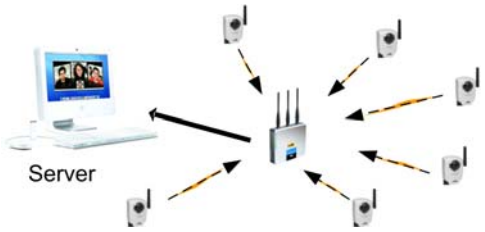


Figure 6.5: Uplink wireless video transmission topology. Videos are streamed from wireless cameras to the server through an access point.

6.4.1 Encoding Distortion

Although the primary source of encoding distortion in modern video encoders like H.264 AVC is quantization of the DCT coefficients, other parameter settings in the encoder have a tremendous impact on how distortions introduced in any stage (i.e. encoding, transmission, decoding) visibly manifest themselves to the end-user. Some of these parameters include:

- slicing and packetization strategies (slice mode, slice argument, number of slice groups, slice group map type, and number of slices per RTP packet)
- intra-frame period
- IDR period
- rate-distortion optimization
- allowable range of quantization parameters for I and P slices

Thus, in order to accurately estimate the end-user's video signal, we must account for all aspects of the encoding process, not just distortions introduced due to quantization. Fortunately, it is easy to capture this information at the encoder since both the original (pristine) "reference" signal and the distorted (encoded and packetized) "test" signal are already available there. The primary task at this point is to keep track of each of these signals in separate buffers so we can periodically access them. The H.264 Video Coding Layer (VCL) is responsible for encoding pristine reference frames into H.264 bitstreams, and consequently, as shown in Figure 6.6b, we add a buffer called $referenceBuffer$ to the VCL in order to store the incoming reference frames. After

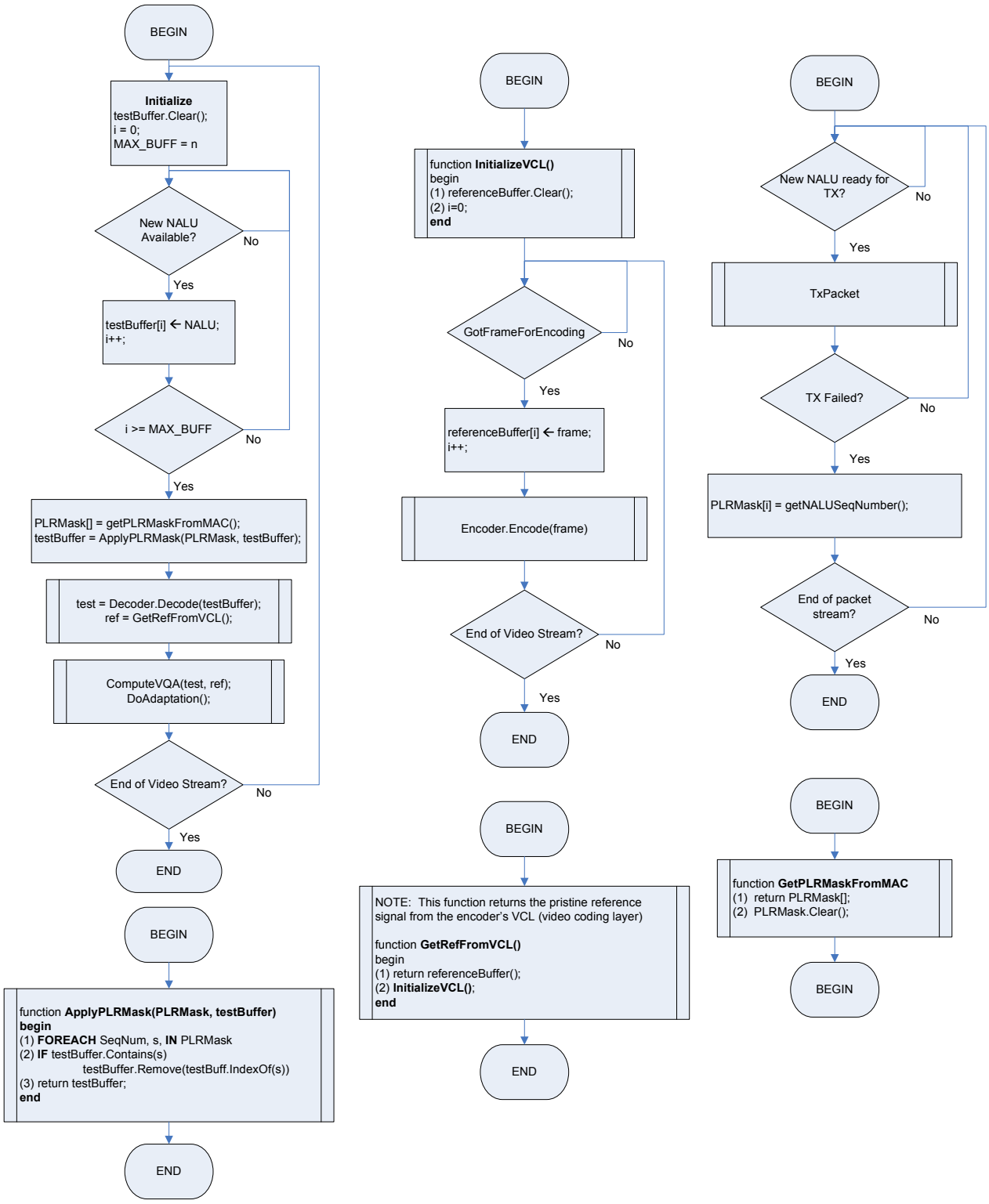


Figure 6.6: (a) Modified code in the H.264 Network Abstraction Layer (NAL) for reconstructing the received video signal using cross-layer feedback from the MAC regarding packet loss; (b) Modified code in the H.264 Video Coding Layer (VCL) for accessing the pristine reference signal; (c) Modified wireless MAC code for keeping track of the sequence numbers of lost video packets (NALUs). The algorithm was designed with the IEEE 802.11 MAC in mind, but it could work in any wireless technology that supports MAC-level acknowledgments.

the VCL encoding process, H.264 bitstreams are sent to the Network Abstraction Layer. The H.264 NAL specifies the mapping from the H.264 VCL to transport layers, such as [2]:

- RTP/IP for real-time wire-line and wireless conversational or streaming services
- File formats (e.g. ISO MP4 for storage and MMS)
- H.32X for wire-line and wireless conversational services
- MPEG-2 systems for broadcasting services, etc.

In this dissertation, we're interested in the first case (where the NAL transforms the VCL bitstream into an RTP packet stream). These RTP packet streams contain all the information about the impact of the encoding process on the pristine reference signal, and therefore, as shown in Figure 6.6a, we add a buffer called $testBuffer$ to the NAL in order to store these packetized encoded frames.

6.4.2 Transmission Distortion

Distortions introduced by transmission are associated with packet loss and jitter. For the sake of simplicity of discussion, we assume for now that the receiver has a sufficiently large jitter buffer to guarantee that packets are never lost due to jitter. This assumption is realistic in the uplink WLAN topologies considered in this dissertation since (1) there is only one hop in the network; and (2) our approach is likely to be used inside a resource allocation algorithm, so assuming the algorithm works reasonably well, resources will be controlled such that jitter is not a factor.

Packet loss may be due to collisions, channel errors, or interface queue (IFQ) overflow (e.g. from excessive congestion), and we must consider all of them. Fortunately, the design of the H.264 NAL and the presence of MAC-level acknowledgments in wireless local area systems such as IEEE 802.11 make it easy to design MAC-level inspection modules which identify the RTP sequence numbers of lost packets. In particular, we can construct a MAC-level module that (1) tracks packet loss; and (2) precisely identifies

which NALUs in the transmitted H.264 packet stream were associated with those packet losses.

In H.264 NAL's RTP/IP mode, each packet can carry a NALU fragment, a single NALU, or several NALUs (aggregation mode). In any case, each NALU header contains an RTP sequence number that uniquely identifies its position in the H.264 packet stream ([13]). At the same time, the MAC-level inspection module can track in real-time which NALUs have failed to reach the transmitter. In the case of IFQ overflow, this simply involves logging the RTP sequence number of the dropped packet. In the case of collisions or channel errors, this involves tracking MAC-level acknowledgments and the MAC retry limit to determine if the NALU was lost, and then logging the RTP sequence number of the lost NALU.

Therefore, we can build a MAC-level inspection module that records this information in real-time, and periodically reports it to the encoder. Since the encoder tracks the original NALU RTP packet stream in the $testBuffer$, it can compare the sequence numbers of the lost NALs (reported by the MAC) with the sequence numbers of the NALUs in the $testBuffer$, and simply remove the lost NALUs from the $testBuffer$. This so-called "'packet loss overlay"' step is represented by the block "'PLRMask[]=getPLRMaskFromMAC()"' in Figure 6.6a, and the resulting buffer represents a new encoded NALU RTP stream which accounts for all of the distortions introduced by encoding and transmission.

6.4.3 Decoding Distortion

The distortions introduced in the decoding process are primarily due to error concealment, which we now briefly review. Although error concealment is not normative in H.264, the following approach is the one that's used in most cases.

Before errors can be concealed, they must first be detected. The decoder maintains a map of macroblocks (MBs) for the current frame number, n_c. This map simply indicates whether or not each MB in the frame was correctly received. There is an additional mapping, which indicates which slice each MB belongs to. If the decoder receives a slice (for simplicity of discussion, assume one slice per packet, and one packet

per NALU) with a header that indicates it belongs to frame number $n_c + 1$, then the decoder first assumes that all slices of the current frame have been correctly received, and it should move on to the next frame. To verify this assumption, it looks at the MB map for n_c, and checks that all entries in its MB map are marked as correctly received. If they are, then it moves on to the next frame, as originally assumed. Otherwise, a slice loss is inferred (meaning at least one slice was lost from the current frame), and the MBs are concealed using one of the techniques we shall soon explain. If the decoder receives a packet with a frame number $> n_c + 1$, then it assumes a loss of pictures (e.g. all slices belonging to the preceding one or more pictures were lost), and inserts concealed pictures (using previous frame copy) into the reference picture buffer.

Error concealment is performed differently for intra (I) and inter (P) frames. For I-Frames, a weighted averaging approach is utilized. For an incorrectly received MB, the decoder first determines how many adjacent neighboring MBs have been correctly received. If there are at least two, then only these two MBs are used in the concealing process. Otherwise, previously concealed neighboring MBs may be used. For each pixel in the MB to be concealed, the decoder finds the closest boundary pixels to that pixel from each of the neighboring MBs selected for the concealment process. The pixel value is assigned as a weighted sum of the boundary pixels. The weight is determined relative to the inverse distance between the pixel being concealed and the selected boundary pixels.

For P-Frames, a technique commonly referred to as "'motion copy"' is applied. First, the decoder examines the motion vectors of the correctly received slices of that picture. If the average motion vector length is less than 1/4 pixel long, then the decoder simply copies co-located pixels from the reference frame. Otherwise, motion-compensated error concealment is applied, as we now describe. Each MB is considered in a special order starting from the left and right columns, and moving inward toward the center, and moving top to bottom down those columns. When an erroneous MB is encountered, the decoder first checks if any neighboring MBs were correctly received. If so, then their motion vectors and reference frames are used to form a candidate for the MB to be concealed. If the motion vectors in all of the neighboring MBs are lost,

then any previously concealed neighboring motion vectors are considered as candidates. Additionally, the spatially co-located block from the previous frame is always one of the candidates. Then, for all of the candidates, a boundary matching error is calculated (defined as the sum of the pixel-wise absolute differences of the adjacent luminance pixels in the candidate block and its decoded or concealed neighbor blocks). The candidate with the lowest boundary matching error is chosen, and therefore, its motion vectors and associated reference picture are used to generate the concealed MB.

Our goal is to enable the encoder to account for these decoding distortions. If the transmitting station is somehow made aware of the receiver's decoder configuration (including error concealment and detection strategies), then we can locally emulate the decoder locally at the transmitting station. Since it is rare for a decoder to change its configuration (it might never change), it is reasonable to exchange this information with the encoder a-priori. Then, we can account for decoding distortions by passing the "'PLR-masked"' NALU RTP stream through the locally running decoder. This is represented in Figure 6.6a by $test = Decoder.Decode(testBuffer)$.

The concept of local decoding at the encoder is not entirely new. In [9, 10], the authors perform multiple simultaneous instances of local encoding and decoding using different parameter settings in the encoder, and then evaluate the distortion (expressed as MSE and PSNR) between the video signals under those various parameter settings. However, the concept of capturing transmission distortion using cross-layer feedback from the MAC, and the more general concept of reconstructing the received video signal at the encoder is, to the best of our knowledge, new.

6.4.4 Putting Everything Together

The output of the locally running decoder in the previous step represents the decoded "'PLR-masked"' NALU RTP packet stream, just as the actual end-user receives it, thus completing our goal of obtaining an accurate estimate of the received video signal locally at the encoder. This reconstructed test signal and the pristine reference signal (obtained in Figure 6.6a during the step $ref = GetRefFromVCL()$, and in Figure 6.6b by the function expansion labeled $GetRefFromVCL()$) could then be used as inputs to any

full-reference perceptual video quality metric that's capable of running in real-time on the transmitting station (as shown in Figure 6.6a by the step $ComputeVQA(test, ref)$). A resource allocation algorithm could then use that metric as the basis for its cost function, thus enabling perceptually-oriented allocation of wireless and video coding resources. This is represented in Figure 6.6a as $DoAdaptation()$.

In the context of a resource allocation algorithm, the length of the $referenceBuffer$ and $testBuffer$ determine how frequently the algorithm wishes to perform the adaptation step. This is represented in Figure 6.6a as the decision step, $i >= MAX\_BUFF$. There is a great deal of freedom in choosing the buffer length. One way to choose the buffer length is according to time (e.g. perform adaptation every 1 second). A second way to choose the buffer length is by absolute number of frames (e.g. perform adaptation every n frames). A third way to choose the buffer length is according to the intra period (I:P frame ratio) (e.g. perform adaptation on every GOP). For simplicity of discussion, our figures are based on the second approach.

6.4.5 Performance Evaluation

We have developed a robust simulation environment for testing and evaluating this approach using the H.264 JM reference software [77]; a packet-loss overlay tool for RTP packet streams; a RTP NALU packet sniffer; a validated ns2 model of IEEE 802.11 [21] for generating realistic packet loss patterns; and a database of 420 distorted H.264 AVC compressed natural video sequences.

Four unique video sequences (each 160 frames long, 720x480 resolution, 30 frames per second) form the seeds for the video database. Then, for each of these four video sequences, we used the H.264 JM Software [77] to encode each sequence at 15 different bitrates (200 kbps - 1500 kbps, and 2000 kbps). The H.264 JM encoder was configured with the following JM encoder settings: ProfileIDC=baseline; IntraPeriod=30; IDR-Period=0; SliceMode=fixed number of MBs in slice; SliceArgument (number of MBs in a slice)=225; NumberOfSliceGroups=3; SliceGroupMapType=Dispersed; RateControl=enabled; RateControlType=original JM RC; RCMinQPPSlice=0; RCMaxQPP-Slice=44;RCMinQPISlice=0; RCMaxQPISlice=36. These settings were chosen in order

to:

- Yield a MTU size that's realistic for transmission over a wireless LAN
- Represent commonly used slicing strategies that offer robustness to packet loss, while simultaneously ensuring an appropriate MTU size.

After that, we used the RTP packet-loss overlay tool to apply seven different levels of packet loss (obtained from realistic IEEE 802.11 WLAN packet loss patterns generated by our ns2 simulator [21] at seven different bit-error-rate (BER) conditions) to each of these 180 video sequences, thus simulating the wireless transmission of each of the $4 \cdot 15 \cdot 7 = 420$ video sequences in our database. The BERs yielded the following packet loss rates: 0.0, 0.5, 2.0, 5.0, 9.0, 13.0, 17.0. Next, we decoded each of the 420 video sequences using the H.264 JM decoder, thus representing the actual video signals received by the end-user. The H.264 JM decoder was configured to use the error detection and concealment strategy explained in section 6.4.3.

To test our approach, we applied our MAC-level inspection module to the ns2 traces of the transmitting station, and overlaid the packet-loss patterns on the $referenceBuffer$, as explained earlier. We then passed the PLR-masked $testBuffer$ through a decoder which was configured identically to the end-user's decoder. We compared this signal to the signal obtained by the end-user in order to gage the accuracy of our approach. The signals are compared using the Mean Squared Error (MSE) and the popular MS-SSIM metric [67] as shown in Table 6.4. The results are averaged for every group of 30 frames (which happens to represent the GOP size), averaged across all GOPs in all 420 video sequences. The results indicate nearly perfect reconstruction of the received video signal at the encoder.

The primary source of error in our algorithm is the sensitivity to the loss of MAC-level acknowledgments (ACKs). In 802.11 uplink topologies like the one's we're considering, ACKs may be lost due to a variety of reasons, but most commonly, they are lost due to: (1) collisions; or (2) channel errors. It should be noted that our algorithm is intended to be used within resource allocation algorithms, and therefore, since such algorithms are generally concerned with the intelligent management of wireless and

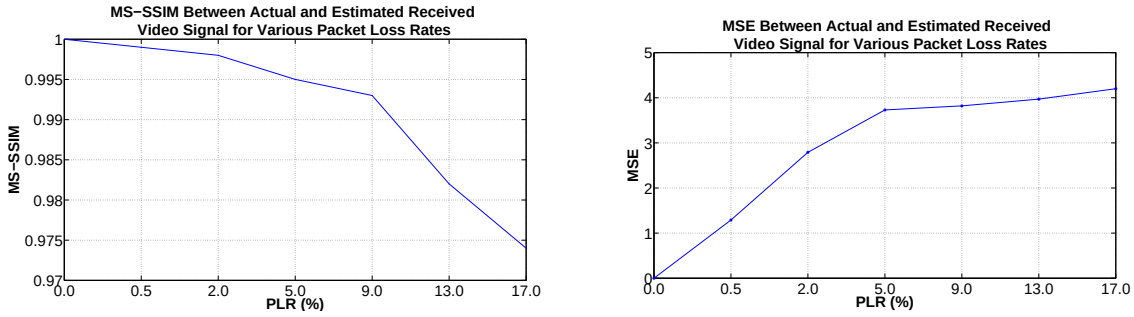
Table 6.4: Average and standard deviation of MSE and MS-SSIM comparisons between the actual received video signal and our estimate of it.

	MSE	**MS-SSIM**
AVERAGE	2.8286	0.9916
STDEV	1.5994	0.0098

video coding resources in order to ensure optimal system performance, it is not at all likely that ACKs will be lost due to congestion.

If an ACK is lost, our algorithm incorrectly assumes that the packet in question was not correctly received by the end-user. Since our encoder is configured to send one slice per packet (as is commonly the case), this means that the loss of an ACK infers the loss of a slice in the current frame. Since our encoder settings also prescribe that each frame contains six slices (3 slice groups, each with 2 slices (due to the fact that we fixed each slice at 225 macroblocks)), then the loss of a single ACK implies the loss of one out of the six slices in the frame (it should also be noted that in our system, it is not at all likely that two consecutive ACKs will be lost). The decoder's error concealment mechanism does a relatively good job at concealing the loss of a single slice under our configuration. This is primarily due to the fact that our encoder uses a dispersed slice group map, which is conducive to finding suitable neighboring MBs to conceal the lost MBs, using the technique described earlier.

The likelihood of loosing an ACK increases with the BER (and similarly, with the PLR). Figure 6.7 breaks-down the performance of our algorithm at each of the seven PLRs, averaged across all videos in the database at each PLR. As before, we use the MSE and MS-SSIM metrics. The results show that performance "'drops"' slightly as the PLR increases, but keeping in mind the scale of the y-axis, these lines are relatively flat. In fact, it is well-known that the range of MSEs and MS-SSIMs shown in these figures correspond to visually indistinguishable errors. The results indicate that our algorithm is robust up to moderate PLRs of 17 percent. Extrapolating the results indicates that the approach is also sound up to significantly higher packet loss rates.



(a) (b)

Figure 6.7: Difference between the actual and estimated video signals at various packet loss rates, expressed as (a) MS-SSIM; and (b) MSE.

6.5 Predicting Video Quality at Adjacent Bitrates and Packet Loss Rates

Since the VQA metric is the adaptive system's underlying decision statistic, we must not only be able to compute the VQA metric under the current instantiation of the parameter space, but also at other instantiations of the parameter space (as shown in step 6 in Figure 6.1). We will focus on two specific changes to the parameter space: (1) the packet loss rate; and (2) the bitrate/quantization-parameter. We chose to focus on these two particular items because they are related to the airtime-fair distributed cross-layer congestion control strategy we discussed in Chapter 4, and to the perceptual-quality based link adaptation strategy we proposed in Chapter 5.

6.5.1 Predicting the VQA Index at Other Packet Loss Rates

In the last section, we described a method for reconstructing the received video signal at the encoder, accounting for distortions due to encoding, transmission, and decoding, thus allowing any FR VQA metric to be used. We would now like to extend this approach to predict the value of the received video signal (and ultimately, the VQA metric), if there were to be some anticipated (not observed) change in the packet loss rate. Our proposed solution is quite simple: If the anticipated packet loss rate is higher relative to the current PLR, then randomly remove additional packets (using a uniform distribution) from the current PLR-masked $testBuffer$ until the new target PLR is reached. If the anticipated packet loss rate is lower than the current PLR,

then randomly add packets (say, from a copy of the un-masked $referenceBuffer$) to the current PLR-masked frame buffer until the new target PLR is reached. As an alternative to this blind estimation approach (i.e. using a uniform distribution), one could consider dynamically learning the probability model of recent packet loss in the network, and randomly add/remove packets from the PLR-mask according to that distribution.

6.5.2 Predicting the VQA Index at Other Bitrates

We would now like to extend this approach to predict the value of the received video signal (and ultimately, the VQA metric), if there were to be some anticipated (not observed) change in the bitrate (quantization parameter). One way to capture the encoding distortion due to quantization is to run an additional simultaneous encoder at that quantization level. There are two problems with this approach: (1) the new bitrate of interest is not known during the encoding process of the $referenceBuffer$ (i.e. in Figure 6.1, encoding is performed in step 2, but alternative parameter instantiations might not be known until step 5; and (2) we would like to avoid duplicating the complete encoding cycle since it is resource intensive. A more reasonable approach is to intervene in the middle of the encoding process at the point right before the DCT coefficients are quantized, and store the unquantized DCT coefficients in a separate buffer. When the $referenceBuffer$ is full, we begin the adaptation step as described earlier, the new bitrate of interest becomes available, and we quantize the buffer of DCT coefficients according to that bitrate. We then perform the remaining encoding steps (e.g. entropy coding, NALU packaging, etc.) on that bitstream. The result is a NALU packet stream which completely captures the encoding distortion at that new bitrate. This approach avoids duplicating the computationally costly parts of encoding such as motion compensation and the discrete cosine transform. If the new parameter space of interest is known prior to the adaptation step, and if there are enough resources such that multiple encoding process are easily supported, then the first approach of using simultaneous encoding could be used. Otherwise, we suggest the approach just proposed.

Chapter 7

Joint Link Adaptation and Congestion Control Driven by User/Task-Centric Resource Allocation

7.1 Introduction

In this chapter, we propose a resource control strategy called TCCLA (Task-Centric Congestion Control and Link Adaptation) which utilizes a synergy between the material in Chapters 4-6 in order to simultaneously perform link adaptation and congestion control according to a task-oriented objective function.

The CLLA algorithm presented in Chapter 5 forms the basis for the resource control strategy proposed in this chapter. The first extension to CLLA is to allocate resources based on user-centric and task-centric criteria, rather than MSE. The second extension to CLLA is to consider scenarios where the client performs bitrate adaptation (for example, in response to a change in the PHY rate). By leveraging the CLC algorithm proposed in Chapter 4, we can not only consider the case where bitrate adaptation is performed, but the case where bitrate adaptation is performed in such a way that congestion is controlled via an airtime fairness policy. The end result is a joint link adaptation and congestion control strategy driven by task-centric resource allocation. We support the proposed algorithm through simulations and experiments with real wireless cameras on which we have implemented our algorithm.

7.2 System Overview

The TCCLA system is shown in Figure 7.1 for a single camera. Since the system is distributed, each camera has its own instantiation. All instantiations share the number N of active cameras in the BSS as well as the receiver's decoder configuration. Both of

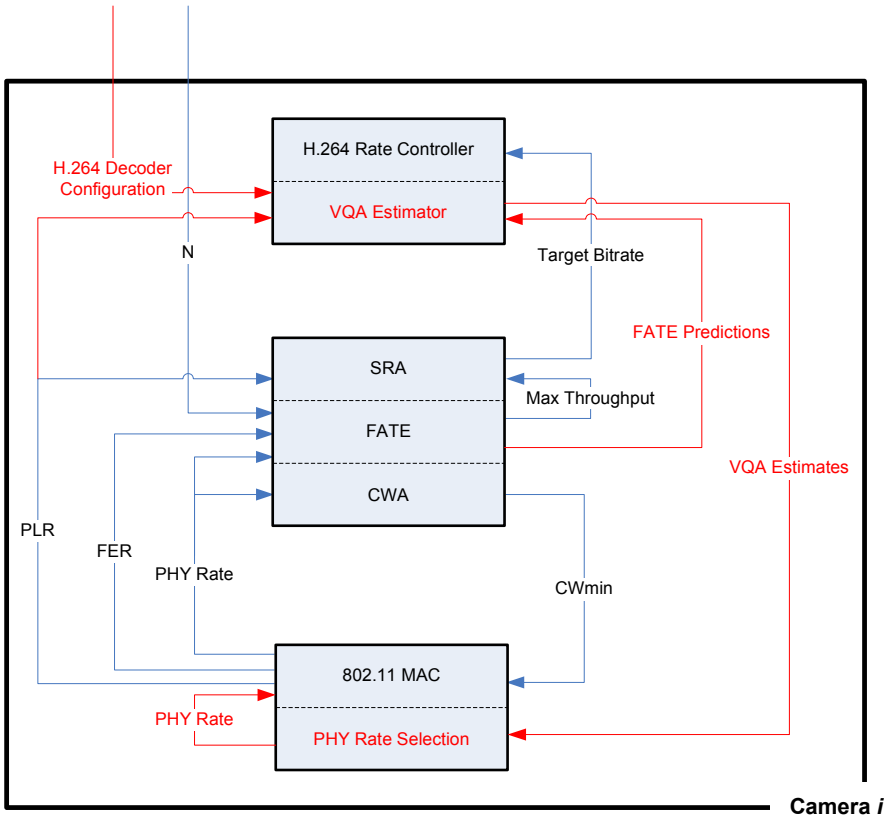


Figure 7.1: Systems level diagram of TCCLA shows that it consists of a synergy between the cross-layer link adaptation strategy presented in Chapter 5, the cross-layer congestion control algorithm presented in Chapter 4, and the video quality assessment strategy presented in Chapter 6.

these parameters may change dynamically.

As mentioned above, the TCCLA system is a synergy of the congestion control, link adaptation, and video quality assessment strategies proposed in Chapters 4, 5, and 6, respectively. Indeed, comparing Figure 7.1 with Figure 4.3 from Chapter 4, we can immediately recognize all of the CLC components. The main additions are the two blocks highlighted in red: (1) VQA Estimator; and (2) PHY Rate Selection.

Also, recall from Chapter 4 that CLC is a passive (reactive) algorithm (i.e. it reacts to changes in network performance (although it implicitly influences network performance)). In contrast, CLLA (Chapter 5) is a proactive algorithm (i.e. it actively

1: **repeat**
2: **CLLA** measures the current PLR, p_e, and p_c (as described in Chapter 5) and passes this information to the VQA Estimator module
3: **VQA Estimator** estimates the received video quality under the current parameter configuration (as described in Chapter 6) and reports this value back to CLLA. Note, the VQA Estimator can compute any full-reference video quality metric, as needed by the application/task in question (i.e. it is not tied to any specific metric like SSIM).
4: **FATE** enters a special prediction mode and predicts how the CLC algorithm will react (i.e. what bitrate will be selected) at the adjacent lower and higher PHY rates. The predicted bitrates are then sent to the VQA Estimator. The prediction is performed by leveraging the FATE equation (see Chapter 4) at the adjacent higher and lower PHY rates. Recall, the FATE equation is expressed in terms of three quantities: (1) PHY rate, (2) frame error rate; and (3) number of contending stations in the BSS. During the prediction phase, the PHY rates are known (they are just the adjacent higher and lower PHY rates); the frame error rate is also known (channel errors are obtained from the SNR/BER curves, and the collision probability is negligible by definition of CLC); and the number of contending stations in the BSS is known (N is an input to the CLC algorithm). Therefore, the prediction process is fairly straightforward.
5: Concurrent with the previous step, **CLLA** estimates the values of p_e, p_c, and PLR, at the adjacent higher and lower bitrates (as described in Chapter 5). It then sends this information to the VQA Estimator.
6: **VQA Estimator** estimates the received video quality at the adjacent parameter instantiations (e.g. higher and lower PHY rates, and the associated bitrates and PLRs computed in the previous steps)
7: Comparing the results of steps 3 and 6, **CLLA** selects the PHY rate associated with the best anticipated video quality
8: **CLC** naturally reacts to the PHY rate selection and performs its airtime fair congestion control tasks (e.g. bitrate adaptation and CW adaptation as described in Chapter 4).
9: **until** end of video session

Figure 7.2: TCCLA consists of a synergy between the cross-layer link adaptation strategy presented in Chapter 5, the cross-layer congestion control algorithm presented in Chapter 4, and the video quality assessment strategy presented in Chapter 6.

selects PHY rates according to some set of perceived performance indicators). It is natural to maintain this relationship in TCCLA.

In principle, TCCLA follows the general framework for perceptual quality-based cross-layer adaptive resource control that we presented in Chapter 6 (Figure 6.1). The specific algorithmic steps are shown in Figure 7.2. The correspondence with the steps outlined in Figure 6.1 is clear.

7.3 Performance Analysis

Next, we evaluate experimentally the performance of TCCLA based on the improvement in SSIM (we use the single-scale (SS) version of the structural similarity index (SS-SSIM))[4], PLR, and reliability, and ultimately show that TCCLA not only performs better than today's most common off-the-shelf link adaptation algorithm, Auto Rate Fallback (ARF) [22], but also the vanilla CLLA approach presented in Chapter 5.

7.3.1 Experimental Setup

We implemented TCCLA on Axis M1031W wireless IP cameras from Axis Communications. The Axis M1031W cameras run a lightweight embedded version of Linux, and we implemented the TCCLA algorithm as a shell script. By default, Axis M1031W cameras use ARF as the link adaptation algorithm.

Our setup is based on a typical surveillance/monitoring application: Six cameras, scattered spatially, stream video (H.264 constant bitrate 600 Kbps, GOV=15, Resolution=640x480, Framerate=15), wirelessly (IEEE 802.11b, Marvell 88w8385-BDK1 Chipset, cf8385 driver) to a security center (Dell OptiPlex GX280) via a single access point (Siemens AP2630). The OptiPlex is wired to the AP, so there is only one wireless hop (from the cameras to the AP). The cameras are looking at scenes recorded from inside a major metropolitan subway station, and on average, the amount of motion in all scenes is the same for all cameras. We monitored system statistics in real-time using custom plugins we developed for WildPackets AiroPeek [17], and on each camera, our shell script dumped statistics computed from inside the TCCLA algorithm. Of the six cameras, five remained stationary at locations with very high SNRs, and one camera was mobile and visited stops along a path that covered a variety of SNRs. The experiments were conducted on a clean, isolated wireless channel. Figure 7.3 shows the SNR for the mobile camera and one of the stationary cameras.

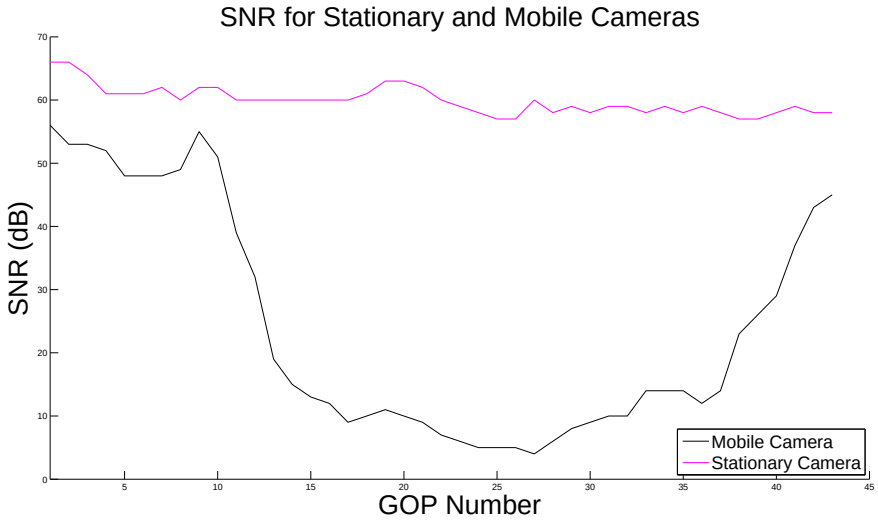


Figure 7.3: Channel SNR for the mobile camera and one of the stationary cameras

7.3.2 Results

Figure 7.4(a) shows the SSIM of the received video from the mobile camera under three different resource control strategies (TCCLA, CLLA, ARF). The figure demonstrates a drastic improvement in received video quality when using TCCLA over both CLLA and ARF (the figure also shows that CLLA outperforms ARF, but this was already discussed thoroughly in Chapter 5; our focus in this chapter is TCCLA). There are two main reasons why TCCLA improves over CLLA. First, CLLA uses a blind estimate of the MSE as the rate selection decision. Although this estimate was mathematically simple to compute, it was perceptually inaccurate. TCCLA leverages the VQA estimation technique presented in Chapter 6, and therefore, can base its rate selection decisions on highly accurate full reference estimates of the received video quality. Second, in isolation, CLLA cannot control the source rate (e.g. video bitrate). Consequently, although it can try to implicitly control congestion by balancing predictions of p_e and p_c, it cannot do anything to protect the network from a camera that utilizes a bitrate that's higher than the network can accommodate, thus causing significant packet loss due to collisions and transmission queue overflow. By adding CLC to CLLA, we can control congestion across OSI layers, thus eliminating such errors; p_c becomes negligible, and

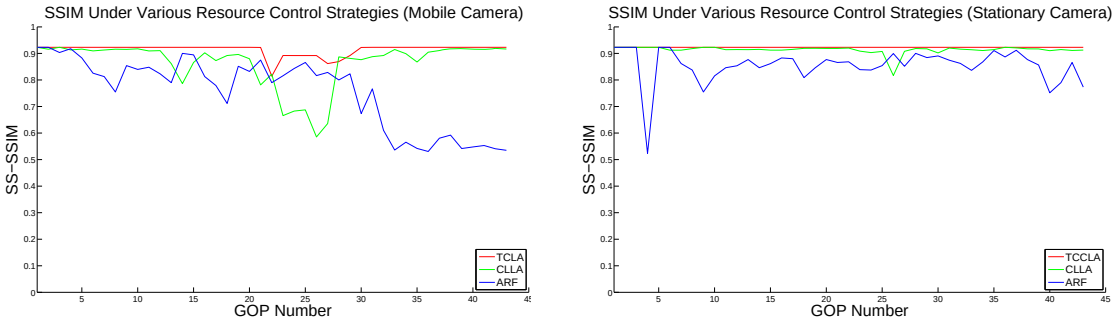
the only losses are due to channel errors that cannot be mitigated by link adaptation.

The CLLA algorithm yields SSIM values that hover between 0.6 and 0.8 for extended periods of time, and studies of the Human Visual System (HVS) [4] demonstrate that this would yield a visibly annoying video signal to the end user. On the other hand, with TCCLA, the SSIM stays above 0.9 most of the time, with two brief excursions within the 0.8-0.9 SSIM range. It has been shown that this would represent a visibly pleasing video to a human observer [4].

Also, notice that the SSIM value never exceeds 0.923 for any of the resource control algorithms. This is because of the rate-distortion response of the H.264/AVC codec at the maximum bitrate of 600 Kbps. Actually, our results from Chapter 4 indicate that we could have easily raised the maximum bitrate from 600 Kbps to 800 Kbps without causing any system instability. However, to meaningfully (and fairly) assess the three resource control algorithms against each other, we kept the maximum bitrate at the level of the saturation threshold of the weakest algorithm (ARF); comparing measurements from a stable system with an unstable system does not provide us with any meaningful insights.

Finally, note that the SSIM performance of the ARF system is significantly worse than the other two approaches. There is a drastic drop-off at about GOP number 28, indicating characteristic snowball behavior. Indeed, as evident by the PHY rates shown in Figure 7.5(a), we see the corresponding PHY rate drop-off characteristic of the snowball effect. This is because the ARF system cannot balance p_e with p_c, and furthermore, it has no control over video bitrates.

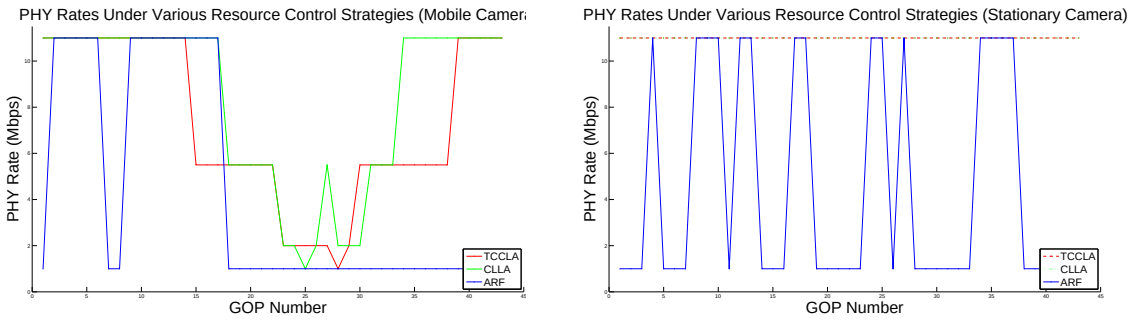
Figure 7.5(a) shows the PHY rates selected under three different resource control strategies (TCCLA, CLLA, and ARF). ARF demonstrates the characteristic snowball behavior, with catastrophic failure ensuing at GOP number 17. CLLA performs better than ARF, as expected, and as explained in Chapter 5. Although Figure 7.4(a) clearly shows that TCCLA outperforms the other two algorithms in terms of SSIM, comparing the PHY rates for TCCLA and CLLA is not so insightful other than the trivial observation that their PHY rate behavior is different. Indeed, there is no reason to expect to draw any meaningful insights by comparing CLLA and TCCLA PHY rate



(a) (b)

Figure 7.4: SSIM of the received video signal under three different resource control strategies (TCCLA, CLLA, ARF); (a) mobile camera, (b) one of the five stationary cameras. TCCLA offers significant improvement over both CLLA and ARF

traces since the two approaches select PHY rates according to a fairly complex set of underlying criteria: CLLA selects PHY rates according to a balance between p_c and p_e, as measured by a blind estimate of MSE; TCCLA selects PHY rates according to the overall received SSIM of the video signal, and this estimate inherently considers PLR performance and bitrate adaptation.



(a) (b)

Figure 7.5: PHY rate selection under three different resource control strategies (TCCLA, CLLA, ARF); (a) mobile camera, (b) one of the five stationary cameras. TCCLA and CLLA yield stable PHY rates, but ARF yields oscillating results which eventually cascade into catastrophic failure due to the snowball effect

Figure 7.4(b) shows the SSIM of the received video from one of the five stationary cameras under three different resource control strategies (TCCLA, CLLA, ARF). The figure demonstrates that under TCCLA, the SSIM performance is ideal (i.e. it tracks the maximum possible SSIM of 0.923 given the rate-distortion response of the H.264/AVC codec). This is because under TCCLA, there are no losses due to collisions (because of

CLC). And furthermore, since this was a stationary camera, there are no losses due to the channel (because it was placed in close proximity to the AP). Note that in reality, it is not at all likely that this curve would be a constant 0.923, but rather it would be a noisy line centered about 0.923. This particular curve has been smoothed to more clearly illustrate the fact that TCCLA tracks the ideal SSIM for a given maximum bitrate of 600 Kbps.

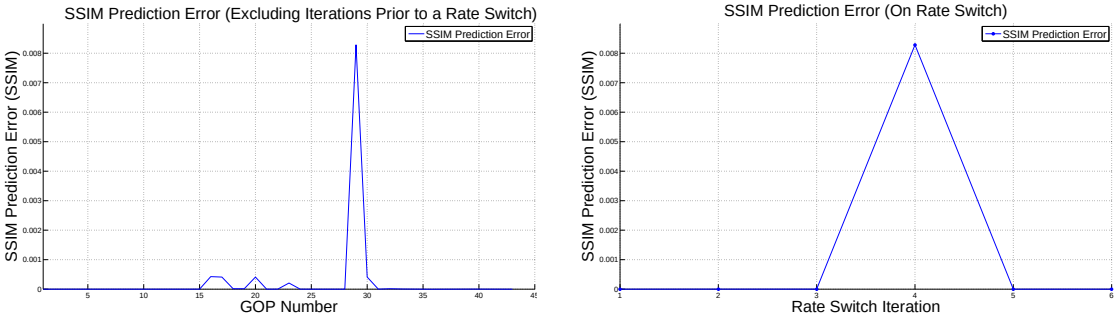
The SSIM under ARF is consistently worse than it is for the other two algorithms. Although the SSIM scores do not indicate snowball behavior, we see oscillations in the PHY rates as shown in Figure 7.5(b). These oscillations usually lead to a full-blown catastrophic failure (you can see this in the PHY rate plot for Camera 1 (Figure 7.5(a)), which oscillated and then snowballed). If we had let the experiment run a bit longer, or if we were not so fortuitous, probably we would have seen the snowball occur for the stationary camera as well, and the SSIM values would have plummeted dramatically as they did for the mobile camera in Figure 7.4(a).

Figure 7.5(b) shows the PHY rates selected under three different resource control strategies (TCCLA, CLLA, and ARF). Both TCCLA and CLLA track the maximum PHY rate of 11 Mbps. Indeed, this is expected since the stationary cameras are placed in close proximity to the AP, with essentially a perfect SNR. Therefore, there are no channel losses, and consequently, there is no reason, in theory, for the PHY rate to change. Since CLLA balances p_e with p_c, as explained in Chapter 5, there is no possibility that losses due to collisions can confuse TCCLA or CLLA. Under ARF, we see repeated PHY rate oscillations throughout the duration of the experiment as mentioned above. ARF is clearly confused because it cannot balance p_e with p_c.

On average, TCCLA offers an 0.1595 absolute SSIM improvement over ARF, and an 0.05 absolute SSIM improvement over CLLA for the mobile camera, which represents a visibly noticeable difference to human observers.

Figures 7.6(a) and (b) demonstrate the prediction accuracy of our VQA Estimation module. As shown in Figure 7.6(b), for the mobile camera, there were six rate switching instances, and with the exception of the 4th switch, the VQA prediction error was negligible (it was on the order of 10^{-6}). Figure 7.6(a) shows the overall VQA prediction

error for every iteration of the TCCLA algorithm except for iterations immediately preceding a rate switch. Periods immediately before a rate switch are associated with higher overall VQA prediction errors because a subsequent rate switch indicates that some major change has occurred in terms of p_e, p_c, or both, and therefore, the predicted VQA estimate will be off. However, this does not degrade the performance of TCCLA in any way because on the very next iteration, the change is detected, and a rate switch is made.



(a) (b)

Figure 7.6: Prediction accuracy of the VQA Estimation module during (a) every iteration of the TCCLA algorithm except for iterations immediately preceding a rate switch; (b) rate switching iterations. The results indicate that the VQA Estimation module is nearly perfect in predicting the received video signal at the transmitter.

It is thus clear that TCCLA offers significant improvement over ARF and CLLA in terms of SSIM, PLR, throughput, and reliability.

7.4 Conclusion

We proposed a resource control strategy called TCCLA (Task-Centric Congestion Control and Link Adaptation) which jointly performs link adaptation and congestion control driven by a task-centric resource allocation strategy. The TCCLA algorithm utilized a synergy of the congestion control, link adaptation, and video quality assessment strategies proposed in Chapters 4, 5, and 6, respectively. Through analysis in both simulation and implementation on real wireless cameras, we demonstrated an average absolute SSIM improvement of 0.1595 over today's state of the art link adaptation algorithm, ARF, which represents a significant visual improvement to human observers.

Although we focused on SSIM (which is a human oriented perceptual quality metric), TCCLA can be driven by any full-reference metric, thus offering the best possible flexibility in offering cross layer resource control for not only user-centric applications, but also machine/task-centric applications.

Chapter 8

Conclusion

Fueled by increasingly robust video codecs and the proliferation of wireless technology, the increasing demand for wireless video streaming in everyday life is undeniable. With applications including such things as IPTV, telemedicine, surveillance, and teleconferencing, digital video streaming is permeating our lives, and the wireless medium has become an attractive alternative to conventional wire-line content delivery. Unfortunately, today's state of the art local wireless technologies, such as IEEE 802.11, can be easily demonstrated to fail when subjected to the conditions associated with many of these real-world applications. This is because wireless video streaming involves complex relationships between the the video codec, the wireless PHY and MAC, and the Human Visual System's (HVS) perception of the video signal itself, and these relationships are not well understood or exploited by today's video streaming systems.

In this dissertation, we discussed several key obstacles to wireless video streaming, and proposed a set of adaptive real-time cross-layer approaches to deal with them.

The key obstacles presented were (1) instability of link adaptation under the congested scenarios associated with wireless transmission of multiple simultaneous uplink video streams; (2) congestion control with respect to the fairness policies employed by the IEEE 802.11 channel access mechanism; and (3) the problem of video quality assessment in adaptive real-time cross-layer video streaming systems.

After that, we proposed several adaptive real-time cross-layer solutions to deal with these obstacles. They included: (i) airtime fair distributed cross-layer congestion control in multi-rate wireless environments; (ii) video quality assessment in adaptive real-time cross-layer video streaming systems; (iii) cross-layer link adaptation based on perceptual

video quality; and (iv) joint link adaptation and congestion control driven by user/task-centric resource allocation. We supported the proposed algorithms through simulations, theory, and experiments with real wireless devices on which we have implemented our algorithms.

Our airtime-fair distributed cross-layer congestion control algorithm (CLC) roughly doubled the steady state aggregate throughput in multi-camera uplink video streaming scenarios, virtually eliminated packet loss across all cameras, and prevented catastrophic failure/loss of service. Our cross-layer link adaptation algorithm resulted in more than a 4x reduction in MSE compared to off-the-shelf link adaptation (ARF), and more than doubled the steady-state aggregate throughput achieved by ARF. Our TCCLA algorithm demonstrated an average absolute SSIM improvement of 0.1595 over today's state of the art link adaptation algorithm, ARF, representing a significant visual improvement for human observers.

In Chapter 6, we investigated the use of perceptually accurate video quality assessment (VQA) in the context of real-time adaptive video streaming systems. Aside from discussing the great difficulty in designing perceptually accurate video quality metrics themselves, we presented a number of other challenges and solutions in using such metrics in a real-time adaptive system. They included: (1) estimating the quality metric at the encoder with no direct access to the received video signal; (2) predicting how the perceptual quality would change if some adjustment is made to the instantiation of the parameter space either in the video codec or in the wireless PHY/MAC; and (3) how to balance complexity with accuracy for suitability in a real-time system with potentially limited computation and memory resources. Solving these issues leads to a framework for VQA-based adaptive cross-layer video streaming systems, which we used in Chapter 7 to enable joint link adaptation and congestion control driven by user/task-centric resource allocation.

Both wireless video streaming and video quality assessment are broad research areas with many opportunities for future work. In our opinion, some of the most interesting topics include: (1) Extending TCCLA to consider a broader set of parameters to control, especially parameters related to error concealment such as slicing strategies; and (2)

developing task-specific metrics for video quality assessment (e.g. metrics that can accurately predict performance of object tracking algorithms).

References

[1] M. Heusse, F. Rousseau, G. Berger-Sabbatel, and A. Duda. Performance Anomaly of 802.11b. In *Proceedings of the IEEE Conference on Computer Communications (Infocom)*, 2003.

[2] T. Wiegand, G.J. Sullivan, G. Bjontegaard, and A. Luthra. Overview of the h.264/avc video coding standard. *Circuits and Systems for Video Technology, IEEE Transactions on*, 13(7):560 –576, july 2003.

[3] Alan C Bovik. *The essential guide to video processing.* Elsevier, Amsterdam, 2009.

[4] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *Image Processing, IEEE Transactions on*, 13(4):600–612, April 2004.

[5] Zhou Wang and A.C. Bovik. Mean squared error: Love it or leave it? a new look at signal fidelity measures. *Signal Processing Magazine, IEEE*, 26(1):98 –117, jan. 2009.

[6] Pierre Ferre, James Chung-How, David Bull, and Andrew Nix. Distortion-based link adaptation for wireless video transmission. *EURASIP Journal on Advances in Signal Processing*, 2008.

[7] Michael Loiacono, Jeffrey Johnson, Justinian Rosca, and Wade Trappe. Cross-layer link adaptation for wireless video. *Communications, 2010. ICC '10. IEEE International Conference on*, pages 2270–2276, May 2010.

[8] Lin Liu, Sanyuan Zhang, Xiuzi Ye, and Yin Zhang. Error resilience schemes of h.264/avc for 3g conversational video services. In *CIT '05: Proceedings of the The Fifth International Conference on Computer and Information Technology*, pages 657–661, Washington, DC, USA, 2005. IEEE Computer Society.

[9] Thomas Stockhammer. Optimized transmission of h.26l/jvt coded video over packet-lossy networks. In *Proc. ICIP 2002*, pages 173–176, 2002.

[10] Thomas Stockhammer and Thomas Wiegand. Video coding and transport layer techniques for h.264/avc-based transmission over packet-lossy networks. In *IEEE International Conference on Image Processing (ICIP 2003*, 2003.

[11] J.-O. Fajardo, F. Liberal, and B. Nagore. Impact of the video slice size on the visual quality for h.264 over 3g umts services. In *BROADNETS '09: Proceedings of the The Sixth International ICST Conference on Broadband Communications, Networks, and Systems*. ICST, 2009.

[12] Michal Ries and Olivia Nemethova. Video quality estimation for mobile h.264/avc video streaming.

[13] T. Stockhammer, M.M. Hannuksela, and T. Wiegand. H.264/avc in wireless environments. *Circuits and Systems for Video Technology, IEEE Transactions on*, 13(7):657–673, July 2003.

[14] J. Yun and S. Seo. Collision Detection based on Transmission Time Information in IEEE 802.11 Wireless LAN. In *Proceedings of the IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOMW)*, 2006.

[15] J. Kim, S. Kim, S. Choi, and D. Qiao. CARA: Collision-Aware Rate Adaptation for IEEE 802.11 WLANs. In *Proceedings of the IEEE Conference on Computer Communications (Infocom)*, 2006.

[16] I. Haratcherev, J. Taal, K. Langendoen, R. Lagendijk, and H. Sips. Automatic IEEE 802.11 Rate Control for Streaming Applications. *Wireless Communications and Mobile Computing*, 5(4):421–437, 2005.

[17] AiroPeek SE, WildPackets, Inc. http://www.wildpackets.com/.

[18] Michele Garetto and Carla-Fabiana Chiasserini. Performance Analysis of 802.11 WLANs Under Sporadic Traffic. In Raouf Boutaba, Kevin C. Almeroth, Ramón Puigjaner, Sherman X. Shen, and James P. Black, editors, *NETWORKING*, volume 3462 of *Lecture Notes in Computer Science*, pages 1343–1347. Springer, 2005.

[19] G. Bianchi. Performance Analysis of the IEEE 802.11 Distributed Coordination Function. *IEEE Journal on Selected Areas in Communications*, 18(3):535–547, 2000.

[20] ANSI/IEEE Std 802.11. Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. 1999.

[21] The network simulator - ns2. http://www.isi.edu/nsnam/ns/.

[22] A. Kamerman and L. Monteban. WaveLAN II: A high-performance wireless LAN for the unlicensed band. *Bell Labs Technical Journal*, 2(3):118–133, 1997.

[23] D. Qiao and S. Choi. Goodput enhancement of IEEE 802.11a wireless LAN via link adaptation. In *Proceedings of the IEEE Conference on Communications (ICC)*, 2001.

[24] P. Chevillat, J. Jelitto, A.N. Barreto, and H.L. Truong. A dynamic link adaptation algorithm for IEEE 802.11 a wireless LANs. In *Proceedings of the IEEE Conference on Communications (ICC)*, 2003.

[25] M. Loiacono, J. Rosca, and W. Trappe. The Snowball Effect: Detailing Performance Anomalies of 802.11 Rate Adaptation. In *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM)*, November 2007.

[26] `http://www.wi-fi.org/`.

[27] Q. Li and M. van der Schaar. Providing Adaptive QoS to Layered Video over Wireless Local Area Networks through Real-Time Retry Limit Adaptation. *IEEE Transactions on Multimedia*, April 2004.

[28] L. Haratcherev, J. Taal, K. Langendoen, R. Lagendijk, and H. Sips. Optimized video streaming over 802.11 by cross-layer signaling. *Communications Magazine, IEEE*, 44:115–121, 2006.

[29] S. Khan, Y. Peng, E. Steinbach, M. Sgroi, and W. Kellerer. Application-driven cross-layer optimization for video streaming over wireless networks. *Communications Magazine, IEEE*, 44:122–130, 2006.

[30] E. Setton, Taesang Yoo, Xiaoqing Zhu, A. Goldsmith, and B. Girod. Cross-layer design of ad hoc networks for real-time video streaming. *Wireless Communications, IEEE [see also IEEE Personal Communications]*, 12(4):59–65, Aug. 2005.

[31] Chih-Wei. Huang, Michael Loiacono, Justinian Rosca, and Jenq-Neng Hwang. Distributed cross layer congestion control for real-time video over wlan. *Communications, 2008. ICC '08. IEEE International Conference on*, pages 2270–2276, May 2008.

[32] Mung Chiang, S.H. Low, A.R. Calderbank, and J.C. Doyle. Layering as optimization decomposition: A mathematical theory of network architectures. *Proceedings of the IEEE*, 95(1):255–312, Jan. 2007.

[33] L. Chen, S. H. Low, M. Chiang, and J. C. Doyle. Cross-layer congestion control, routing and scheduling design in ad hoc wireless networks. *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pages 1–13, April 2006.

[34] Lijun Chen, S.H. Low, and J.C. Doyle. Joint congestion control and media access control design for ad hoc wireless networks. *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, 3:2212–2222 vol. 3, March 2005.

[35] Xiaojun Lin and N.B. Shroff. The impact of imperfect scheduling on cross-layer congestion control in wireless networks. *Networking, IEEE/ACM Transactions on*, 14(2):302–315, April 2006.

[36] S.J. Golestani. A self-clocked fair queueing scheme for broadband applications. In *Proceedings of the IEEE Conference on Computer Communications (Infocom)*, 1994.

[37] N. Vaidya, A. Dugar, S. Gupta, and P. Bahl. Distributed fair scheduling in a wireless LAN. *WMobile Computing, IEEE Transactions on*, 4(6):616– 629, 2005.

[38] Daji Qiao and K. G. Shin. Achieving efficient channel utilization and weighted fairness for data communications in IEEE 802.11 WLAN under the DCF. In *Quality of Service, IEEE International Workshop on*, pages 227–236, 2002.

[39] D.-Y. Yang, T.-J. Lee, K. Jang, J.-B. Chang, and S. Choi. Performance enhancement of multirate IEEE 802.11 WLANs with geographically scattered stations. *Mobile Computing, IEEE Transactions on*, 5:906–919, 2006.

[40] C.-T. Chou, K. G. Shin, and Sai N Shankar. Contention-based airtime usage control in multirate IEEE 802.11 wireless LANs. *IEEE/ACM Transactions on Networking*, 14(6), 2006.

[41] B. Radunovic and J.Y. Le Boudec. Rate performance objectives of multihop wireless networks. *Mobile Computing, IEEE Transactions on*, 3(4):334–349, Oct.-Dec. 2004.

[42] Li Bin Jiang and Soung Chang Liew. Proportional fairness in wireless lans and ad hoc networks. *Wireless Communications and Networking Conference, 2005 IEEE*, 3:1551–1556 Vol. 3, March 2005.

[43] Martin Heusse, Franck Rousseau, Romaric Guillier, and Andrzej Duda. Idle sense: an optimal access method for high throughput and fairness in rate diverse wireless lans. In *SIGCOMM '05: Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 121–132, New York, NY, USA, 2005. ACM.

[44] Hyogon Kim, Sangki Yun, Inhye Kang, and Saewoong Bahk. Resolving 802.11 performance anomalies through qos differentiation. *Communications Letters, IEEE*, 9(7):655–657, July 2005.

[45] Jangeun Jun, P. Peddabachagari, and M. Sichitiu. Theoretical maximum throughput of IEEE 802.11 and its applications. In *Network Computing and Applications, IEEE International Symposium on*, pages 249– 256, 2003.

[46] HFA3861B: Direct Sequence Spread Spectrum Baseband Processor. Intersil, 2000.

[47] Mathieu Lacage, Mohammad Hossein Manshaei, and Thierry Turletti. Ieee 802.11 rate adaptation: a practical approach. In *MSWiM '04: Proceedings of the 7th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems*, pages 126–134, New York, NY, USA, 2004. ACM.

[48] S. Wenger. H.264/avc over ip. *Circuits and Systems for Video Technology, IEEE Transactions on*, 13(7):645–656, July 2003.

[49] Chih-Wei. Huang, Michael Loiacono, Justinian Rosca, and Jenq-Neng Hwang. Airtime fair distributed cross-layer congestion control for real-time video over wlan. *Circuits and Systems for Video Technology, IEEE Transactions on*, 19(8):1158–1168, Aug. 2009.

[50] S. Khan, Y. Peng, E. Steinbach, M. Sgroi, and W. Kellerer. Application-driven cross-layer optimization for video streaming over wireless networks. *Communications Magazine, IEEE*, 44(1):122–130, Jan. 2006.

[51] F. Xiao. DCT-based video quality evaluation. In *Technical report, Stanford University*, 2003.

[52] Telchemy. Proposed QoE algorithm based on PSNR estimation. In *ITUT Focus Group on IPTV*, 2007.

[53] Michael Yuen and H. R. Wu. A survey of hybrid mc/dpcm/dct video coding distortions. *Signal Process.*, 70(3):247–278, 1998.

[54] C. VAN DEN BRANDEN LAMBRECHT and 0. VERSCHEURE. Perceptual quality measure using a spatiotemporal model of the human visual system. In *Proceedings of the SPIE*, volume 2668, 1996.

[55] S. Winkler. A perceptual distortion metric for digital color video. In *Proceedings of the SPIE Conference on Human Vision and Electronic Imaging*, volume 3644 of *Controlling Chaos and Bifurcations in Engineering Systems*, pages 175–184. IEEE, 1999.

[56] Andrew B. Watson, James Hu, and John F. McGowan III. Digital video quality metric based on human vision. *Journal of Electronic Imaging*, 10(1):20–29, 2001.

[57] J. Lubin. A human vision system model for objective picture quality measurements. *IEE Conference Publications*, 1997(CP447):498–503, 1997.

[58] `http://www.tek.com/products/video_test/pqa500/`.

[59] `http://www.acceptv.com/`.

[60] K. Seshadrinathan and A. Bovik. Motion tuned spatio-temporal quality assessment of natural videos. In *IEEE Transactions on Image Processing*, 2010.

[61] International Telecommunications Union Std. Objective perceptual multimedia video quality measurement in the presence of a full reference. 2008.

[62] NTT News Release. 2008.

[63] Opticom.

[64] Matthias Malkowski and Daniel Claben. Performance of video telephony services in umts using live measurements and network emulation. *Wirel. Pers. Commun.*, 46(1):19–32, 2008.

[65] M. Barkowsky, J. Bialkowski, R. Bitto, and A. Kaup. Temporal registration using 3d phase correlation and a maximum likelihood approach in the perceptual evaluation of video quality. pages 195 –198, oct. 2007.

[66] A. P. Hekstra, J. G. Beerends, D. Ledermann, F. E. de Caluwe, S. Kohler, R. H. Koenen, S. Rihs, M. Ehrsam, and D. Schlauss. Pvqm - a perceptual video quality measure. *Signal Processing: Image Communication*, 17(10):781 – 798, 2002.

[67] Z. Wang, E.P. Simoncelli, and A.C. Bovik. Multiscale structural similarity for image quality assessment. In *Asilomar Conference on Signals, Systems and Computers*, volume 2, pages 1398 – 1402 Vol.2, nov. 2003.

[68] A. K. Moorthy and A. C. Bovik. Perceptually significant spatial pooling techniques for image quality assessment. In *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, volume 7240 of *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, February 2009.

[69] U. Rajashekar, I. van der Linde, A.C. Bovik, and L.K. Cormack. Gaffe: A gaze-attentive fixation finding engine. *Image Processing, IEEE Transactions on*, 17(4):564 –573, april 2008.

[70] Zhou Wang and Qiang Li. Video quality assessment using a statistical model of human visual speed perception. *J. Opt. Soc. Am. A*, 24(12):B61–B69, 2007.

[71] Alan Stocker and Eero Simoncelli. Noise characteristics and prior expectations in human visual speed perception. *Nature Neuroscience*, 9:578–585.

[72] B. A. Wandell. *Foundations of Vision*. Sinauer Associates Inc., Sunderland, MA, 1995.

[73] Video Quality Experts Group.

[74] A. K. Moorthy and A. C. Bovik. Wireless video quality assessment. *Circuits and Systems for Video Technology, IEEE Transactions on*, PP(99):1 –1, 2010.

[75] D. Comaniciu, V. Ramesh, and P. Meer. Real-time tracking of non-rigid objects using mean shift. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 142 –149 vol.2, 2000.

[76] Vasu Parameswaran, Visvanathan Ramesh, and Imad Zoghlami. Tunable kernels for tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2179–2186, 2006.

[77] H.264/avc jm reference software, 2008.

Vita

Michael T. Loiacono

Education

2007-2011 Ph.D. in Electrical and Computer Engineering, Rutgers University

2005-06 M.S. in Electrical and Computer Engineering, Rutgers University

2000-04 B.S.E. in Computer Systems Engineering, Arizona State University

Key Positions Held

2011 - **Siemens Energy, Inc.** *Wendell, NC*
Director, Communications Technology, U.S. Market
Smart Grid Applications

2003 - 2011 **Siemens Corporate Research** *Princeton, NJ*
Research Scientist
Intelligent Systems and Control

2002 **Glemser Technologies** *Bethlehem, PA*
Computer Systems Validation Analyst (Internship)