End-to-end QoS management for multimedia over IP

Zhan, Wenning

2005

Zhan, W. (2005). End-to-end QoS management for multimedia over IP. Master's thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/2576>

<https://doi.org/10.32657/10356/2576>

Nanyang Technological University

Downloaded on 26 Apr 2023 11:54:26 SGT

402352

End-to-End QoS Management for Multimedia over IP

Zhan Wenning

School of Computer Engineering

A thesis submitted to the Nanyang Technological University
in fulfillment of the requirement for the degree of
Master of Engineering

2005

TK
5105.15
Z63
2005

Abbreviation

ABW : Available Bandwidth
ACK : Acknowledgement
AIMD : Additive Increase Multiplicative Decrease
APP : Application-Defined Packet
ARQ : Automatic Repeat reQuest
BER : Bit Error Rate
BTT : Backward Trip Time
BYE : Goodbye Packet
CSRC : Contributing Source
DTS : Decoder Time Stamp
FEC : Forward Error Correction
FTT : Forward Trip Time
GEC : Gilbert-Elliott channel
IETF : Internet Engineering Task Force
IP : Internet protocol
MTU : Maximum Transmission Unit
NAK : Negative Acknowledgement
PDF : Probability Density Function
PHB : Per-Hop Behavior
PLR : Packet Loss Rate
PSNR : Peak Signal to Noise Ratio
QoS : Quality of Service
RCPC : Rate-Compatible Punctured Convolutional

RR : Receiver Report Packet
RS : Reed-Solomon
RSVP : Resource Reservation Protocol
RTO : Retransmission TimeOut
RTP : Real-time Transport Protocol
RTCP : RTP Control Protocol
RTSP : Real-time Streaming Protocol
RTT : Round Trip Time
SDES : Source Description Packet
SR : Sender Report Packet
SSRC : Synchronization Source
TCP : Transmission Control Protocol
TFRC : TCP Friendly Rate Control
TTL : Time to Live
UDP : User Datagram Protocol
UEP : Unequal Error Protection
UXP : Unequal Erasure Protection

Abstract

Due to the rapidly growing demands of multimedia communications with guaranteed quality of service (QoS), traditional Internet communications based on best-effort are not suitable any more, and the topic of end-to-end QoS management for multimedia communications over IP has received great attention in the past few years. It is a very challenging task to study end-to-end QoS management because of the unexpected packet loss, the bandwidth and delay fluctuation, and the heterogeneous property of networks.

The objective of this research is to study, design and implement end-to-end QoS management schemes, particularly on congestion control and error control in the transportation layer, for multimedia communications over IP. In my two-year study, we focus on two theoretical problems, i.e., wireless link modeling and optimal retransmission timeout (RTO) selection. we also develop a practical RTP-based transportation system for multimedia streaming over IP with end-to-end QoS management.

In particular, we propose to model the wireless link layer by converting the widely used Gilbert-Elliott channel (GEC) at different levels. We develop an analytical procedure to derive the parameters of a desired symbol-level GEC model, from the parameters of a given bit-level GEC. It can be applied to any problem of converting GEC from a level of shorter data segment to a level of longer data segment as long as the longer data segment is a multiple of the shorter data segment.

We also study the problem of optimal retransmission timeout (RTO) selection for end-to-end delay-constrained multimedia communications. Based on a statistical delay model of IP network, we propose a novel algorithm for optimal RTO selection in delay-constrained multimedia communications. By appropriately choosing the RTO, we are able to adjust packet delay and transmission rate, and then achieve the optimal tradeoff between the error probability and the rate cost.

Moreover, we develop a RTP-based transportation system based on Darwin Streaming Server and MPEG4IP player. We have incorporated TCP friendly rate control (TFRC) into the system for congestion control, and two error control schemes to combat with packet loss, the class-based fast retransmission and unequal erasure protection (UXP). A simple interface is defined for the application and the transportation system to exchange information for QoS management. Experimental results show that the developed system is able to provide end-to-end QoS management appropriately.

Acknowledgments

First of all, I would like to express my deep sense of gratitude to my supervisor, Dr. Cai Jianfei, for his heuristic instruction, kindly criticism and constant encouragement during these years. His guidance, patience, and prudence are my drive of inspiration. He is impressing me so much that it is impossible for me to complete the thesis without his supervision and encouragement.

I must express my sincere acknowledgement to Mr. Ni Zefeng and Wu Jianhua, who provide technical support and assistance to my programs. Without their help, I can hardly finish the last project. I am also thankful to other workmates, Li Xiangjun, Yang Wenxian and Dr. Gao Deyun. I benefit so much from their friendly suggestion, discussion and cooperation.

Last but not the least, I must give the special thanks to my parents and my girl friend. Their love is the power for me to overcome the difficulties and complete my research work.

Contents

Abbreviation	i
Abstract	iii
Acknowledgments	v
List of Figures	ix
List of Tables	xi
1 Introduction	1
1.1 Research Background	1
1.1.1 Multimedia Communications	1
1.1.2 Quality of Service	4
1.1.3 QoS Management	5
1.2 Problem Statement and Research Objective	7
1.3 Summary of Contributions	9
1.4 Outline	10
2 Literature Review	11
2.1 RTP and RTCP	11
2.2 Congestion Control	16
2.2.1 Window-based Congestion Control	16
2.2.2 Rate-based Congestion Control	18
2.3 Error Control	20
2.3.1 Forward Error Correction (FEC)	21
2.3.2 Auto Repeat reQuest (ARQ)	25
2.3.3 Hybrid ARQ	26
2.4 Summary	28

3	GEC Conversion for Wireless Link Modeling	29
3.1	Related Work	29
3.2	Converting GEC to A Level of Longer Data Segment	30
3.2.1	Conversion of Simplified GECs	31
3.2.2	Conversion of General GECs	34
3.3	Experimental Results	35
3.3.1	Results of Converting Simplified GECs	35
3.3.2	Results of Converting General GECs	35
3.3.3	Results of GEC Conversion for Wireless Link Modeling	37
3.4	Summary	39
4	Optimal Retransmission Timeout (RTO) Selection for End-to-End QoS	40
4.1	Related Work	40
4.2	Optimal RTO selection	41
4.2.1	Network Setup	43
4.2.2	Proposed Optimal RTO Selection	45
4.3	Preliminary Results	46
4.4	Summary	49
5	RTP-Based Transportation Platform with End-to-End QoS Manage-	
	ment	50
5.1	Background	50
5.2	System Architecture	51
5.3	Congestion Control	54
5.4	Error Control	58
5.4.1	Class-Based Fast Retransmission	58
5.4.2	Unequal Erasure Protection (UXP)	60
5.5	System Implementation	63
5.6	Experimental Results	65
5.7	Summary	67

6 Conclusion and Future work	69
6.1 Conclusion	69
6.2 Future work	70
A System Integration	72
A.1 Darwin Streaming Server	72
A.2 MPEG4IP Player	75
References	78
Publications	86

List of Figures

1.1	Internet protocol suite for real-time multimedia applications.	4
1.2	The architecture for a typical wireless multimedia streaming system. . . .	8
2.1	The RTP packet format.	13
2.2	The architecture of a RTP/RTCP based video transmission system (copied from [14]).	15
2.3	Window-based congestion control in TCP Reno.	18
2.4	Structure of a RS codeword.	22
2.5	Comparison between interleaving and pre-interleaving (Copied from [41]). Top: original interleaving. Bottom: pre-interleaving.	24
2.6	(a):Common type-II hybrid ARQ. (b) Type-II hybrid ARQ with selective combining.	27
3.1	Gilbert-Elliott Channel.	31
3.2	(a) An error burst at the beginning of a symbol; (b) An error burst in the middle of a symbol; (c) An error burst at the end of a symbol.	33
3.3	The Probabilities of codeword error for RS codes of length 31 transmitted through a general GEC with $\alpha=0.01$, $\beta=0.003$, $eg=0.001$, $eb=0.12$	37
3.4	The Probabilities of codeword error for RS codes of length 31 transmitted through a general GEC with $\alpha=0.04$, $\beta=0.001$, $eg=0.001$, $eb=0.12$	38
4.1	The performance comparison of the Actual Throughput between the traditional RTO selection and the proposed RTO selection under different average RTT. Left: with 5% packet loss rate. Right: with 20% packet loss rate.	48

4.2 The performance comparison of the Actual Throughput between the traditional RTO selection and the proposed RTO selection under different packet loss rates. Left: with the average RTT of 60ms. Right: with the average RTT of 120ms. 48

5.1 The architecture of the developed video streaming system. 53

5.2 TFRC feedback loops implemented in RTCP [71]. 55

5.3 The receiver report format for TFRC. 56

5.4 The receiver APP format [73]. 57

5.5 The sender APP format. 58

5.6 The format of a fast ACK packet. 60

5.7 The structure of UXP. 61

5.8 The format of the profile block. 62

5.9 Flow chart of QoS management. 64

5.10 The performance comparison between the transmission rates with and without congestion control. Left: PLR = 10%. Right: PLR = 20%. . . . 66

5.11 The PSNR results without error control. Left: PLR = 10%. Right: PLR = 20%. 68

5.12 The PSNR results of fast retransmission. Left: PLR = 10%. Right: PLR = 20%. 68

5.13 The PSNR results of UXP. Left: PLR = 10%. Right: PLR = 20%. . . . 68

A.1 Darwin server architecture (copied from [73]). 73

A.2 MPEG4IP player architecture (copied from [68]). 76

List of Tables

1.1	The comparison among TCP, UDP and RTP.	3
3.1	Symbol error rate for simplified GEC.	36
3.2	Average symbol-level burst length for simplified GEC.	36
3.3	Packet loss rate on IP layer.	39
4.1	The performance comparison between the traditional RTO selection and our proposed scheme under different average RTT with $\epsilon_F = \epsilon_B = 5\%$. . .	47
4.2	The performance comparison between the traditional RTO selection and our proposed scheme under different average RTT with $\epsilon_F = \epsilon_B = 20\%$. .	47
5.1	Identification of error control mode.	63
5.2	The comparison of the average PSNR (dB) in different error control modes.	67

Chapter 1

Introduction

1.1 Research Background

During the past decade, we have witnessed the immense success of Internet. With the rapidly growing demands of multimedia communications with guaranteed quality of service (QoS), traditional Internet communications based on best-effort are not suitable any more and the topic of end-to-end QoS management for multimedia communications over IP has received great attention in the past few years.

1.1.1 Multimedia Communications

Multimedia is defined as the digital signal representation of mixed information forms including data, text, audio, graphics, image and video [1]. For multimedia communications, unlike traditional data communications, multimedia has broader quality of service (QoS) requirement in terms of throughput, bit error rate (BER) / packet loss rate (PLR), delay and delay jitter. For example, the characteristics of video communications are quite different from data communications in several aspects. Firstly, video signals typically have huge volume in data size and thus large network throughput is needed. To reduce the volume and increase storage memory efficiency, video compression becomes a very important role in multimedia communications, which introduces video encoding and decoding in the applications. Secondly, video communications usually require the

CHAPTER 1. INTRODUCTION

information to be transmitted in a timely manner. Long transmission delay or large delay jitter variation are not acceptable. In addition, video communications may not need error-free transmission since certain types of degradation may be imperceptible to the human visual system. Lastly, video communications require synchronization among media streams, e.g., the synchronization between video and audio. Typically, if the audio data arrives late, the video stream has to pause and waits for the audio stream. If the video data is late, the audio will continue to play while the video data will be discarded until it catches up with audio. Therefore, directly employing the traditional data communication systems for multimedia communications is inappropriate.

Although Internet has achieved great success in multimedia communications, it only supports multimedia communications in a best-effort manner. In particular, Internet generally consists of different sub-networks with various technologies such as Ethernet, Token Ring, X.25, frame relay and etc. All these sub-networks are connected together by using the Internet protocol (IP). IP has no bandwidth restriction and it delivers whatever the line services can provide. Since IP is a connectionless network protocol and has no congestion control, this results in varying delay and packet loss [2].

In order to provide a reliable end-to-end transmission service over IP network, the Transmission Control Protocol (TCP) is commonly employed. TCP [2] provides connection-oriented service. It uses automatic repeat request (ARQ) mechanism with cyclic redundancy check (CRC) codes to ensure all data are received correctly at the destination. Although TCP can support reliable end-to-end transmission, the delay introduced is not tolerable for multimedia applications with stringent delay requirements. For those delay-constrained applications, the User Datagram Protocol (UDP) is often used. UDP [2] is an unreliable end-to-end transport protocol without congestion control and sequence numbering. Since UDP does not guarantee reliable packet transmission, the receiver needs to rely on upper layer protocols to detect packet loss and recover lost data. The

CHAPTER 1. INTRODUCTION

Real-time Transport Protocol (RTP) is the one which usually accompanies UDP to provide end-to-end transport functions for real-time multimedia applications [3]. Comparing with UDP, a key feature of RTP is to use packet sequence number to detect packet loss, but it is also lack of congestion control and error control, which may result in unfair competition with other TCP connections on network bandwidth. To provide feedback information for managing RTP streams, RTP Control Protocol (RTCP) [3] is designed as a companion control protocol with RTP, which enables end systems to monitor and handle network QoS. In addition, Real-time Streaming Protocol (RTSP)[4] is often used to enable controlled, on-demand delivery of multimedia. Although both RTCP and RTSP supply control information for RTP, neither of them consider congestion control, flow control and error control. Table 1.1 shows the comparison of TCP, UDP and RTP.

Table 1.1: The comparison among TCP, UDP and RTP.

Services	RTP over UDP	TCP	UDP
Timing information	✓	×	×
Error control	×	✓	×
Congestion/Flow control	×	✓	×
Sequence number	✓	✓	×
Media source indication	✓	×	×
Low delay	✓	×	✓

In practical multimedia application, RTP, UDP and TCP are usually combined together, i.e., using RTP/UDP to deliver delay-sensitive media data and using TCP to deliver control packets. Fig. 1.1 shows such an Internet protocol suite for multimedia services.

In a word, Internet protocols have no QoS guarantee, and the best-effort services provided by Internet limit the further development of multimedia applications. Numerous schemes such as Integrated Services [5] and Differentiated Services [6] have been developed to provide QoS for Internet. The great improvement has been achieved in the test beds.

Figure 1.1: Internet protocol suite for real-time multimedia applications.

1.1.2 Quality of Service

Quality of Service (QoS) is a term indicating how well a network performs in dealing with a multimedia application. The typical QoS parameters include delay, delay jitter, error rate, bandwidth and throughput. Delay is the time taken to transmit a packet from the sender to the receiver. Delay jitter is the variance of delay and it indicates the variance of the inter-packet arrival times. There are two types of error rates: bit error rate (BER) and packet loss rate (PLR). Bit error is usually caused by channel noise and interference while packet loss is usually caused by network congestion and delay. Bandwidth is the amount of data that can be sent across a network per second, and throughput is the amount of data that are successfully received [2]. In multimedia communications, the QoS specifies the degree of a multimedia presentation on both networked and stand alone systems.

According to the QoS parameters described above, the QoS for multimedia communications can be classified into four categories: Conversational, Streaming, Interactive and Background. The main distinguishing factor among these classes is the delay sensitivity.

CHAPTER 1. INTRODUCTION

The Conversational class is the most sensitive one while Background is the least one. Conversational and Streaming classes are designed for real-time multimedia communications. They both preserve time relation (variation) between information entities of a stream, but Conversational has stricter and lower delay requirements [7]. For the Interactive and Background classes, instead of delay, throughput is the major issue in difference. Web browsing and email downloading are the examples for Interactive and Background applications, respectively [7]. In chapter 5, we will demonstrate our end-to-end QoS management for the streaming class of multimedia communications.

To the end-to-end network, the QoS means the capability of a network to deliver service needed by specific network traffic from end to end or edge to edge. In the following we will discuss QoS management.

1.1.3 QoS Management

The goal of the QoS management is to allocate, monitor, and optimize the resources of the wired and wireless network elements, and to distribute their capacities in a fair manner among the competing applications and the associated data flows connections. QoS management has been studied at various protocol levels. It can be briefly divided into two categories: network QoS management and end-to-end QoS management.

Network QoS management is targeted to provide QoS support through carefully designing network protocols at different level and control mechanisms at different network devices, including routing and queuing management, admission control, traffic management and etc. In order to meet the increasing demand for QoS, the Internet Engineering Task Force (IETF) has proposed many service models and mechanisms. Among them, notable are the Integrated Service (IntServ) [5] / resource reservation protocol (RSVP) [8] and the Differentiated Service (DiffServ) [6].

IntServ is flow specific and is characterized by resource reservation, in which a flow is a distinguishable stream of related IP packets that results from a single user activity and

CHAPTER 1. INTRODUCTION

requires the same QoS. IntServ includes two sorts of service better than best-effort service: the controlled-load service and guaranteed service. The controlled-load service provides the data flow with QoS that is similar as best-effort service under unloaded conditions, and this service is assured even the network is overloaded. The control-load service has no firm QoS guarantees and does not provide a worst-case delay bound, thus it can tolerate a certain amount of loss and delay, which is not suitable for the applications that require very low latency. On the other hand, the guaranteed service provides an assured level of bandwidth, a firm end-to-end delay bound and no queuing loss for data flow. In order to reserve resource for each flow, it is necessary to set aside certain resource such as buffer and bandwidth, and then create and maintain resource reservation on each link along the transport path. Therefore, resource reservation protocol (RSVP) [8] is introduced and works on top of IP for realtime traffic from source to destination. For end-to-end communications, hosts use RSVP to request a QoS level and routers use RSVP to deliver QoS requests to other routers along the path of data stream.

IntServ works well in end-to-end traffic. However it causes low efficiency in transmitting large number of data flows. Since it acts on each flow, it does not scale well due to high router memory requirement and large processing complexity with the increasing number of flows. Therefore, Differentiated Service (DiffServ) is introduced, which is simpler and more scalable [6]. Instead of per-flow process, DiffServ aims at per-class forwarding, so the amount of state information is proportional to the number of service classes rather than the number of flows. Each service class is an aggregation of many flows that have similar QoS requirement. The same class of flows are together maintained and forwarded router by router, and this efficient characteristic is called per-hop behavior (PHB). QoS management of DiffServ is mainly concerning at the edge and border routers, by which the complex processing is moved from the core of a network to the edge. Therefore, DiffServ does not guarantee end-to-end QoS [9].

In this research, we focus on end-to-end QoS management since network QoS management is beyond our study. For end-to-end QoS management, we only concern the QoS issues at the end points of sender and receiver, and we do not focus on the middle process in the routers. The primary target for end-to-end QoS management is to match the application QoS requirements with the network QoS provisions. The particular components of end-to-end QoS management such as congestion control and error control will be introduced in Chapter 2.

1.2 Problem Statement and Research Objective

In the previous section, we have pointed out that the major problem for multimedia over Internet is that Internet mainly provides best-effort services and it does not provide overall QoS in terms of bandwidth, delay, packet loss and etc. Although the combination of RTP, UDP and TCP can be used for multimedia communications, the QoS performance is far from satisfaction. In particular, there is no congestion control in RTP, which results in unfair competition with other TCP flows. In addition, there is also no error control in RTP and thus the multimedia communications on top is unreliable. Although video data can tolerate a certain degree of degradation, loss of data packets may cause large error propagation due to inherent video compression mechanisms.

Note that multimedia communications over IP should not just limit to wired networks. The underlying connections can consist of wireless links [10, 11]. In the wireless scenario, end-to-end QoS management is even more challenging. One of the main reasons is that in wireless mobile network there exists severe error corruption due to multipath and time-varying characteristics of wireless channels, which result in heavy bit errors and packet loss. The other problem is the bandwidth fluctuation and limitation due to time-varying and non-stationary physical channels [12]. Even 3G mobile network can only support a bit rate up to 384 kbps and there is no guarantee to maintain a stable

CHAPTER 1. INTRODUCTION

throughput. Moreover, wireless multimedia communications typically involves heterogeneous networks. For example, Fig. 1.2 shows the client-server architecture for a general wireless multimedia streaming system. As shown in the figure, the multimedia server is usually located in a wired line network such as Internet and wireless devices access the server through mobile access network. Since the data are delivered over both wired networks and wireless networks, the transmission may encounter both packet loss due to wired network congestion and burst errors due to wireless channel fading. In this case, the traditional Internet transport layer protocols such as TCP is not suitable any more since the basic assumption of TCP is that packet loss is only caused by network congestion. Hence new schemes are needed for the end-to-end QoS management in multimedia communications over IP with wireless links.

Figure 1.2: The architecture for a typical wireless multimedia streaming system.

Therefore, the objective of this research is to study, design and implement end-to-end QoS management schemes, particularly on congestion control and error control in the transportation layer with RTP, for multimedia communications over IP. In order to realize this goal, we will investigate the existing schemes of end-to-end QoS management for multimedia communications over IP, and then based on the investigation we will either propose new schemes or apply the existing schemes to practical systems. In addition, we also consider the case of containing wireless links in multimedia communications. Currently, we focus on converting the error models of wireless links to higher layers so

that the characteristics of underlying wireless links can be taken into account in the end-to-end QoS management.

1.3 Summary of Contributions

The major contributions of this thesis are list as follows:

- GEC conversion for wireless link modeling

In this research, we propose to model the wireless link layer by converting the widely used Gilbert-Elliott channel (GEC) at different levels. In particular, we develop an analytical procedure to derive the parameters of a desired symbol-level GEC model, from the parameters of a given bit-level GEC. Although we only present the algorithms of analytically converting GEC from bit-level to symbol-level, it can be applied to any problem of converting GEC from a level of shorter data segment to a level of longer data segment as long as the longer data segment is a multiple of the shorter data segment. Experimental results show that the theoretical analysis matches the simulation results very well.

- Optimal retransmission timeout (RTO) selection for end-to-end QoS

The RTO selection is a fundamental problem for any acknowledgment-based transmission. In this research, we model the statistical delay of IP network, and propose a novel algorithm for optimal RTO selection in delay-constrained multimedia communications based on the model. By appropriately choosing the RTO, we are able to adjust packet delay and transmission rate, and then achieve the optimal tradeoff between the error probability and the rate cost.

- Design and implementation of a RTP-based transportation platform for multimedia streaming over IP with end-to-end QoS management.

In particular, we develop the RTP-based transportation system based on Darwin Streaming Server and MPEG4IP. We integrate TCP friendly rate control (TFRC) algorithm [13] into the system for congestion control. We also implement a class-based fast retransmission scheme and an unequal erasure protection (UXP) scheme into the system for error control. A simple interface is defined for the application and the transportation system to exchange information for QoS management. Experimental results show that the developed system is able to provide end-to-end QoS management.

1.4 Outline

The remainder of this thesis is organized as follows. Chapter 2 gives a comprehensive literature review on RTP and QoS management in detail. Chapter 3 introduces a novel scheme for Gilbert-Elliott channel conversion from bit-level to packet-level. Chapter 4 describes an optimal retransmission timeout selection scheme for end-to-end QoS. Chapter 5 shows the design and implementation of a practical transportation system for multimedia streaming with QoS management. Finally, the conclusion remarks and future work are given in Chapter 6.

Chapter 2

Literature Review

As stated in Chapter 1, our objective is to study, design and implement end-to-end QoS management schemes, particularly on congestion control and error control in the transportation layer with RTP, for multimedia communications over IP. In this chapter, a literature review on the issues of RTP, RTCP, congestion control and error control. The chapter is organized as follows. Section 2.1 introduces the details of RTP, RTCP and the architecture of a RTP-based video transportation system. Section 2.2 describes some existing congestion control schemes. Section 2.3 reviews the current error control schemes to recover lost packets in the network. Finally, summary is given in Section 2.4.

2.1 RTP and RTCP

Real-time Transport Protocol (RTP) is an Internet standard protocol designed to provide end-to-end network transport functions suitable for real-time data transmission. RTP in itself, does not guarantee any kind of QoS or reliable delivery, but rather it provides basic functionalities that could be used by most real-time applications [3], the timestamp, sequence number and payload type identification, as shown in Table 1.1.

In multimedia communications, timestamp is absolutely needed for synchronization. The timestamp is set by the sender according to the instant that the first byte of the packet is sampled, and then transmitted to the receiver with media data. After receiving

CHAPTER 2. LITERATURE REVIEW

the data packets, the receiver uses the timestamp to reconstruct the original timing so that media can be played in the correct rate. Timestamp is also used to synchronize different streams, such as audio and video. Notice that RTP itself is not responsible for the synchronization.

As media packets may arrive at the receiver out of sequence, sequence number is included in RTP to reconstruct the received packets in their original sending order before further processing. It also helps the receiver to detect packet loss and distinguish whether the same packet has been received more than once. The initial value of sequence number for a multimedia stream is randomly generated, and the number is incremented by one for every RTP packet.

Payload type identification indicates the format of the RTP payload and enables the receiver to interpret the content of a RTP packet. Fig. 2.1 shows the format of RTP packet with fixed overhead.

In Fig. 2.1, the fixed RTP header occupies the first 12 octets. In particular, the version bits (V) identifies the version of RTP. Padding bit (P) indicates whether the packet contains appending octets at the end which are not part of the payload. If the extension bit (X) is set, the fixed header should be followed by a header extension. It is used for individual implementations with new payload-independent functions. Marker bit (M) is for indicating significant events such as frame boundaries in packet streams. PT denotes the payload type identification. CSRC count (CC) specifies the number of contributing source (CSRC) identifiers that follow the fixed header, and the maximum number is set to 15. CSRC identifier is used to identify the contributing sources for the payload contained in one packet. For example, in audio conferencing, CSRC identifiers indicate all the talkers whose speech has been combined to produce the outgoing packet.

Another important field in the RTP header is synchronization source (SSRC) identifier, which identifies the media source and distinguishes different synchronization sources.

CHAPTER 2. LITERATURE REVIEW

Note:
V: version; P: padding bit; X: extension bit; M: marker bit
CC: CSRC count; PT: payload type identifier
SSRC(32 bits): identifies data source
CSRC(32 bits): identifies contributing sources

Figure 2.1: The RTP packet format.

Different synchronization sources within the same RTP session must have different SSRC identifier. A RTP session is defined by the destination transport address, consisting of one IP address and a pair of ports for RTP and RTCP. Typically a multimedia session contains multiple RTP session and each type of media is carried in one RTP session. If one RTP session carries multiple streams with the same media type, each stream must be identified by a unique SSRC.

As mentioned in Chapter 1, the RTP Control Protocol (RTCP) is commonly used in conjunction with RTP for multimedia transportation. Similar to RTP, RTCP does not perform QoS management either, but it provides basic information for QoS management. RTCP packets include sender and receiver reports that announce network statistics which contain the number of packet sent, the number of packet lost, and inter-arrival jitter. These statistics enable end systems to monitor and manage some QoS performance. For

CHAPTER 2. LITERATURE REVIEW

example, the sender may use the network bandwidth and packet loss information to adjust its transmission rate, which is the key point of congestion control.

In RTCP, five packet types are defined: sender report packet (SR), receiver report packet (RR), source description packet (SDS), BYE packet (BYE) and the application-defined RTCP packet (APP). In particular, for each RTP stream in a multimedia session, the sender must create and transmit a SR. The SR contains the information of the RTP stream such as SSRC of the stream, timestamp when the SR is generated, the numbers of packets and bytes sent in the stream. Furthermore, sender reports can be used to synchronize multiple media streams. To respond the sender, the receiver sends receiver report as a feedback. For each RTP stream that a receiver receives, the receiver generates a RR. The RR includes the SSRC of the RTP stream, the fraction of packets lost within the stream, the last sequence number received, and the inter-arrival jitter. The SDS packet is for the sender to describe the source information of each RTP stream including sender's CNAME, email address, SSRC and etc. SDS provides a mapping between the SSRC and the host name. The BYE packet is used to inform all participants when a participant is leaving a session. The APP packet is designed for new applications and new features. More details can be found in [3].

Note that since RTCP packets are sent by the receivers to the corresponding senders, there is a problem arising, i.e., if RTCP packets are sent at a constant rate, the control traffic will grow linearly with the number of participants. Therefore, the transmission rate of RTCP must be scaled dynamically to avoid flooding the network. In practice, the bandwidth for RTCP is typically limited to 5% of the total session bandwidth. Among the bandwidth reserved for RTCP, 25% are allocated to sender reports and 75% to receiver reports. RTCP packets are usually recommended to be transmitted with an interval of not more than 5 seconds at the sender or receiver, except the first one should be sent at 2.5 seconds after the start up of an application [3].

CHAPTER 2. LITERATURE REVIEW

Figure 2.2: The architecture of a RTP/RTCP based video transmission system (copied from [14]).

After the describing properties of RTP and RTCP have been introduced, in the following we introduce a RTP-based end-to-end video transportation system proposed by Wu in [14]. In particular, for the sending part, video source is generated and compressed by MPEG-4 encoder, and then packetized into RTP packets. The RTP packets together

with the RTCP packets are passed down to UDP/IP layer for transport over the Internet [14]. At the receiver side, the received packets are firstly un-packed by UDP/IP layer, and then dispatched by filter and dispatcher to RTCP and RTCP analyzers, respectively. If there is packet loss detected, the error concealment component will be triggered. In the meantime, the RTCP analyzer unpacks the RTCP packets and sends the information to the feedback control protocol component [14]. One problem of the system architecture shown in Fig. 2.2 is that it does not include the feedback channel from receiver to sender, which should be considered in the design of multimedia system with QoS management.

2.2 Congestion Control

The target of congestion control is to reduce packet loss and delay by matching the application sending rate with the network bandwidth. Plenty of schemes have been proposed for congestion control. Among them, additive increase multiplicative decrease (AIMD) algorithm [15, 16, 17, 18, 19, 20] is the most well-known one, which in fact is adopted in TCP. The key principle of AIMD is to increase the sending rate by a constant if no loss occurs, and reduce the sending rate to a fraction if packet loss is detected. As we have mentioned, TCP is not suitable for multimedia transmission. However, since TCP is the dominant transportation protocol in Internet, any new end-to-end congestion control scheme for multimedia communications must be TCP-friendly [21, 22, 23, 24], which means the bandwidth competition among non-TCP flows and TCP flows should be fair. In general, congestion control can be classified into two categories, window-based and rate-based congestion control.

2.2.1 Window-based Congestion Control

In window-based congestion control, typically, each sender maintains two windows. One is the window granted by receiver and the other is the congestion window. The maximum

CHAPTER 2. LITERATURE REVIEW

number of bytes that the sender may transmit is the minimum of the two windows [2]. The congestion window is increased by a constant if no loss occurs, and would be reduced to a fraction if packet loss is detected. In the following, we use TCP Reno [25] as an example to introduce window-based congestion control algorithms in detail, because Reno is currently the most widely used scheme in all types of Internet applications.

1) *Slow Start Algorithm*: At the beginning of a connection, Reno sets the congestion window size to one and sends out the first packet. At each time an acknowledgement (ACK) is received, the window size is increased by one. Thus, congestion window size increases from one to two after the first round trip time (RTT), from two to four after the second RTT, and so on. This results in an exponential increase of the sending rate until packet loss is detected [26].

2) *Additive Increase Algorithm*: In addition to congestion window, there is a window threshold (slow-start threshold). When the congestion window is below the threshold, the slow start algorithm is used. However, when the congestion window size exceeds the threshold, the window increase rate is slowed down to avoid congestion. Specifically, it will be increased by one after each RTT rather than after the reception of each ACK.

3) *Multiplicative Decrease Algorithm*: When a packet loss is detected, the congestion window size is reset to one and the slow-start threshold is set to half of the current congestion window size. The slow start phase is initiated again. Fig. 2.3 describes such a process of slow start, additive increase and multiplicative decrease.

For wireless Internet access, applying Reno will cause the problem of wasting network bandwidth. This is because Reno assumes all the packet loss is caused by network congestion while in wireless Internet access the packet loss may be resulted from wireless channel impairment. Under the cases of wireless loss, depressing the network throughput has no use to prevent packet loss.

In order to mitigate this drawback, TCP Veno [19] has been proposed, which adopts a network state judgement scheme to distinguish the congestion loss and wireless loss [18],

CHAPTER 2. LITERATURE REVIEW

Figure 2.3: Window-based congestion control in TCP Reno.

and provides different congestion control mechanisms for them. In particular, if packet loss is detected in congestion state, Veno assumes the loss is due to congestion; otherwise, it is considered as wireless loss. Based on such differentiation, Veno refines the AIMD procedure of Reno, making the alteration of window size more smoothly and holding a higher and steadier throughput. Experimental results shown that in a typical wireless access network with 1% packet loss rate, Veno achieves a throughput improvement up to 80%.

2.2.2 Rate-based Congestion Control

Although AIMD based window congestion control is very effective at rapidly utilizing network bandwidth when it becomes available, it is not suitable for multimedia applications, which require smooth sending rates. Here comes the need for rate-based congestion control.

CHAPTER 2. LITERATURE REVIEW

Rate-based congestion control schemes are those which do not use a congestion window to control the amount of data in the network; instead, they choose sending rate based on current network parameters which are appropriate for the application [27]. In order to be TCP-friendly, the maximum of sending rate must be not higher than that of a TCP connection with the same path. When there is no congestion, the sender may keep their desired maximum sending rate. While sending data, the sender should monitor the overall packet loss. High levels of packet loss may force the sender to reduce the sending rate. If the loss level in the network decreases, the sending rate may be increased, but should not exceed the limits of the maximum rate. After modifying its sending rate, prior to performing additional increase or decrease to their sending rate in a new round, the sender must wait for at least one RTT to obtain new RTT and loss rate. This will prevent the sender from over-reacting in the time of congestion [27].

Most rate-based congestion control schemes have some latitude in their choice of data rates. In [28], the authors observed the exponential behavior of round trip delay, and proposed an exponential increase to overcome the problem of using constant additive increase, which improves the smoothness and the convergence of the congestion control. In [16], some design options for specifying AIMD adaptation were proposed to perform dynamic additive increase and decrease. Application level QoS on TCP-friendly video transferring was considered in [21], and the congestion control for real-time packet video using prediction was adopted in [23]. Although these schemes have their specific points, they all take into account the effects of packet loss on the bandwidth, which is usually calculated by [27]:

$$B = \frac{k \cdot MTU}{RTT \cdot \sqrt{p}}, \quad (2.1)$$

where MTU is the maximum packet size used on the connection, RTT is the mean round trip time, and p is the mean packet loss rate. k is a constant value of 1.22 or 1.3 depending

CHAPTER 2. LITERATURE REVIEW

on whether delayed or non-delayed acknowledgement is used. This simple throughput estimation formula is based on the assumption that the retransmission timeout never happens and packet loss occurs randomly, which does not correctly reflect the reality of current Internet, i.e., retransmission occurs as results of network congestion and packet loss may not appear at random [29]. Therefore, the bandwidth estimation should take into account the retransmission timeout as [13]:

$$B = \frac{MTU}{RTT\sqrt{\frac{2p}{3}} + RTO\sqrt{\frac{27p}{8}}p(1 + 32p^2)}, \quad (2.2)$$

where RTO denotes the retransmission timeout and p turns into loss event rate, which is more accurate to reflect long-term network properties than packet loss rate. Eqn. (2.2) is widely used in TCP-friendly bandwidth estimation and it provides the fundamental for many other rate-based congestion control schemes, such as in [20], [30] and [31].

Moreover, congestion control can work together with error control to achieve better performance of QoS [22, 32]. In the next section, we will introduce the error control component for end-to-end QoS management.

2.3 Error Control

In multimedia communications, media data may be altered or lost during transmission due to errors. The transmission errors can be roughly classified into two categories: bit errors and erasure errors. Bit errors are caused by the interference of physical channel noise, which results in bit inversion, bit insertion, and bit deletion. Erasure errors are due to packet loss caused by congestion. Therefore, error control should be included as a component of QoS management in the design of real-time multimedia protocol to combat with errors [33].

In [33] and [34], Wang summarized the error control techniques in current research as four categories: error control in source encoder, error control in source decoder, encoder

CHAPTER 2. LITERATURE REVIEW

and decoder interactive error control, and transport-level error control. Error concealment is not strictly an error control scheme because it does not recover the lost data, but rather reduces the visual effect due to the lost data by creating an approximation based on the information before or after the loss. In this thesis, we focus on the transport-level end-to-end error control, without considering error concealment in the source encoder and decoder.

Basically, there are 4 options to handle packets with errors in transport-level.

1) *No checking*: Receiver forwards data packets to the application layer without checking whether the packets contain any error.

2) *Discard*: Receiver directly discards any packets with errors.

3) *Receive*: Receiver delivers and flags the packets that have errors to the application.

4) *Correct*: Receiver corrects the errors or recovers the lost data by auto repeat request (ARQ) or forward error correction (FEC), which are the pivots of this section.

2.3.1 Forward Error Correction (FEC)

The error control techniques can be mainly classified into two categories: forward error correction (FEC) and automatic repeat request (ARQ) [35]. ARQ needs a feedback channel to send retransmission requests while FEC does not have such requirement.

FEC can be used to repair the damage to media streams due to packet loss or bit errors. It employs error correcting codes as redundant parity to add into information packets before they are transmitted, where the redundancy is used by the receiver to detect and correct errors [36].

There are two basic types of error correcting codes: block codes and convolutional codes. Block codes are generated from the information data, and appended with the information data to form a codeword. Binary Bose-Chaudhuri-Hocquenghem (BCH) codes and non-binary Reed-solomon (RS) codes are two widely used linear cyclic block

CHAPTER 2. LITERATURE REVIEW

codes. Both BCH and RS codes have efficient coding and decoding schemes, and RS codes have optimal error correction capability given a fixed amount of redundancy [35].

Figure 2.4: Structure of a RS codeword.

Fig. 2.4 shows the structure of a RS codeword. An (n, k) RS code has the following parameters:

- Block length: n ,
- number of information symbols: k ,
- number of parity-check symbols: $n - k = 2t$,
- minimum distance: $d = t + 1$,

An (n, k) RS code can correct up to t errors. n is usually chosen to be $2^m - 1$, where m denotes the number of bits in one symbol. Sometimes, the block codes of desirable natural length or suitable number of information data may not exist. In this case, block codes can be shortened by setting a selected number of information data to zero at the encoder. The zero data would not be transmitted, and would be re-inserted at the decoder. For example, a $(255, 223)$ RS code can be shortened to $(200, 168)$. The encoder takes a block of 168 data symbols, adds 55 zeros, creating a $(255, 223)$ codeword, and then transmitting only the 168 data symbols and 32 parity symbols. At the receiver, the 55 zeros will be re-inserted to the shorten codes to decode the codeword. For RS codes, the minimum distance is unchanged after shortening.

Unlike block codes, which are memoryless since successive information blocks are coded independently, convolutional codes are memorial because the coding of current data is a function of the previous data [36]. An (n, k, m) convolutional code generates n

CHAPTER 2. LITERATURE REVIEW

codeword symbols for every k information symbols, where coding rate is k/n and m is the memory of encoder. In order to have flexible error correction abilities and adaptive coding rate, puncturing codes were proposed, without changing the basic structure of the convolutional codes. The rate-compatible punctured convolutional (RCPC) codes [37] are the most popular convolutional punctured codes. In RCPC coding scheme, all high rate codes are used by the low rate codes, which means high rate codes are embedded into the low rate codes of the family. When errors in the higher rate codes can not be corrected, instead of retransmitting all the code bits of a completely different low rate codes, the lower rate codes should utilize the bits already received. In this way, it only needs the additional incremental redundant bits. Furthermore, there are some new error control techniques recently coming out based on convolutional codes. For instance, a far end error decoder (FEED) algorithm is proposed in [38] for wireless progressive video transmission. FEED is based on sequential decoding, and it aims on delaying the first error in the decoded frame as far as possible [39].

Both block codes and convolutional codes can be combined with interleaving to mitigate the effects of error bursts. For symbol-level block interleaving, $N \times L$ symbols are loaded into the interleaver row by row, where N is the codeword length and L is the interleaving depth [40]. The value of N is typically determined by the constraint of delay, and the data dimension is consistent to the FEC coding direction, shown as the top of Fig. 2.5. After encoded by FEC, symbols are read out from the interleaver column by column. When data reach the receiver, reverse process will be performed in a deinterleaver. Interleaving does not decrease the error rate but it is successful in decreasing the number of errors in each codeword. Moreover, interleaving can reduce the correlations among the bursty errors caused by network degradation.

A new interleaving scheme called pre-interleaving was proposed in [41] and [42]. It is designed for progressive image transmission. The key idea is to delay the occurrence of

Figure 2.5: Comparison between interleaving and pre-interleaving (Copied from [41]). Top: original interleaving. Bottom: pre-interleaving.

the first unrecoverable error since the performance of the progressive image transmission is actually determined by the location of the first unrecoverable error. Different from original interleaving, in pre-interleaving the data are put into the interleaver column by column and the data dimension is orthogonal to the FEC coding direction. Fig. 2.5 shows the comparison between original interleaving and pre-interleaving, in which the first row contains header information and needs higher-level protection. By using pre-interleaving, the performance gain will be even more significant in the cases of small values of N and

large values of L such as Internet packet sizes [42].

2.3.2 Auto Repeat reQuest (ARQ)

FEC has been widely used in transmitting video in the Internet [43, 44], on the other hand, ARQ is also effective to recover errors and packet loss relying on retransmission.

ARQ is briefly operating as follows: The sender transmits the packets to the receiver and sets a timer for each packet, and the receiver sends ACKs for every received packet. The Packets would not be acknowledged until they are successfully received. After a predetermined time interval such as retransmission timeout, the unacknowledged packets are assumed to be lost and then will be retransmitted. In some cases, negative acknowledgements (NAKs) are transmitted by the receiver for every packet received in error. If NAKs are employed, a packet is retransmitted following the receipt of a negative acknowledgement.

There are three basic ARQ protocols: Stop-and-Wait, Selective-Repeat and Go-Back-N ARQ [45]. When using Stop-and-Wait (SW) ARQ, the sender waits for an ACK after transmitting a packet. Once ACK has been received, the next packet is transmitted. However, if no ACKs arrive until timeout expires, the sender must retransmit the packet. Since in SW the sender spends plenty of time on waiting for ACKs, throughput is not fully utilized.

In Selective-Repeat (SR) ARQ, packets are transmitted continuously and the receiver sends ACKs back for each successfully received packet. If an ACK is not received for a packet before the expiration of the timeout, the packet is retransmitted. Moreover in SR, packets can be accepted out of sequence, thus the out-of-order packets have to be buffered and sequenced before transmission [36]. Since the sender can continuously transmit packets without waiting, SR acts more efficiently than SW.

In Go-Back-N (GBN) ARQ, packets are transmitted as in SR. However, the receiver accepts packets only in the order of which they were transmitted. The packets received

out of sequence are discarded and not acknowledged. After the timeout expires, the sender retransmits the unacknowledged packet and the all following packets. Hence, the sender needn't buffer the out-of-sequence packets [36]. When each timeout occurs, all unacknowledged packets are to be retransmitted.

2.3.3 Hybrid ARQ

Of these two error control techniques, ARQ has been shown to be more effective than FEC but with longer delay due to retransmission. Thus, ARQ may not be suitable for delay-constrained real-time applications. As a consequence, hybrid ARQ is proposed which combines ARQ with FEC codes [46, 47] to make a good compromise.

There are two types of hybrid ARQ schemes, type-I and type-II. Type-I hybrid ARQ includes FEC parity codes for both error detection and error correction in every transmitted packet. If the number of erroneous symbols in a received packet is within the error correction capability of the FEC codes, the errors can be successfully corrected. If the errors are uncorrected, the receiver will discard the whole packet and request a retransmission, then the sender will retransmit the same packet of codeword. If the retransmitted packet still arrives with uncorrectable errors, the process continues until the codeword is successfully accepted or the maximum allowed retransmission attempts have been exhausted. In [48], a class of adaptive Type-I hybrid ARQ schemes are proposed to improve the process, by adaptively increasing and decreasing the error correction capability according to the number of retransmissions.

In type-II hybrid ARQ schemes, any codewords that could not be successfully decoded are saved, and the receiver simultaneously requests for a retransmission. Since punctured convolutional codes are used, the sender will not retransmit the same codeword but transmit the incremental redundant codes. When the retransmitted redundant codes reach the receiver, it will be combined with the saved codeword to form a new

CHAPTER 2. LITERATURE REVIEW

codeword, which has lower coding rate and higher error correction capability. This process is repeated until the codeword can be successfully decoded. In [49], Q. Zhang and A. kassam proposed a novel hybrid ARQ scheme with selective combining for wireless fading channels, where all the possible combinations are considered by exploiting the flexibility of RCPC codes.

Figure 2.6: (a):Common type-II hybrid ARQ. (b) Type-II hybrid ARQ with selective combining.

According to the characteristics of RCPC code, the hybrid ARQ with selective combining works well in wireless fading channel. As Fig. 2.6 describes, if receiver can not correct the errors in packet of code A, it sends back a negative acknowledgement (NAK) to sender. Then redundancy of code B is transmitted by sender and combined with A at receiver. If errors are still uncorrectable in the combined codes, more redundancy of code C is required. In case of fading channel, errors are bursty, combination of AC or BC codeword may have fewer errors than combination of ABC, because A or B may be corrupted by the bursty errors. Such is so-called selective combining, of which the main idea is to deploy the best combination for error correction. The whole process gets along until all the redundancies are sent.

CHAPTER 2. LITERATURE REVIEW

2.4 Summary

In this chapter, we have studied the existing multimedia transport protocols and QoS management techniques. An overview of RTP and RTCP protocols are given and the features of them are highlighted. Moreover, congestion control and error control schemes have been classified and introduced. The literature review builds the foundation for our further research works.

Chapter 3

GEC Conversion for Wireless Link Modeling

In the case of multimedia over IP with wireless links, it is necessary to characterize wireless network performance in terms of QoS parameters by studying the communication mechanisms in the lower layers of wireless networks. In this chapter, we show our first attempt in wireless link modeling through GEC conversions. The chapter is organized as follows. Section 3.1 introduces the related work on GEC generation and applications, and points out the GEC conversion problem. Section 3.2 describes the proposed analytical conversion algorithms. Extensive experimental results are shown in Section 3.3 and finally conclusion remarks are given in Section 3.4.

3.1 Related Work

The Gilbert-Elliott Channel (GEC) [50, 51] has been widely used in numerous research work due to its simplicity and able to model the bursty patterns of wireless channel errors and Internet packet loss [52, 53]. In [40, 54], bit-level GEC models were constructed from practical land mobile channels. In [55, 56], the performance of Reed-Solomon block codes was studied based on symbol-level GEC models. Bit-level and symbol-level GEC models were also used in [57] and [58], respectively, for image and video transmission. As IP

could likely traverse over the wireless networks, packet-level GEC models were employed in [52, 53] to simulate bursty packet loss patterns in Internet and wireless networks, respectively. In all these works, the authors either assumed GEC models available in a particular level or directly modelling GEC into a level they wanted. However, in many practical situations, we may only have some well modeled GECs in a level of shorter data segment (say bit-level) while GECs in a level of longer data segment (say packet-level) are desired. Therefore, a question is arising, i.e., whether we can have a GEC in a level of longer data segment given the practical channel conforms to a GEC in a level of shorter data segment.

It is not difficult to confirm that the channel will still remain a GEC in a level of longer data segment as long as the longer data segment is a multiple of the shorter data segment. This can be explained as follows: for a GEC in a level of shorter data segment, the corresponding channel in a level of longer data segment is also a two-state Markov channel since, in the level of longer data segment, the channel state of a current longer data segment only depends on the channel state of the previous longer data segment. Then, the next question is how to convert GEC from a level of shorter data segment to a longer one. To the knowledge of the authors, little attention has been paid on this problem.

3.2 Converting GEC to A Level of Longer Data Segment

Fig. 3.1 shows the general GEC model, which has two states: Good state and Bad state. The parameters of GEC include α , β , $P(G)$, $P(B)$, e_g , e_b , ϵ , λ , where α denotes the transition probability from Bad state to Good state while β denotes the reverse transition probability, $P(G)$ and $P(B)$ are the probabilities that channel stays in Good state and Bad state, respectively, e_g and e_b are the bit error rates (BERs) at Good state

CHAPTER 3. GEC CONVERSION FOR WIRELESS LINK MODELING

and Bad state, respectively, ϵ denotes the average BER and λ denotes the average burst length. In the special case of $e_g = 0$ and $e_b = 1$, the GEC channel model is referred as the simplified GEC model [56].

Figure 3.1: Gilbert-Elliott Channel.

Notice that $P(G)$, $P(B)$, ϵ , λ can be easily derived from α , β , e_g , e_b . Given the simplified GEC model, there are

$$P(B) = \frac{\beta}{\alpha + \beta}, \quad \epsilon = P(B),$$

$$P(G) = \frac{\alpha}{\alpha + \beta}, \quad \lambda = \frac{1}{\alpha}.$$

In the following, we will show how to convert GEC from bit-level to symbol-level. Notice that a Good state symbol is defined as a symbol without any Bad state in bit-level while others are belonging to Bad state symbols. We will study the conversion of simplified GECs first and then extend it to general GECs.

3.2.1 Conversion of Simplified GECs

The conversion of simplified GECs can be summarized as: given a bit-level simplified GEC model with the parameters of (α, β) , how to convert the model analytically into a symbol-level simplified GEC with the parameters of (α', β') . It is clear that, for symbol-level simplified GECs, a Good state symbol does not contain any bit error while a Bad state symbol must contain some bit errors. Given the previous symbol is in Good state,

CHAPTER 3. GEC CONVERSION FOR WIRELESS LINK MODELING

the probability of current symbol to be error-free is $(1 - \beta)^q$, where q denotes the number of bits in each symbol. According to the definition, β' can be calculated as

$$\beta' = 1 - (1 - \beta)^q. \quad (3.1)$$

It is not easy to derive α' since the last bit of the previous Bad state symbol can come from both Good state and Bad state in bit-level. When the last bit is from Good state, the probability of current symbol to be error-free is $(1 - \beta)^q$ while it is $\alpha(1 - \beta)^{q-1}$ when the last bit is from Bad state. Hence, α' can be derived as

$$\alpha' = \frac{P_q(G)}{P_q(G) + P_q(B)}(1 - \beta)^q + \frac{P_q(B)}{P_q(G) + P_q(B)}\alpha(1 - \beta)^{q-1}, \quad (3.2)$$

where $P_q(G)$ is the probability that the previous symbol is an error symbol and the last bit of the previous symbol is in Good state. $P_q(B)$ is similar except that the last bit of the previous symbol is in Bad state. $P_q(G) + P_q(B)$ represents the probability that the previous symbol is an erroneous symbol. $P_q(G)$ and $P_q(B)$ can be computed by using the recursive method proposed in [56], which is not convenient for direct analysis. To reduce the computation complexity, we derive closed-form formulas to compute $P_q(G)$ and $P_q(B)$ based on the assumption that all errors occurring in a symbol are consecutive. This is reasonable for the highly bursty wireless channels and a small value of q (say $q = 8$). In particular, $P_q(G)$ and $P_q(B)$ can be written as

$$P_q(G) = \sum_{k=1}^{q-1} \frac{\beta}{\alpha + \beta} (1 - \alpha)^{k-1} \alpha (1 - \beta)^{q-k-1} + \sum_{k=1}^{q-2} (q - k - 1) \frac{\alpha}{\alpha + \beta} \beta (1 - \alpha)^{k-1} \alpha (1 - \beta)^{q-k-2}, \quad (3.3)$$

$$P_q(B) = \frac{\beta}{\alpha + \beta} (1 - \alpha)^{q-1} + \sum_{k=1}^{q-1} \frac{\alpha}{\alpha + \beta} (1 - \beta)^{q-k-1} \beta (1 - \alpha)^{k-1}, \quad (3.4)$$

where k is the number of consecutive errors. In Eqn. (3.3), the first term and the second term are the probabilities of error bursts in the beginning and the middle of a symbol, respectively, as shown in Fig. 3.2 (a) and (b). In Eqn. (3.4), the first term is

Figure 3.2: (a) An error burst at the beginning of a symbol; (b) An error burst in the middle of a symbol; (c) An error burst at the end of a symbol.

the probability that all the bits of one symbol are corrupted while the second term is the probability of an error burst at the end of a symbol, as shown in Fig. 3.2 (c).

The simplified algorithm shown in Eqn. (3.3) and Eqn. (3.3) works well in the cases of small q values. However, it is not suitable in the cases of large q values since the previous assumption that all bit errors occurring in one symbol are consecutive is not reasonable any more. Experimental results show that using the simplified algorithm results in the mismatch error between simulation results and theoretic results about 1.5% when $q = 8$, 16% when $q = 16$ and 20% when $q = 32$. On the other hand, using the recursive method is also not appropriate since the executing time will increase exponentially with the increase of q . Therefore, we propose to calculate the probabilities in the cases of large q as

$$P_q(G) = \sum_{d=1}^{q-1} \left(\frac{\alpha}{\alpha + \beta} P_q(d|GG) + \frac{\beta}{\alpha + \beta} P_q(d|BG) \right), \quad (3.5)$$

$$P_q(B) = \frac{\beta}{\alpha + \beta} (1 - \alpha)^{q-1} + \sum_{d=1}^{q-1} \left(\frac{\alpha}{\alpha + \beta} P_q(d|GB) + \frac{\beta}{\alpha + \beta} P_q(d|BB) \right), \quad (3.6)$$

where $P_q(d|GB)$ is the probability of having d Bad bits in one symbol given that the first bit in the symbol is in Good state and the last bit is in Bad state. Other probabilities are defined accordingly, and their detailed derivation can be found in [40].

3.2.2 Conversion of General GECs

The conversion of general GECs can be summarized as: given a general bit-level GEC with the parameters of $(\alpha, \beta, e_g, e_b)$, how to convert the model analytically into a general symbol-level GEC with the parameters of $(\alpha', \beta', e'_g, e'_b)$. Although, comparing with the conversion of simplified GECs, the conversion of general GECs has two extra parameters: e_g and e_b , in the following, we will show that this conversion problem can be solved by the similar approach.

According to the definition of Good state in symbol-level, a Good state symbol contains all the Good state bits while it may have corrupted bits. On the other hand, according to the definition of Bad state in symbol-level, a Bad state symbol must contain at least one Bad state bit although it may be error free. Based on these definitions, α' and β' can be derived in the same way as that for the conversion of simplified GECs.

For e'_g , it can easily be derived as

$$e'_g = 1 - (1 - e_g)^q. \quad (3.7)$$

e'_b can be calculated as

$$e'_b = 1 - \frac{1}{\tilde{T}} \sum_{d=1}^q T(d)(1 - e_b)^d(1 - e_g)^{q-d}, \quad (3.8)$$

where $T(d)$ is the probability of having d Bad state bits in a symbol, $\tilde{T} = \sum_{d=1}^q T(d)$ and $\frac{1}{\tilde{T}} \sum_{d=1}^q T(d)(1 - e_b)^d(1 - e_g)^{q-d}$ is the probability of having no bit error in a Bad state symbol. Now, the key is how to compute $T(d)$. One simple way is to use the recursive method, where the executing time increases exponentially with the increase of q . Alternatively, in the cases of small q values, similar to the assumption made in the conversion of simplified GECs, we assume all the Bad state bits in a symbol are consecutive. Based on this assumption, we can derive a close-form formula for computing $T(d)$ in the cases of small q values as

$$T(d) = \begin{cases} \frac{\beta}{\alpha+\beta}(1-\alpha)^{d-1}\alpha(1-\beta)^{q-d-1} + (q-d-1)\frac{\alpha}{\alpha+\beta}\beta(1-\alpha)^{d-1}\alpha(1-\beta)^{q-d-2} \\ \quad + \frac{\alpha}{\alpha+\beta}(1-\alpha)^{d-1}\beta(1-\beta)^{q-d-1} & \text{if } 1 \leq d \leq q-1 \\ \frac{\beta}{\alpha+\beta}(1-\alpha)^{q-1} & \text{if } d = q \end{cases} \quad (3.9)$$

Similarly, we can also develop the formula for the cases of large q values based on Eqn. (3.5) and Eqn. (3.6).

3.3 Experimental Results

Although we have proposed the approaches basing on converting bit-level GECs to symbol-level GECs, the proposed approaches can also be applied to the packet-level GECs. Experiments are performed to show our correctness.

3.3.1 Results of Converting Simplified GECs

To verify the validity of the proposed conversion algorithms, simulations are conducted on bit-level simplified GECs with different (α, β) pairs, and the symbol-level parameters (ϵ', λ') are measured. Then, the measured parameters are compared with the theoretic values of (ϵ', λ') computed from the converted symbol-level parameters (α', β') , where $\epsilon' = \frac{\beta'}{\alpha'+\beta'}$ and $\lambda' = \frac{1}{\alpha'}$. The comparison is shown in Table 3.1 and Table 3.2, where Scheme A is the conversion algorithm using the recursive method while Scheme B is that using the derived close-form formulas. It can be seen that both theoretic results consistently match the simulation results very well and the mismatch errors are always less than 2%.

3.3.2 Results of Converting General GECs

Following the previous experiments, we conduct simulations on bit-level general GECs with different parameters of $(\alpha, \beta, e_g, e_b)$. Then, we compare the simulation results and

CHAPTER 3. GEC CONVERSION FOR WIRELESS LINK MODELING

Table 3.1: Symbol error rate for simplified GEC.

ϵ'		$\alpha = 0.04$	$\alpha = 0.03$	$\alpha = 0.02$	$\alpha = 0.01$
$\beta = 0.003$	Simulation	0.09009	0.11108	0.15026	0.24948
	Scheme A	0.09066	0.11109	0.14918	0.24699
	Scheme B	0.08903	0.10974	0.14845	0.24671
$\beta = 0.002$	Simulation	0.06173	0.07644	0.10485	0.17956
	Scheme A	0.06198	0.07636	0.10407	0.17846
	Scheme B	0.06083	0.07550	0.10352	0.17823
$\beta = 0.001$	Simulation	0.03136	0.03904	0.05414	0.09696
	Scheme A	0.03180	0.03947	0.05456	0.09739
	Scheme B	0.03119	0.03900	0.05425	0.09724

Table 3.2: Average symbol-level burst length for simplified GEC.

λ'		$\alpha = 0.04$	$\alpha = 0.03$	$\alpha = 0.02$	$\alpha = 0.01$
$\beta = 0.003$	Simulation	4.139	5.223	7.390	13.891
	Scheme A	4.198	5.253	7.383	13.811
	Scheme B	4.115	5.190	7.340	13.790
$\beta = 0.002$	Simulation	4.110	5.170	7.310	13.683
	Scheme A	4.159	5.203	7.311	13.672
	Scheme B	4.076	5.140	7.268	13.650
$\beta = 0.001$	Simulation	4.048	5.079	7.153	13.336
	Scheme A	4.120	5.154	7.239	13.534
	Scheme B	4.038	5.090	7.196	13.512

theoretic results on the symbol-level parameters (ϵ', λ') . We can draw a similar conclusion as that for the conversion of simplified GECs.

In order to further illustrate the correctness of the proposed conversion algorithms, we study the performance of the popular non-binary channel codes, Reed-Solomon (RS) block codes with parameters (N, K) , 8 bits/symbol and symbol error-correcting ability $t = \lfloor \frac{(N-K)}{2} \rfloor$. In particular, we conduct experiments on different bit-level general GECs with different RS codes and then measure the probabilities of codeword error P_{cw} , where a codeword is a N -symbol channel block. On the other hand, based on the converted

CHAPTER 3. GEC CONVERSION FOR WIRELESS LINK MODELING

parameters $(\alpha', \beta', e'_g, e'_b)$ and the formulas provided in [40], we can theoretically calculate P_{cw} . Experimental results are shown in Fig. 3.3 and Fig. 3.4, which demonstrate the excellent match between the simulation results and the theoretical results. Notice that the theoretical results are obtained by using the derived close-form formulas. We have also tried the recursive approach, which obtains almost the same performance as the other one.

Figure 3.3: The Probabilities of codeword error for RS codes of length 31 transmitted through a general GEC with $\alpha=0.01$, $\beta=0.003$, $e_g=0.001$, $e_b=0.12$.

3.3.3 Results of GEC Conversion for Wireless Link Modeling

In order to study the end-to-end QoS management in wireless multimedia communications, we need to characterize the wireless network performance in terms of QoS parameters from an end point of view. Hence, it is necessary to learn the properties of lower layers in wireless networks. Since physical channels of wireless mobile networks can be

CHAPTER 3. GEC CONVERSION FOR WIRELESS LINK MODELING

Figure 3.4: The Probabilities of codeword error for RS codes of length 31 transmitted through a general GEC with $\alpha=0.04$, $\beta=0.001$, $eg=0.001$, $eb=0.12$.

well modeled by GECs as mentioned before, the next step would be wireless link modeling. In the following, we will show how to generate packet-level GECs from symbol-level GECs. It is essentially a conversion example in the cases of large q values.

Simulations are conducted on symbol-level simplified GECs with different (ϵ, λ) pairs, and packet loss rates in the wireless link layer are measured. Then, the measured packet loss rates are compared with the theoretical packet loss rates obtained from the proposed GEC conversion algorithms. In particular, the length of a link-layer packet is 512 bytes and the length of a symbol is one byte. In this case, since the value of q is very large (equal to 512), we need to use the formulas of Eqn. (3.5) and Eqn. (3.6) for conversion. Although we can directly convert GECs from a level of 1 byte to a level of 512 byte, in fact, in this experiment we use conversion twice, i.e., from 1 byte level to 32 byte level and then from 32 byte level to 512 byte level, in order to further demonstrate the accuracy of the proposed conversion algorithms even after multiple conversions. Experimental results

are shown in Table 3.3. It can be seen that the average mismatch error is smaller than 1%.

Table 3.3: Packet loss rate on IP layer.

PLR		$\lambda = 16$	$\lambda = 32$	$\lambda = 64$	$\lambda = 128$
$\epsilon = 0.002$	Simulation	0.06386	0.03355	0.01751	0.00988
	Theoretical	0.06388	0.03343	0.01784	0.00995
$\epsilon = 0.005$	Simulation	0.15312	0.08171	0.04433	0.02464
	Theoretical	0.15255	0.08173	0.04413	0.02476
$\epsilon = 0.01$	Simulation	0.28366	0.15810	0.08665	0.04906
	Theoretical	0.28305	0.15750	0.08671	0.04913

3.4 Summary

In this chapter, we have proposed the approaches which are able to theoretically convert any GEC from a level of shorter data segment to a level of longer data segment as long as the longer data segment is a multiple of the shorter data segment. The validity of the proposed conversion algorithms has been demonstrated in the experiments. We have applied the proposed GEC conversions for wireless link modeling. This will benefit the performance study of wireless link-layer under different error control schemes such as ARQ, FEC and hybrid ARQ. The resulted link-layer characteristics will further make contribution to the upper layer performance characterization and finally to be used in the study of end-to-end QoS management.

Chapter 4

Optimal Retransmission Timeout (RTO) Selection for End-to-End QoS

In this chapter, we study an issue in end-to-end QoS management, i.e., optimal RTO selection for delay-constrained multimedia communications. This work is quite general and can be extended to different transportation systems. The chapter is organized as follows. Section 4.1 gives some background on RTO selection. Section 4.2 describes the need and the proposed algorithm for optimal RTO selection. Preliminary results are shown in Section 4.3. Finally conclusion remarks and discussions are given in Section 4.4.

4.1 Related Work

In end-to-end communications, many transport protocols rely on retransmission to recover lost packets [59, 60]. Although retransmission is generally considered inappropriate for continuous multimedia applications due to round trip delay, it may be feasible in many cases where the round trip delay is relatively small (e.g., LANs and MANs). Moreover, the playout buffer increases the time available for retransmission. Retransmission-based error recovery is still attractive and used in high-speed protocols [61].

For these retransmission-based transport protocols, such as TCP, a fundamental question is how long a sender should wait until retransmitting in the absence of receiving an

CHAPTER 4. OPTIMAL RETRANSMISSION TIMEOUT (RTO) SELECTION FOR END-TO-END QoS

acknowledgment (ACK). The waiting time is defined as retransmission timeout (RTO).

Traditionally, RTO is selected according to the predicted value of the next round-trip time (RTT), such as

$$RTO = \beta RTT_t, \quad (4.1)$$

where β is a constant, usually equal to 2. RTT_t is commonly estimated based on the receiver's feedback packets. A typical RTT estimator commonly used in previous work [62, 63] is

$$RTT_t = \alpha RTT_{t-1} + (1 - \alpha)M_t, \quad (4.2)$$

where RTT_t is the lately estimated RTT, RTT_{t-1} is the previous estimated RTT, M_t is the RTT sample measured from the last ACK, and α is a constant with a default value of 0.9. By noticing the RTO selection based on Eqn. (4.1) does not take into account of the large fluctuation of RTT, Jacobson [17] further proposed to update RTO as

$$RTO = RTT_t + 4 \cdot Var, \quad (4.3)$$

where Var is the average deviation of RTT. Although both of the RTO selection schemes work fine for conventional data communications, it may not be able to achieve optimal performance for delay-constrained multimedia communications since they do not consider the delay constraint and packet loss rates provided by communication networks.

To overcome this problem, some research works have been addressed on time-sensitive transmission, such as time-lined TCP [64] and soft ARQ [65]. Most of them focused on designing delay-aware congestion control schemes.

4.2 Optimal RTO selection

Recently, Chou et al. [66] proposed a framework for streaming packetized media over a lossy packet network in a rate-distortion optimized way. The proposed framework

CHAPTER 4. OPTIMAL RETRANSMISSION TIMEOUT (RTO) SELECTION FOR END-TO-END QoS

is able to minimize the end-to-end distortion under a rate constraint by choosing the right packets to transmit at a given transmission opportunity. Since in this research we assume all the packets have the same distortion importance and delay constraint, the problem of optimally transmitting a group of packets can be simplified as that of optimally transmitting one packet. The performance of transmitting one packet can be described by two parameters: the error probability ϵ and the rate cost ρ . ϵ is defined as the probability that the data unit does not arrive at its destination by the decoder time stamp (DTS), which is the delay deadline. ρ is defined as the expected number of bytes transmitting per source byte [66]. In the following, we will introduce three common scenarios of delivering one packet described in [66] which involve different end-to-end QoS management.

The simplest scenario is that the sender has only one choice at time t_0 , i.e., transmitting a data packet or not. If the data unit is not transmitted, the error probability will be one and the rate cost will be zero. On the other hand, if the data unit is transmitted, the error probability will be $\epsilon = P\{FTT > DTS - t_0\}$, where FTT is the forward trip time, while the cost per source byte is one. Based on this simplest scenario, the more complicate scenarios can be further studied.

The second scenario is to adopt forward error correction (FEC) codes for packet transmission. For example, the original packet is grouped together with other $k - 1$ packets, and an (n, k) Reed-Solomon code is applied to produce $n - k$ parity units. All these n packets are transmitted at the same time t_0 . The original data packet can not be recovered at the receiver by the time of DTS only if the packet itself is late or lost, which happens with probability $\epsilon = P\{FTT > DTS - t_0\}$, and at least $(n - k)$ of the other $n - 1$ packets are also late or lost, which happens with probability of

$$f_{n,k} = \sum_{i=n-k}^{n-1} \binom{n-1}{i} \epsilon^i (1 - \epsilon)^{n-1-i}. \quad (4.4)$$

CHAPTER 4. OPTIMAL RETRANSMISSION TIMEOUT (RTO) SELECTION FOR END-TO-END QoS

Thus, comparing with the simplest scenario, the error probability in this case is reduced by a factor $f_{n,k}$ and the rate cost for transmitting per source byte is n/k per source byte transmitted.

Another scenario is to use retransmission for end-to-end loss recovery. Let $t_0, t_1, \dots, t_{N-1}$ be N discrete transmission opportunities. If we repeatedly transmit the packet at all N opportunities, it will result in an error probability of $\prod_i P\{FTT > DTS - t_i\}$ and the rate cost equals to N . By smartly selecting some of the opportunities for transmission, we are able to tradeoff between the error probability and the rate cost.

In our research, we also consider the third scenario, i.e., using retransmission for end-to-end loss recovery. However, unlike that in [66], which has a fixed total number of transmission opportunities and studies whether a packet should be transmitted at each transmission opportunity, we look into the problem of the RTO selection. In other words, in our research, the total number of transmission opportunities is not fixed and the optimal tradeoff between the error probability and the rate cost is achieved through optimal RTO selection. We use the actual throughput as the optimization cost function, which is defined as

$$ActualThroughput = ABW \cdot \frac{1 - \epsilon}{\rho}, \quad (4.5)$$

where ABW is the available network throughput. Actual Throughput can be viewed as the real throughput for transmitting source information. Therefore, the optimal RTO problem can be summarized as: given the delay constraint DTS , how to optimally select RTO so that the Actual Throughput can be maximized.

4.2.1 Network Setup

In this research, the network is modeled as an independent time-invariant packet erasure channel with random delays. When a packet is transmitted at the time t , it could be lost with a probability of ϵ_F . If the packet is not lost, it will arrive at the receiver at the

CHAPTER 4. OPTIMAL RETRANSMISSION TIMEOUT (RTO) SELECTION FOR END-TO-END QoS

time t' , where the forward trip time $FTT = t' - t$ is randomly generated according to a probability density function $p_F(\tau|notlost)$. The lost and delay events of one packet are independent of those of other packets.

As in [66], a Gamma function with parameters of κ_F , n_F and α_F is employed for the delay probability density function (PDF) $p_F(\tau|notlost)$, which can be described as

$$p_F(\tau|notlost) = \frac{\alpha_F}{\Gamma(n_F)} (\alpha_F(\tau - \kappa_F))^{n_F-1} e^{-\alpha_F(\tau - \kappa_F)}, \quad (4.6)$$

where $\tau \geq \kappa_F$. In fact, this distribution is the same as a constant κ_F plus the sum of n_F independent identically distributed exponential random variables, each with parameter α_F . It can be interpreted as each packet going through n_F routers, each of which requires a constant processing time κ_F/n_F plus the waiting time in a steady state queue. The mean value of such a distribution is $\mu_F = \kappa_F + n_F/\alpha_F$ and the variance is $\sigma_F^2 = n_F/\alpha_F^2$. Note that FTT with the PDF shown in Eqn. (4.6) must be larger than κ_F .

By denoting $FTT = \infty$ as the event that the packet is lost, the packet loss probability and the packet delay density can be combined into a single probability measure. Thus the cumulative distribution function for the forward trip time is

$$P\{FTT \leq \tau\} = \int_0^\tau (1 - \epsilon_F) p_F(t|notlost) dt, \quad (4.7)$$

and its counterpart is

$$P\{FTT > \tau\} = \epsilon_F + \int_\tau^\infty (1 - \epsilon_F) p_F(t|notlost) dt. \quad (4.8)$$

The backward channel is setup similarly. In particular, when the receiver sends a packet to the sender, the packet could be lost with a probability of ϵ_B . If the packet is not lost, it will be delayed with a probability density function $p_B(\tau|notlost)$, which is also a Gamma function but with parameters of κ_B , n_B and α_B . Therefore, we can compute the corresponding probability for the round trip time, $RTT = FTT + BTT$, as

$$P\{RTT > \tau\} = \epsilon_F + (1 - \epsilon_F)\epsilon_B + (1 - \epsilon_F)(1 - \epsilon_B) \times \int_{\tau}^{\infty} \int_0^t p_F(t' | \text{notlost}) p_B(t - t' | \text{notlost}) dt' dt. \quad (4.9)$$

The first term of Eqn. (4.9) is the probability that the packet is lost in the forward trip, the second term is the probability that the ACK is lost in the backward path, and the last term is the probability that the ACK arrives but with a delay exceeding τ .

4.2.2 Proposed Optimal RTO Selection

Based on the analysis and the system setup described above, we propose the optimal RTO selection scheme in the following. As discussed before, our objective is to maximize the Actual Throughput by choosing appropriate RTO value given a delay constraint DTS . For a given DTS and a selected RTO, the total number of transmission opportunities N is determined as $N = \lceil \frac{DTS}{RTO} \rceil$ assuming the first transmission occurs at the time 0. When the sender sends out a packet, in the condition of ACK lost or delayed, it will keep waiting for a period of RTO and then resend the packet if not receiving an ACK. This process would stop after N times of transmissions.

In order to compute the Actual Throughput shown in Eqn. (4.5), we need to find the relationship between the error probability ϵ and RTO and the relationship between the rate cost ρ and RTO. Let $P\{\text{send } i \text{ times}\}$ denote the probability that a packet is exactly transmitted for i times. We can formulate the average cost of delivering one packet as

$$\rho = \sum_{i=1}^N i \cdot P\{\text{send } i \text{ times}\}, \quad (4.10)$$

and the error probability as

$$\epsilon = P\{\text{send } N \text{ times}\} \prod_{i=1}^N \frac{P\{FTT > DTS - (i-1)RTO\}}{P\{RTT > (N-i)RTO\}}. \quad (4.11)$$

CHAPTER 4. OPTIMAL RETRANSMISSION TIMEOUT (RTO) SELECTION FOR END-TO-END QoS

Eqn. (4.11) means that after N transmissions none of the packets reaches the receiver in time. For $i = 1, 2, \dots, N - 1$, we calculate $P\{\text{send } i \text{ times}\}$ by

$$P\{\text{send } i \text{ times}\} = (1 - P\{RTT > i \cdot RTO\}) \prod_{j=1}^{i-1} P\{RTT > j \cdot RTO\}, \quad (4.12)$$

which can be interpreted as the probability that the sender gets the first ACK at the time between the i th and the $(i + 1)$ th transmissions. When $i = N$, $P\{\text{send } N \text{ times}\} = 1 - \sum_{i=1}^{N-1} P\{\text{send } i \text{ times}\}$. Those RTT and FTT terms in Eqn. (4.12) and Eqn. (4.11) can be referred to Eqn. (4.8), Eqn. (4.9) and Eqn. (4.6).

After the derivation for ϵ and ρ , we can easily compute the Actual Throughput for any value of RTO. The optimal RTO value can be obtained by numerical methods such as using exhaustive search or using bisect search method in the range of all possible RTO values.

4.3 Preliminary Results

In this section, we conduct some experiments to compare the performance of the traditional RTO selection and our proposed RTO selection. The forward path is the same as the backward path. The parameters are set as : $DTS = 300ms$, $n_F = n_B = 2$, $\kappa_F = \kappa_B = 25ms$ and $\alpha_F = \alpha_B$. Based on the transmission model, for a given average RTT and a packet loss rate, we attempt to find out the optimal RTO within the range from $\kappa_F + \kappa_B$ to DTS . In particular, we sample different RTO values with a constant interval of 25 ms. For each RTO sample, we compute the theoretical Actual Throughputs and choose the one with the largest Actual Throughput. On the other hand, for comparison purpose, the Actual Throughput results by using the traditional RTO selection, i.e., $RTO = 2RTT$, are also calculated under different average RTT and different packet loss rate.

CHAPTER 4. OPTIMAL RETRANSMISSION TIMEOUT (RTO) SELECTION FOR END-TO-END QoS

Table 4.1, Table 4.2, Fig. 4.1 and Fig. 4.2 show the performance comparison between the traditional RTO selection and our proposed scheme under different average RTT and packet loss rates.

Table 4.1: The performance comparison between the traditional RTO selection and our proposed scheme under different average RTT with $\epsilon_F = \epsilon_B = 5\%$.

<i>RTT</i>	<i>60ms</i>		<i>80ms</i>		<i>100ms</i>		<i>120ms</i>	
α_F, α_B	0.400		0.130		0.080		0.057	
	Tra.	Opt.	Tra.	Opt.	Tra.	Opt.	Tra.	Opt.
<i>RTO</i>	120 ms	150ms	160 ms	200ms	200 ms	200ms	240 ms	225ms
ρ_N	1.1070	1.0975	1.0978	1.0975	1.0996	1.0996	1.1025	1.1070
ϵ_N	0.0001	0.0025	0.0025	0.0025	0.0033	0.0033	0.0219	0.0131
<i>throughput</i>	0.9032	0.9089	0.9086	0.9089	0.9064	0.9064	0.8872	0.8915

Table 4.2: The performance comparison between the traditional RTO selection and our proposed scheme under different average RTT with $\epsilon_F = \epsilon_B = 20\%$.

<i>RTT</i>	<i>60ms</i>		<i>80ms</i>		<i>100ms</i>		<i>120ms</i>	
α_F, α_B	0.400		0.130		0.080		0.057	
	Tra.	Opt.	Tra.	Opt.	Tra.	Opt.	Tra.	Opt.
<i>RTO</i>	120 ms	150ms	160 ms	175ms	200 ms	200ms	240 ms	200ms
ρ_N	1.4896	1.3600	1.3602	1.3600	1.3615	1.3615	1.3636	1.3786
ϵ_N	0.0080	0.0400	0.0400	0.0400	0.0428	0.0428	0.1052	0.0517
<i>throughput</i>	0.6660	0.7059	0.7058	0.7059	0.7031	0.7031	0.6562	0.6878

The left columns with title of "Tra." in Table 4.1 and Table 4.2 denote the performance of traditional RTO selection while the right columns with "Opt." denote that of our optimal scheme. From these tables and figures, one observation is that the Actual Throughput performance of our proposed RTO selection scheme is always better than that of the traditional approach, up to 5% gain. The Actual Throughput gain is more

CHAPTER 4. OPTIMAL RETRANSMISSION TIMEOUT (RTO) SELECTION FOR END-TO-END QoS

Figure 4.1: The performance comparison of the Actual Throughput between the traditional RTO selection and the proposed RTO selection under different average RTT. Left: with 5% packet loss rate. Right: with 20% packet loss rate.

Figure 4.2: The performance comparison of the Actual Throughput between the traditional RTO selection and the proposed RTO selection under different packet loss rates. Left: with the average RTT of 60ms. Right: with the average RTT of 120ms.

significant in the cases of larger packet loss rates, smaller average RTT or larger average RTT.

As mentioned above, the optimal RTO value is obtained by numerical methods such as exhaustive search or bisection search. In our experiments, the computation complexity of using exhaustive search is $O(N^2 K^2)$ and that of using bisection search is $O(N^2 K \lg K)$,

where N is the maximum times of transmissions and K is calculated as the total number of RTO samples. In practical system, the proposed optimal RTO and retransmissions maybe influence the available bandwidth, which should be further considered in the bandwidth estimation procedures.

4.4 Summary

In this chapter, we have proposed the algorithm of optimal RTO selection for end-to-end delay-constrained multimedia communications. Experimental results have shown our proposed scheme significantly outperforms the traditional RTO selection in the cases of larger packet loss rates, smaller average RTT or larger average RTT.

At the current stage, we have only conducted the experiments based on the transmission model, where we assume the sender knows the characteristics of the transmission paths such as RTT. In practice, the transmission model needs to be dynamically updated based on the feedback packets. Under such a circumstance, the estimated RTT may not be accurate which in consequence will affect the RTO selection.

Chapter 5

RTP-Based Transportation Platform with End-to-End QoS Management

Video streaming over IP is one of the hottest applications for Internet. Streaming video requires low-loss, low-delay and bandwidth-assured service. However, current multimedia transport protocol, the real-time transport protocol (RTP) does not support the QoS guaranteed services. To implement QoS management such as congestion control and error control into the RTP-based transportation platform is the goal of this research. In this chapter, we present our design and implementation of a practical RTP-based transportation platform with end-to-end QoS management for video over IP. The chapter is organized as follows. Section 5.1 gives some background and discusses the design consideration. Section 5.2 describes the proposed system architecture. Section 5.3 and Section 5.4 introduces the developed congestion control and error control. Experimental results to evaluate the streaming platform are shown in Section 5.5. Finally, summary is given in Section 5.6.

5.1 Background

As discussed in Chapter 1, packet loss in IP-layer could be caused by either network congestion and delay or underlying physical channel degradation such as wireless links.

CHAPTER 5. RTP-BASED TRANSPORTATION PLATFORM WITH END-TO-END QoS MANAGEMENT

In this chapter, we mainly focus on packet loss caused by network congestion and delay. In particular, when the network is busy, the packet arrival rate could become much larger than the service rate at router since the network bandwidth is limited. If a large number of packets produced by misbehaving senders are queuing at routers and the queue length exceeds the routers' buffer length, congestion occurs and routers will have to drop some packets. In another scenario, a packet might be discarded not due to congestion, but because the transmission time exceeds its time to live (TTL) and the packet becomes useless and "dead". In addition, if a media packet survives in the network but its timestamp is too late for the decoder, the packet still can not be used. To provide a high-quality transportation, QoS must be guaranteed to some extension by the network and the end-point systems.

However, the multimedia transport protocol RTP does not guarantee reliable QoS. In order to support fluent multimedia streaming, QoS management schemes such as congestion control and error control should be introduced into the transportation platform of our system.

5.2 System Architecture

In this research, we choose a client-server architecture based on the Darwin Streaming Server [67] and the MPEG4IP player [68]. Basically, Darwin streaming server implements standard RTP and RTSP protocols to stream media over Internet and MPEG4IP player is able to receive RTP packets and play out the media. Detailed descriptions of Darwin Streaming Server and MPEG4IP player can be found in Appendix A.

Although a straightforward integration of Darwin streaming server and MPEG4IP can stream video over Internet, there is no QoS management at all. Therefore, we need to add in necessary components for end-to-end QoS management. In particular, RTCP packets are used to monitor the QoS parameters and provide information for QoS management.

CHAPTER 5. RTP-BASED TRANSPORTATION PLATFORM WITH END-TO-END QoS MANAGEMENT

For each RTP stream, the server must periodically create and transmit RTCP sender report packets, and the client responds receiver report packets. These reports provide fundamental materials for congestion control and error control.

The developed multimedia streaming system should have the following properties.

- *Modularity.* The developed system should be divided into different functional modules such as encoding, packetizing, transporting, decoding and displaying, and each module should be well cooperating with others. This research focuses on the transporting modules which are responsible for controlling RTP and RTCP transportation. The management taken on the transporting modules should not affect the basic functions of other modules.
- *Compatibility.* The modifications we make should not damage the standard functions of the original software. In other words, a particular module we develop with QoS management should be able to work with other modules with or without QoS management. For example, the modified Darwin Streaming Server can communicate normally with the unmodified MPEG4IP player. In this way, different modules developed in this system can be easily integrated into other software.
- *Flexibility.* The developed system should provide flexible options and interfaces for applications and future development. For example, the application should be able to choose different error control schemes implemented in the transportation platform. Interface should be provided to pass information between the application and the transportation system. New algorithms on congestion control and error control can be easily used to replace the old ones.

Fig. 5.1 shows the implemented system architecture, which is based on the end-to-end transport model in [14]. As shown in the figure, the modules between source encoder

CHAPTER 5. RTP-BASED TRANSPORTATION PLATFORM WITH END-TO-END QoS MANAGEMENT

and decoder are monitored and managed by the RTP-based transportation platform. In particular, the QoS manager estimates current available bandwidth according to the RTCP feedbacks, and then informs the source encoder to adjust its coding rate to fit the available bandwidth. In this way, the transmission rate can be dynamically changed with the varying network conditions.

Figure 5.1: The architecture of the developed video streaming system.

When video data produced from the source encoder, they are transmitted to Internet by using RTP protocol. If packet loss occurs, the receiver will record the sequence numbers of lost packets, and inform the sender by the RTCP Receiver Report (RR). Then the sender adjusts its sending rate, and retransmits the lost packet in time if the packet's timestamp is still valid. Alternatively, forward error correction (FEC) can also be used to recover lost packets instead of retransmission. Specifically, media data will be interleaved and appended with Reed-Solomon (RS) codes before transmission. When data arrive at the receiver, they must be decoded and deinterleaved before delivered to the source decoder. The system is quite flexible and it is up to the user or the application to choose the appropriate error control mechanisms.

5.3 Congestion Control

We choose the TCP friendly rate control (TFRC) [13] as the congestion control scheme in our developed transportation system. TFRC is a kind of rate-based congestion control mechanism, where the sender adjusts its sending rate according to a throughput equation that is derived from the estimated RTT and loss event rate [31]. The throughput equation is shown in Eqn. (2.2), where the loss event rate can be computed as $1/I_{mean}$. The value I_{mean} denotes the average loss interval, and the loss interval is defined as the number of received packets between two consecutive "lost" RTTs, in which at least one packet is lost. In other words, once packet loss occurs in one RTT, this RTT is regarded "lost".

It has been proved in [69] that TFRC changes its sending rate more smoothly than typical AIMD schemes, while having a similar transient response to a sudden increase in congestion. Furthermore, recent works such as [70] and [71] indicate TFRC can finely cooperate with RTP in multimedia communications. Thus it is wise to choose TFRC as the congestion control scheme for our transportation platform. Note that, in TFRC, retransmission timeout (RTO) is set to four times of the mean RTT. Although more accurate calculation of RTO is possible, the experiments in [13] have demonstrated that such a simple configuration can achieve reasonable fairness with existing TCP implementations. Therefore, we follow this simple setting of $RTO = 4 * RTT$, without employing our optimal RTO selection scheme in this practical system.

To implement TFRC, the receiver records the arrival history for each received data packet. The sender must periodically send the current estimated RTT to the receiver so that the receiver can use the RTT to compute the loss intervals and loss event rate. Then the receiver sends the computed loss event rate to sender to calculate the available bandwidth and adjust its sending rate [71]. In the developed system, we use the application-defined (APP) packet to the sender's estimated RTT, following the sender

CHAPTER 5. RTP-BASED TRANSPORTATION PLATFORM WITH END-TO-END QoS MANAGEMENT

report (SR). The receiver report (RR) is used at the client side to report the loss event rate. Such interactive procedures between server and client are illustrated in Fig. 5.2.

Figure 5.2: TFRC feedback loops implemented in RTCP [71].

In order to satisfy the TFRC specification, four fields are added into RTCP/RR packet [3], and each field occupies 32 bits, as shown in Fig. 5.3. These four fields are the sending time of the last received packet, the delay of generating RR, the receiver rate and the loss event rate [72]. The sending time indicates when the packet is transmitted and it is required to be sent by the sender [13]. However, it is unnecessary if the sender has recorded the sending time of each transmitted packets, which is the case in Darwin Streaming Server. Hence, this field is not used in our system, but it is reserved in RR for future use. The field of the delay of generating RR is the amount of time elapsed between the arrival of the last data packet at receiver and the generation of RR. Suppose the last packet received at the client has sequence number i , its sending time is ts_i and the delay is t_{delay} . When the feedback RR is received by the server at time t_{now} , the current RTT sample is calculated as

$$RTT_{sample} = (t_{now} - ts_i) - t_{delay}, \quad (5.1)$$

and the new estimated RTT becomes

$$RTT_t = 0.9 * RTT_{t-1} + 0.1 * RTT_{sample}. \quad (5.2)$$

CHAPTER 5. RTP-BASED TRANSPORTATION PLATFORM WITH END-TO-END QOS MANAGEMENT

The next two fields, the receiver rate and the loss event rate are used by TFRC to estimate available bandwidth and adjust its sending rate. Note that the sending rate can not be larger than two times of the receiver rate.

Figure 5.3: The receiver report format for TFRC.

The feedback APP packets from the receiver are used to tell the sender the sequence numbers of the lost packets. The format of a feedback APP packet is shown in Fig. 5.4. As we know, in TCP protocol, the sender expects to receive an ACK for each RTP packet that it sends. If it does not receive an ACK for a packet, the sender will consider it a loss and retransmit the packet. However, in the case of RTP-based video streaming, usually, the sender transmits packets incessantly rather than stop-and-wait for ACKs. Therefore, in our developed system, the receiver does not send back ACK for each data packet received from the sender. Instead, ACKs for several packets are coalesced together and sent to the sender in a single packet [73].

CHAPTER 5. RTP-BASED TRANSPORTATION PLATFORM WITH END-TO-END QoS MANAGEMENT

Fig. 5.4 shows the adopted receiver APP format. Besides the standard APP packet header, some additional fields are added including a RTP sequence number and a variable length bit mask. The sequence number identifies the first RTP packet that the receiver is acknowledging. Each additional RTP packet being acknowledged is indicated by a bit set in the bit mask. The bit mask is an offset from the specified sequence number, e.g., the first bit represents the sequence number that is one greater than the initial sequence number.

The bit mask must be sent in multiples of 4 octets (32 bits). Setting a bit to 0 in the mask simply means that the receiver does not wish to acknowledge this sequence number right now, and does not imply a negative acknowledgment in any way [73]. For example, the receiver obtains the sequence numbers from 12025 to 12055, and the packets from 12030 to 12035 are lost. The "seq num" should be 12025, and the "mask" should be 11110000,00111111,11111111,11111100.

Figure 5.4: The receiver APP format [73].

As mentioned above, the sender needs to tell the receiver its estimated RTT values which are delivered by sender's APP packet. Fig. 5.5 shows the format of the sender APP packet. When the sender's RTT arrives at the receiver, it will be used to calculate the packet loss interval and loss event rate, based on the packet arrival history recorded at

CHAPTER 5. RTP-BASED TRANSPORTATION PLATFORM WITH END-TO-END QOS MANAGEMENT

the receiver. In particular, the receiver considers a packet lost occurs when the received sequence number is higher than the expected one. A loss event is defined as the event of packet loss within one RTT. Once an initial loss occurs, the following loss within this RTT is ignored. To implement TFRC, the receiver records the sequence numbers of every data packet in the arrival history. When it detects a packet loss, the receiver refers to the value of RTT to judge whether it raises a new loss event or not. The loss interval is defined as the number of received packets between two loss events, and the loss event rate is computed as the reciprocal of the mean loss interval. Suppose l is the mean loss interval, the lost event rate $1/l$ means there is a loss event arising every l packets received. More details on computing loss event rate can be found in [13].

Figure 5.5: The sender APP format.

5.4 Error Control

Two error control schemes, a class-based retransmission scheme [74] and a simple unequal erasure protection (UXP) scheme, are implemented in the developed transportation system.

5.4.1 Class-Based Fast Retransmission

In the RTP profile for Darwin Streaming Sever [73], the receiver informs the sequence numbers of lost packets to the sender through RTCP/APP. Once the sender gets the feed-

CHAPTER 5. RTP-BASED TRANSPORTATION PLATFORM WITH END-TO-END QoS MANAGEMENT

back, it will retransmit the corresponding packets if the timestamp of the media packet is still valid. Because of round trip delay and out-of-order delivery, retransmission is usually not recommended in real-time multimedia applications. However, streaming video allows certain playout buffer delay (generally 2-5 seconds) to adopt latency-constrained retransmission. The playout buffer delay in MPEG4IP is typically set to 2 seconds, and hence we conservatively adopt retransmission in the developed system and limit the lost packet to be retransmitted only once.

Since RTCP feedbacks are generated with a large delay [3], it may be too late to recover some important packets. In our design, we employ a class-based fast retransmission scheme, i.e., fast ACKs are used for the significant packets, while other unimportant packets have to wait for APP feedback packets. Currently, data packets are classified into 4 categories: Class 0 to Class 3, with priorities declining. Class 0 is not used in retransmission-based error control. In this research, we consider video coding with I, P and B frame types, and we simply classify I-frame data, P-frame data and B-frame data as Class 1, 2 and 3, respectively. In order to support more classes in the future, we use 1 byte to indicate the data class type. Currently, we only apply fast retransmission to Class 1 packets, i.e., I-frame packets. Four bytes are appended at the end of each RTP data packet, which indicate the classes of the four recently received packets. The reason to choose 4 bytes is that in TFRC the loss of packet is detected by the arrival of at least three out-of-sequence packets. For instance, if the receiver receives 6 packets belonging to B, B, I, B, B, P frames, the last 4 bytes of the last received packet should be 1332. Since 4 octets are padded at the end of the packet payload, the padding bit "P" in RTP fixed header should be set to 1. Once the receiver receives a packet, it will check the 4 padded bytes and the arrival history to find out whether there is Class 1 packet lost or not. If it detects Class 1 packet lost and the time is enough to retransmit the packet, the receiver will send a fast ACK to the sender immediately and the sender must retransmit

the packet in time. For the lost packets of other classes, the receiver will keep waiting until the time to send RTCP feedback packets, as described in Fig. 5.4.

The fast ACK is generated to recover the lost Class 1 packet timely. To be easily understood by the sender, the fast ACK simulates the format of RTCP APP. It has a fixed size of eight bytes, and its most important fields are PT and sequence number. PT is set to 220 to indicate a fast ACK, and sequence number indicates the sequence number of the latest lost Class 1 packet. Fig. 5.6 shows the format of a fast ACK packet.

Figure 5.6: The format of a fast ACK packet.

5.4.2 Unequal Erasure Protection (UXP)

An unequal erasure protection (UXP) scheme [75] is developed in our system as another option for error control. Unequal erasure protection (UXP) based on FEC and interleaving is an efficient way to recover erasure errors. The reason we use UXP is that different portions of video data have different importance. RS code is chosen because it can offer maximum erasure correction capability with minimum amount of redundancy. An (n, k) RS code can correct up to $t = n - k$ erasure errors given the position of the erasure errors is known. To reduce coding delay and make FEC flexible, a shortened (n, k) RS code is adopted, which has been explained in Section 2.3.1.

Fig. 5.7 shows the structure of UXP, which consists of L rows and n columns. It is basically a block interleaver. The input n RTP packets are divided into RTP header and payload data. Part of RTP header will be placed at the top and repeated in the

CHAPTER 5. RTP-BASED TRANSPORTATION PLATFORM WITH END-TO-END QOS MANAGEMENT

profile block as Class 0 data for heavy protection since RTP header is not protected by FEC. The RTP payload will be put in RTP data block from left to right and row by row. Each packet occupies a data block and the corresponding FEC code is determined by the packet class. If a packet payload can not fill out multiple rows exactly, stuffing bytes of 0x00 are inserted. After FEC encoding, data are read out column by column with the corresponding headers to form new RTP packets.

Figure 5.7: The structure of UXP.

In our design, the number of interleaving packets n must be smaller than 256 symbols, and a symbol consists of 8 bits. Considering the delay requirement, n is chosen to be

CHAPTER 5. RTP-BASED TRANSPORTATION PLATFORM WITH END-TO-END QOS MANAGEMENT

16, and L is limited to be smaller than MTU, the maximum size of an IP packet. The FEC configuration for Class 0 to 3 can be set by the user. When packets arrive at the receiver, they are put into a deinterleaver column by column and decoded with FEC. If a large number of packets are lost and FEC decoding fails, all packets in the deinterleaver would be dropped by the receiver. If there are only a few packets lost, the RS codes can recover the media data within the range of correction capacity.

Fig. 5.8 shows the format of the profile block which contains timestamp, marker bit (M), class, start position and the length of each RTP data block. The reason to heavily protect these data is that they are very important. In particular, if the timestamp in a RTP header is lost, the media data can not be played even though they are recovered. The marker bit (M) is used to indicate the packets at frame boundaries. Given the recovered timestamp and the "M" bit, the lost RTP header can be reconstructed according to the lately received packet header. The packet class is very important because the class priority determines the FEC configuration. The start position indicates at which row the data block starts. It works with the packet data length to reconstruct the original RTP packet.

Figure 5.8: The format of the profile block.

Note that since the sender may use different error control schemes, it is needed to tell the receiver which scheme is being used. In our design, we combine the "X" bit and the padding bit "P" in the RTP header to indicate the error control scheme, as shown in Table 5.2.

Table 5.1: Identification of error control mode.

X	P	Error control mode
0	0	No error control
0	1	Fast retransmission
1	0	UXP
1	1	UXP, and the last column packet

5.5 System Implementation

Fig. 5.9 illustrates the flow chart of the QoS management process in the developed system. Specifically, RTP packets are transmitted incessantly by the server, and the media data are presented continuously by the client. Meanwhile, the RTCP reports are generated and sent in an interval according to the allocated RTCP bandwidth, and the server’s transmission rate will be updated upon the arrival of every RR from the client. The error control modes are currently predefined by users.

We also define the interface between the application and the transportation system to exchange information for QoS management. The interface includes four functions:

```
int GetEstimatedBandwidth();  
int GetLossRate();  
int GetErrorControlMode();  
int GetPacketClass(int seq);
```

In our design, QoS management is performed on every RTP stream. The functions of *GetEstimatedBandwidth()* and *GetLossRate()* report the estimate bandwidth and the loss event rate of current stream to the upper layer. According to the estimated bandwidth and the loss rate of each stream, the source encoder will adjust its coding rate to fit the network bandwidth. On the other hand, the transportation module needs the upper layer to provide the error control mode for each media session and the class priority for each RTP packet. *GetErrorControlMode()* determines which error control scheme is

CHAPTER 5. RTP-BASED TRANSPORTATION PLATFORM WITH END-TO-END QoS MANAGEMENT

used, fast retransmission or UXP. *GetPacketClass(int seq)* returns the class of a packet for class-based protection. In this way, the QoS management in the RTP-based transportation platform are transparent to the upper layer. Details about integrating the QoS management in the server and client can be referred to Appendix A.

Figure 5.9: Flow chart of QoS management.

5.6 Experimental Results

Experiments are conducted to evaluate our developed transportation system with QoS management, compared to the system without QoS management. In the experiments, Darwin Streaming Server runs at a desktop computer in Windows 2000, and MPEG4IP player runs at a laptop computer in Windows XP. These two computers are connected through wired link. An MPEG-4 video sequence at the server is packetized and transported to the client via the wired link. As the server and client are connecting through LAN, no packet loss occurs during transmission. To simulate packet loss in Internet, we use the GEC model, described in Chapter 3, to model consecutive loss in network [52, 53]. Particularly, media packets are dropped according to the GEC model deployed at the client side.

First we measure the performance of congestion control. For this case, the error control mode is set to no error control to avoid the extra bandwidth cost by retransmissions or FEC redundancies. We choose the average packet loss rate (PLR) of the GEC model to be 10% and 20%, and the average burst length is set to 3 packets. A standard 10-second QCIF video sequence "coastguard" coded at 176 kbps is extended to 100 seconds by repeating 10 times, and transmitted from the server to the client.

Fig. 5.10 shows the measured transmission rates with and without congestion control under 10% and 20% packet loss rates. The curve TR_1 denotes the estimated transmission rate with congestion control, TR_0 indicates the current transmission rate without congestion control, and p is the loss event rate. It can be seen that, for the case with congestion control, higher loss event rate results in lower transmission rate. If loss event rate increases in the current interval, the transmission rate will be affected and fall down in the next interval. On the other hand, for the case without congestion control, the sender always transmits the video in a fixed rate of 30 frames per second, and it never

CHAPTER 5. RTP-BASED TRANSPORTATION PLATFORM WITH END-TO-END QoS MANAGEMENT

considers the network changes. This experiment demonstrates that the congestion control can make the system adaptive to the varying network conditions.

Figure 5.10: The performance comparison between the transmission rates with and without congestion control. Left: PLR = 10%. Right: PLR = 20%.

Next, we measure the reconstructed video quality in term of PSNR under different error control modes. Note that if there is an uncorrectable error in a frame, the source decoder in the MPEG4IP player will drop the entire frame and the lost frame will be concealed by the lately received frame.

Fig. 5.11 shows the PSNR results over different video frames with 10% and 20% packet loss rates. It can be seen that, without error control, the PSNR values have great degradation at most of the time. Fig. 5.12 shows the PSNR results of the fast retransmission with 10% and 20% packet loss rates. Since the fast retransmission is only applied to lost I-frame data, the additional bandwidth cost is insignificant, around 8% of the original bandwidth. The PSNR degradation shown in Fig. 5.12 is not as severe as that without error control, but it is still not satisfactory. Fig. 5.13 shows the PSNR results of the unequal erasure protection with 10% and 20% packet loss rates. We use RS code (16,10) to protect the profile data (Class 0), and use RS code (16,11) to protect the media data (Class 1-3). The additional bandwidth cost is about 32% of the original

CHAPTER 5. RTP-BASED TRANSPORTATION PLATFORM WITH END-TO-END QoS MANAGEMENT

bandwidth. We can see that UXP significantly outperforms the other two cases. UXP achieves much smaller number and much shorter period of severe PSNR degradation. Those severe PSNR degradation peaks are caused by the highly burst packet loss, which exceeds the correction capability of RS codes.

Table 5.2 lists the average PSNR results in different error control modes.

Table 5.2: The comparison of the average PSNR (dB) in different error control modes.

PLR	no error control	fast retransmission	UXP
10%	28.16	28.68	29.28
20%	26.62	26.93	27.97

5.7 Summary

In this chapter, we have designed and implemented a practical RTP-based transportation system for multimedia streaming with end-to-end QoS management. In particular, we have developed the system based on Darwin Streaming Server and MPEG4IP player. We have incorporated TFRC into the system for congestion control, and two error control schemes to combat with packet loss, the class-based fast retransmission and UXP. Experimental results have demonstrated that our developed system is able to perform end-to-end QoS management properly.

CHAPTER 5. RTP-BASED TRANSPORTATION PLATFORM WITH END-TO-END QoS MANAGEMENT

Figure 5.11: The PSNR results without error control. Left: $PLR = 10\%$. Right: $PLR = 20\%$.

Figure 5.12: The PSNR results of fast retransmission. Left: $PLR = 10\%$. Right: $PLR = 20\%$.

Figure 5.13: The PSNR results of UXP. Left: $PLR = 10\%$. Right: $PLR = 20\%$.

Chapter 6

Conclusion and Future work

6.1 Conclusion

This thesis has presented my two-year research work on end-to-end QoS management for multimedia over IP. Our objective is to study, design and implement end-to-end QoS management schemes, particularly on congestion control and error control in the transportation layer with RTP. Until now, we have studied some theoretical problems including wireless link modeling and optimal RTO selection. We have also implemented a practical transportation system for multimedia streaming over IP with end-to-end QoS management.

In order to study wireless network performance in terms of QoS parameters, we have looked into wireless link modeling and have developed a simple algorithm for converting GECs from lower level to higher level. This will benefit the performance study of wireless link-layer under different error control schemes such as ARQ, FEC and hybrid ARQ. The resulted link-layer characteristics will further make contribution to the upper layer performance characterization and finally to be used in the study of end-to-end QoS management.

We have also studied the problem of optimal RTO selection for end-to-end delay-constrained multimedia communications and proposed a method that is able to theoretically obtain the optimal RTO value based on the network delay model. Experimental

CHAPTER 6. CONCLUSION AND FUTURE WORK

results have shown our proposed scheme significantly outperform the traditional RTO selection in the cases of larger packet loss rates, smaller average RTT or larger average RTT.

In addition, we have developed the RTP-based transportation system based on Darwin Streaming Server and MPEG4IP player. We have incorporated TFRC into the system for congestion control, and two error control schemes to combat with packet loss, the class-based fast retransmission and UXP. A simple interface is defined for the application and the transportation system to exchange information for QoS management. Experimental results have demonstrated that our developed system is able to perform end-to-end QoS management properly.

6.2 Future work

End-to-end QoS management for multimedia communications is a very interesting and challenging research area and has many unsolved problems. In the future, our research works can be enhanced in the following areas:

- Dynamic end-to-end network performance characterization

As we have obtained wireless link models in Chapter 3, the wireless link performance in the IP layer would be summarized in terms of the QoS parameters such as packet loss and delay. Note that wireless channels are time-varying and non-stationary, the characterization of wireless network performance should be dynamically updated according to wireless channel feedbacks and estimation.

It is also interesting to consider the scenario of hybrid wired and wireless transmission. The obtained characterization of wireless network performance can be integrated into existing wired network models and simulation platforms to characterize end-to-end network performance. Again, wired networks such as Internet is

CHAPTER 6. CONCLUSION AND FUTURE WORK

also time-varying due to traffic load changing. The end-to-end network performance characterization must also be dynamic according to the upper layer feedback and estimation. The channel and network estimations are very important since they are the foundations for the proposed end-to-end QoS management. If the estimation is not accurate, it will significantly affect the performance of end-to-end QoS management.

- Improved end-to-end QoS management

In our design of the RTP-based transportation system for multimedia streaming over IP, error control schemes such as fast retransmission and UXP are currently predefined by users. The system can be enhanced to dynamically select an optimal error control scheme according to network conditions. Moreover, according to different packet loss rates, we should use different FEC codes. In addition, the optimal RTO selection can also be employed in the developed platform. In the case of multimedia over IP with hybrid wired and wireless links, the congestion control should distinguish the packet loss caused by network congestion and wireless errors, and react accordingly.

Appendix A

System Integration

Since two open-source software, the Darwin Streaming Server and MPEG4IP player, are chosen as the server and client in our multimedia streaming system, functionalities and characteristics of these two software should be introduced. Moreover, we would explain some details about integrating the QoS management schemes in Darwin Streaming Server and MPEG4IP player, so that others can follow easily to set up the system.

A.1 Darwin Streaming Server

Darwin Streaming Server is an open-source version of Apple's QuickTime Streaming Server, supporting QuickTime and MPEG-4 steaming media on alternative platforms such as Windows, Linux, and Solaris. It implements standard RTP and RTSP protocols to send streaming media to clients across Internet, and it provides a high level of customizability and runs on a variety of platforms which allow users to manipulate the source codes to fit their requirements [67].

Darwin Streaming Server works with modules to process requests from clients by invoking modules in a particular role. Each role is designed to perform a particular task. In practice, clients use RTP and RTSP to send requests and receive responses, and server uses modules to process the requests and send packets to the clients. A core server acts

CHAPTER A. SYSTEM INTEGRATION

as an interface between modules and network clients, by creating four types of threads [73]:

- *Server's Main thread*, which monitors whether the server needs to shut down, log status information, or print statistics.
- *Idle Task thread*, which manages a queue of tasks, like timeout tasks and socket tasks that occur periodically.
- *Event thread*, which listens for socket events such as received RTSP request or RTP packet and forwards them to a Task thread.
- *Task threads*, which receive RTSP and RTP requests from the Event thread and forward them to the appropriate server module for processing and send packets to the clients.

The relationship between clients, the core server's threads and server modules is illustrated in Fig. A.1 .

Figure A.1: Darwin server architecture (copied from [73]).

CHAPTER A. SYSTEM INTEGRATION

There are three types of modules: server-support modules, which perform server data gathering and logging functions; access control modules, which provide authentication and authorization functions as well as URL path manipulation; and content-managing modules, which manage RTSP requests and responses related to media sources. Each of content-managing modules is responsible for interpreting the client's request, reading and parsing their supported files or network source, and then responding with RTSP and RTP operations [73].

To integrate QoS management into Darwin Streaming Server, some important content-managing modules which run RTP and RTCP transportation would be modified, such as RTPStream module, RTPPacketResender module, RTPBandwidthTracker module and RTCPUtilities modules, etc. To maintain the compatibility of Darwin Server, other modules unrelated to transportation are kept intact.

Referring to the system flow chart (Fig. 5.9), the QoS management schemes such as TFRC, UXP and the class-based fast retransmission are integrated into Darwin Streaming Server as follows.

In function *RTPStream::ProcessIncomingRTCPPacket()*, each packet of RTCP feedbacks (such as RR, APP and fast ACK) is justified and parsed by the sender. In the case of RR, the sender gets the loss event rate and the receiver rate from RR. In the case of receiver's APP, the sender records the sequence number of ACKed packets and finds out whether there is packet loss. In the case of fast ACK, the sender gets the sequence number of the lately lost Class 1 packet and directly calls function *RTPPacketResender::ResendDueEntries()* to retransmit the packet.

Function *RTPStream::ReliableRTPWrite()* is used to transmit each RTP packet. Firstly, it justifies whether there is unacknowledged packet or not, and then calls function *RTPPacketResender::ResendDueEntries()* to launch retransmission if packet loss occurs. Following, the original RTP packets are reorganized for class-based fast retransmission

or UXP. In the case of fast retransmission, 4 bytes of class information should be attached at the end of each RTP packet payload. In the case of UXP, the sender calls function *RTPStream::ProcessInterleaver()* to perform interleaving and FEC for every 16 RTP packets. After re-packetization, the new packets are sent out to the receiver.

The function *RTPStream::ProcessInterleaver()* is created by wenning. In this function, the profile data (Class 0) and media data (Class 1-3) of the 16 packets are put into interleaver row by row, while the length of profile data in each row is 10 bytes and the length of media data is temporarily set to 11 bytes. After encoded by RS codes in rows, the data packets are output column by column from the interleaver with their corresponding RTP headers.

Function *RTPBandwidthTracker::AddToRTTEstimate()* estimates new RTT values based on Eqn. (5.2), and function *RTPBandwidthTracker::AdjustWindowForRetransmit()* is used to calculate the transmission rate according to the TFRC throughput equation.

It is in function *RTPStream::SendRTCP SR()* that RTCP Sender Reports are generated. Here the estimated RTT value is inserted into the sender's APP followed by SR, as shown in Fig. 5.5.

A.2 MPEG4IP Player

MPEG4IP is a kind of open-source standards-based software for encoding, streaming, and playing MPEG-4 encoded audio and video. It provides an end-to-end system to explore streaming multimedia. MPEG4IP includes and integrates many existing open-source packages which are standards-oriented and free from proprietary protocols and extensions, therefore, it enables developers to create streaming servers and clients freely to fit their needs [68].

The MPEG4IP player supports most MPEG-4 videos, and some other video formats such as MPEG1/2, H.261, and YUV, etc. Its streaming functionality supports standard

CHAPTER A. SYSTEM INTEGRATION

protocols such as RTSP and RTP. Moreover, MPEG4IP provides further platforms for multicasting [68].

Figure A.2: MPEG4IP player architecture (copied from [68]).

Fig. A.2 indicates the architecture of MPEG4IP player. In the figure, a general multimedia session received from remote server consists of two streams, video stream and audio stream. There are three main tasks to handle this player session, RTP task, decoder task and synchronization task, each of which is an independent thread. Since both video stream and audio stream have their own queues and codec, RTP task and decoder task have to treat these two streams respectively. At first, RTP task puts the received data packets into queues, and then decoder task takes the packets from queues and decodes them with corresponding video or audio codec. Finally, synchronization task synchronizes the audio and video, and delivers them to displayer.

CHAPTER A. SYSTEM INTEGRATION

To support congestion control and error control in transport layer, the RTCP reports and the RTP task (*in RTP.C*) of MPEG4IP are partially modified. In terms of Fig. 5.9, some details about modification are simply introduced below.

Function *rtp_rcv()* is used to parse every RTP and RTCP packet received from the server. Once a RTP packet received, the packet's sequence number and arrival time are recorded in the arrival history and forwarded to function *rtp_rcv_data()*. In the case of RTCP packet, the receiver will extract the sender's estimated RTT from the packet, and then go to function *rtp_process_ctrl()*.

In function *rtp_rcv_data()*, if the error control mode is defined as UXP ("X"=1), the received packets are gathered and waiting for deinterleaving. When the last column packet of the sender's interleaver ("P"=1) is received, the client starts deinterleaving and FEC decoding to restore the original RTP packets. If packet loss is burst and unable to restore, all packets in the deinterleaver should be discarded.

In function *rtp_process_rcv_data()*, if class-based fast retransmission mode is predefined ("X"=0, "P"=1), the last 4 bytes of each packet's payload should be truncated. On the other hand, once the 4 bytes indicate there is Class 1 packet lost, fast ACK will be sent to the server immediately if time is enough for retransmission. Duplicate packet will be dropped if received.

According to the arrival history, function *format_rtcp_app()* constructs the APP feedback packet as described in Fig. 5.4, while in function *format_report_blocks()* the receiver rate and loss event rate are computed and integrated into RR block with other two fields as shown in Fig. 5.3. Therefore, the size of RR block grows to 40 bytes instead of original 24 bytes. After the APP blocks and RR blocks are formatted, the client calls function *send_rtcp()* to send RTCP feedbacks to the server.

References

- [1] F. Kuo, W. Effelsberg, and J. Garacia-Luna-Aceves, *Multimedia communications: protocols and applications*. Prentice-hall International, Inc., 1998.
- [2] A. S. Tanenbaum, *Computer networks 3rd Ed.* Prentice-hall International, Inc., 1996.
- [3] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: a transport protocol for real real-time applications," in *RFC 3550, IETF*, Jul. 2003.
- [4] H. Schulzrinne, A. Rao, and R. Lanphier, "Real time streaming protocol (RTSP)," in *RFC 2326, IETF*, Apr. 1998.
- [5] A. Uziel and M. Tummala, "Protocol architecture for tactical integrated services mobile networks," in *MILCOM 97 Proceedings*, vol. 3, pp. 1532–1536, 1997.
- [6] J. Shin, J. W. Kim, and C.-C. J. Kuo, "Quality-of-service mapping mechanism for packet video in differentiated services network," *IEEE Trans. on Multimedia*, vol. 3, pp. 219–231, Jun. 2001.
- [7] S. Maniatis, E. Nikolouzou, and I. Venieris, "QoS issues in the converged 3G wireless and wired networks," *IEEE Communications Magazine*, vol. 40, pp. 44–53, Aug. 2002.
- [8] S. Dixit, Y. Guo, and Z. Antoniou, "Resource management and quality of service in third-generation wireless networks," *IEEE Communications Magazine*, vol. 39, pp. 125–133, Feb. 2001.

REFERENCES

- [9] P. Flegkas, P. Trimintzios, and G. Pavlou, "A policy-based quality of service management system for IP DiffServ networks," *IEEE Network*, vol. 16, pp. 50–56, Mar. 2002.
- [10] K. Svanbro, J. Wiorek, and B. Olin, "Voice-over-IP-over-wireless," in *PIMRC 2000*, vol. 1, pp. 24–28, Sep. 2000.
- [11] H. Chaskar, T. V. Lakshman, and U. Madhow, "On the design of interfaces for TCP/IP over wireless," in *MILCOM '96*, vol. 1, pp. 199–203, Oct. 1996.
- [12] Q. Zhang, W. Zhu, and Y.-Q. Zhang, "A cross-layer QoS-supporting framework for multimedia delivery over wireless Internet," in *PV 2002*, 2002.
- [13] M. Handley, S. Floyd, J. Padhye, and J. Widmer, "TCP friendly rate control (TFRC): protocol specification," in *RFC 3448, IETF*, Jan. 2003.
- [14] D. Wu, Y. T. Hou, W. Zhu, H.-J. Lee, T. C. Chiang, Y.-Q. Zhang, and H. J. Chao, "On end-to-end architecture for transporting MPEG-4 video over the Internet," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 10, pp. 923–941, Sept. 2000.
- [15] D. M. Chiu and R. Jain, "Analysis of the increase and decrease algorithms for congestion avoidance in computer networks," in *DEC-TR-509*, Aug. 1987.
- [16] D. Sisalem and A. Wolisz, "Towards TCP-friendly adaptive multimedia applications based on RTP," in *IEEE International Symposium on Computers and Communications*, pp. 166–172, Jul. 1999.
- [17] V. Jacobson, "Congestion avoidance and control," in *ACM SIGCOMM'88*, pp. 314–329, Aug. 1988.
- [18] L. S. Brakmo, S. W. O'Malley, and L. L. Peterson, "TCP Vegas: New techniques for congestion detection and avoidance," in *SIGCOMM'94*, pp. 24–35, Oct. 1994.
- [19] C. P. Fu and S. C. Liew, "TCP Veno: TCP enhancement for transmission over wireless access networks," *IEEE Journal on Selected areas in Communications*, vol. 21, pp. 216–228, Feb. 2003.

REFERENCES

- [20] D. Sisalem and A. Wolisz, "LDA+: a TCP-friendly adaptation scheme for multimedia communication," in *ICME 2000*, vol. 3, pp. 1619–1622, Jul. 2000.
- [21] N. Wakamiya, M. Murata, and H. Miyahara, "On TCP-friendly video transfer with consideration on application-level QoS," in *ICME 2000*, vol. 2, pp. 843–846, Aug. 2000.
- [22] J. Kim, Y.-G. Kim, H. Song, T.-Y. Kuo, Y. J. Chung, and C.-C. J. Kuo, "TCP-friendly Internet video streaming employing variable frame-rate encoding and interpolation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 10, pp. 1164–1177, Oct. 2000.
- [23] Y. S. Sun, F.-M. Tsou, M. C. Chen, and Z. Tsai, "A TCP-friendly congestion control scheme for real-time packet video using prediction," in *Global Telecommunications Conference, 1999*, vol. 3, pp. 1818–1822, 1999.
- [24] M. Miyabayashi, N. Wakamiya, M. Murata, and H. Miyahara, "MPEG-TFRCP: video transfer with TCP-friendly rate control protocol," in *IEEE International Conference on Communications*, vol. 1, pp. 137–141, Jun. 2001.
- [25] C. Casetti, M. Geria, S. S. Lee, S. Mascolo, and M. Sanadidi, "TCP with faster recovery," in *MILCOM 2000*, vol. 1, pp. 22–25, Oct. 2000.
- [26] W. Stevens, "TCP slow start, congestion avoidance, fast retransmission and fast recovery algorithms," in *RFC 2001*, Feb. 1997.
- [27] J. Mahdavi and S. Floyd, "TCP-friendly unicast rate-based flow control." <http://www.psc.edu/networking/papers/tcp-friendly.html>, Jan. 1997.
- [28] D.-H. Hoang and D. Reschke, "TCP-friendly exponential congestion control for multimedia communication," in *GLOBECOM '02.*, vol. 2, pp. 1586–1590, Nov. 2002.
- [29] S. Hassan and M. Kara, "Performance evaluation of end-to-end TCP-friendly video transfer in the Internet," in *IEEE International Conference on Networks*, pp. 56–61, Oct. 2001.

REFERENCES

- [30] Y.-G. Kim, J. Kim, and C.-C. J. Kuo, "Smooth and fast rate adaptation mechanism for TCP-friendly Internet video," in *Packet video Workshop 2000*, May. 2000.
- [31] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-based congestion control for unicast applications," in *SIGCOMM 2000.*, pp. 43–56, Aug. 2000.
- [32] Y.-G. Kim, J. Kim, and C.-C. J. Kuo, "Network-aware error control using smooth and fast rate adaptation mechanism for TCP-friendly Internet video," in *Ninth International Conference on Computer Communications and Networks*, pp. 320–325, May. 2000.
- [33] Y. Wang and Q.-F. Zhu, "Error control and concealment for video communication: A review," in *Proceedings of the IEEE*, vol. 86, pp. 974–997, May. 1998.
- [34] Y. Wang, S. Wenger, J. Wen, and A. K. Katsaggelos, "Error resilient video coding techniques," *IEEE Signal Processing Magazine*, vol. 17, pp. 61–82, Jul. 2000.
- [35] S. B. Wicker, *Error control systems for digital communication and stroage*. Prentice-hall International, Inc., 1995.
- [36] H. Liu, H. Ma, M. E. Zarki, and S. Gupta, "Error control schemes for networks: An overview," *Mobile Networks and Applications*, vol. 2, no. 2, pp. 167–182, 1997.
- [37] J. Hagenauer, "Rate-compatible punctured convolutional codes (RCPC codes) and their applications," *IEEE Transactions on Communications*, vol. 36, pp. 389–400, Apr. 1988.
- [38] T. Stockhammer, H. Jenkac, and C. Weiss, "Error control for wireless progressive video transmission," in *International Conference on Image Processing. 2002.*, vol. 1.
- [39] C. Weiss, T. Stockhammer, and H. Jenkac, "The far end error decoder with application to image transmission," in *Global Telecommunications Conference, 2001*, vol. 2, pp. 25–29, Nov. 2001.
- [40] L. Wilhelmsson and L. B. Milstein, "On the effect of imperfect interleaving for the Gilbert-Elliott channel," *IEEE Trans. Commun.*, vol. 47, pp. 681–688, May. 1999.

REFERENCES

- [41] J. Cai and C. W. Chen, "FEC-based video streaming over packet loss networks with pre-interleaving," in *Information Technology: Coding and Computing, 2001.*, pp. 10–14, Apr. 2001.
- [42] J. Cai, X. Li, and C. W. Chen, "Layered unequal loss protection with pre-interleaving for progressive image transmission over packet loss channels," in *ICIP 2004*, pp. 2535–2538, Oct. 2004.
- [43] T. Nguyen and A. Zakhor, "Distributed video streaming with forward error correction," in *Packet Video Workshop, Pittsburgh*, Apr. 2002.
- [44] J.-C. Bolot and T. Turetti, "Adaptive error control for packet video in the Internet," in *International Conference on Image Processing, 1996.*, vol. 1, pp. 25–28, Sep. 1996.
- [45] M. Zorzi, R. R. Rao, and L. B. Milstein, "ARQ error control for fading mobile radio channels," *IEEE Trans. Veh. Technol.*, vol. 46, pp. 445–455, May. 1997.
- [46] G. Wang, Q. Zhang, W. Zhu, and Y.-Q. Zhang, "Channel-adaptive error control for scalable video over wireless channel," in *The 7th International Workshop on Mobile Multimedia Communications*, Oct. 2000.
- [47] D. Wu, Y. T. Hou, Y.-Q. Zhang, W. Zhu, and H. J. Chao, "Adaptive QoS control for MPEG-4 video communication over wireless channels," in *IEEE ISCAS 2000*, vol. 1, pp. 28–31, May. 2000.
- [48] S. Choi and K. G. Shin, "A class of adaptive hybrid ARQ schemes for wireless links," *IEEE Trans. Veh. Technol.*, vol. 50, pp. 777–790, May. 2001.
- [49] Q. Zhang and A. kassam, "Hybrid ARQ with selective combining for fading channels," *IEEE Journal on Selected Areas in Communications*, vol. 17, pp. 867–880, May. 1999.
- [50] E. N. Gilbert, "Capacity of a burst-noise channel," in *Bell Syst. Tech. J.*, vol. 39, pp. 1253–1265, Sept. 1960.
- [51] E. O. Elliott, "Estimates of error rates for codes on burst-noise channels," in *Bell Syst. Tech. J.*, vol. 42, pp. 1977–1997, Sept. 1963.

REFERENCES

-
- [52] D. Wu, Y. T. Hou, B. Li, W. Zhu, Y.-Q. Zhang, and H. J. Chao, "An end-to-end approach for optimal mode selection in Internet video communication: theory and application," *IEEE J. Select Areas Commun.*, vol. 18, pp. 977–995, Jun. 2000.
 - [53] Q. Zhang, W. Zhu, and Y.-Q. Zhang, "Network-adaptive rate control with TCP-friendly protocol for multiple video objects," in *IEEE ICME 2000*, vol. 2, pp. 1055–1058, Aug. 2000.
 - [54] H. S. Wang and N. Moayeri, "Finite-state Markov channel: A useful model for radio communication channels," *IEEE Trans. Veh. Technol.*, vol. 44, pp. 163–171, Feb. 1995.
 - [55] K. Sakakibara, "Performance analysis of the error-forecasting decoding for interleaved block codes on Gilbert-Elliott channels," *IEEE Trans. Commun.*, vol. 48, pp. 386–395, Mar. 2000.
 - [56] J. R. Yee and E. J. Weldon, "Evaluation of the performance of error-correcting codes on a Gilbert channel," *IEEE Trans. Commun.*, vol. 43, pp. 2316–2323, Aug. 1995.
 - [57] J. Cai and C. W. Chen, "Robust joint source-channel coding for image transmission over wireless channels," *IEEE Trans. Circuits Syst. Video Technol.*, pp. 962–966, Sept. 2000.
 - [58] K. Stuhlmuller, N. Farber, M. Link, and B. Girod, "Analysis of video transmission over lossy channels," *IEEE J. Select Areas Commun.*, vol. 18, pp. 1012–1032, Jun. 2000.
 - [59] D. Sisalem, "Fairness of adaptive multimedia applications," in *ICC 98*, vol. 2, pp. 891–895, Jun. 1998.
 - [60] P.-H. Hisao, H. T. Kung, and K.-T. Tan, "Streaming video over TCP with receiver-based delay control," *Special Issue on Internet Technology, IEICE Trans. Comm.*, vol. E86-B, no. 2, pp. 572–584, 2003.
 - [61] C. Papadopoulos and G. M. Parulkar, "Retransmission-based error control for continuous media applications," *Proc. 6th Intl. Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, 1996.

REFERENCES

- [62] M. Allman and V. Paxson, "On estimating end-to-end network path priorities," in *ACM SIGCOMM*, Sep. 1999.
- [63] D. Loguinov and H. Radha, "On retransmission schemes for real-time streaming in the Internet," in *IEEE INFOCOM 2001*, vol. 3, pp. 1310–1319, Apr. 2001.
- [64] B. Mukherjee and T. Brecht, "Time-lined TCP for the TCP-friendly delivery of streaming media," in *International Conference on Network Protocols*, pp. 165–176, Nov. 2000.
- [65] M. Podolsky, S. mcCanne, and M. Vetterli, "Soft ARQ for layered streaming media," in *Tech. Rep. UCB/CSB-98-1024*, Nov. 1998.
- [66] P. A. Chou and Z. Miao, "Rate-distortion optimized streaming of packetized media," in *Microsoft Research Technical Report*, Feb. 2001.
- [67] "Website of darwin streaming server." <http://developer.apple.com/darwin/projects/streaming/>.
- [68] "Website of mpeg4ip." <http://mpeg4ip.sourceforge.net/>.
- [69] S. Floyd, M. Handley, and J. Padhye, "A comparison of equation-based and AIMD congestion control." <http://www.icir.org/tfrc/aimd.pdf>, May. 2000.
- [70] M. Zink, C. Griwodz, J. Schmitt, and R. Steinmetz, "Scalable TCP-friendly video distribution for heterogeneous clients," in *MMCN'03.*, pp. 102–113, Jan. 2003.
- [71] L. Gharai and C. Perkins, "Implementing congestion control in the real world," in *IEEE International Conference on Multimedia and Expo*, Aug. 2002.
- [72] L. Gharai, "RTP profile for TCP friendly rate control." Internet draft, IETF. <http://www.ietf.org/internet-drafts/draft-ietf-avt-tfrc-profile-03.txt>, Oct. 2004.
- [73] "Quicktime streaming server modules." <http://developer.apple.com/documentation/QuickTime/QTSS/QTSS.pdf>, Aug. 2003.

REFERENCES

- [74] C. Ladas, R. M. E. Amiee, M. Mahdavi, and G. A. Manson, "Class based selective-ARQ scheme for high performance TCP and UDP over wireless links," in *4th International Workshop on Mobile and Wireless Communications Network*, pp. 311–315, Sep. 2002.
- [75] G. Liebl, M. Wagner, and J. Pandel, "An RTP payload format for erasure-resilient transmission of progressive multimedia streams." Internet draft, IETF. <http://www.ietf.org/internet-drafts/draft-ietf-avt-uxp-07.txt>, Oct. 2004.

Publications

- [1] J. Cai, W. Zhan, and Z. He, "Optimal retransmission timeout selection for delay-constrained multimedia communications," in *IEEE ICIP 2004*, pp. 2035-2038, Oct. 2004.
- [2] W. Zhan, J. Cai, and C. W. Chen, "Conversion of the Gilbert-Elliott Channel to a level of longer data segment," to be submitted.