End-to-End Streaming Protocols with QoS Control for Secure IP Multimedia Communications

by

Siu Ping CHAN

A Thesis Submitted to
The Hong Kong University of Science and Technology
in Partial Fulfillment of the Requirements for
the Degree of Doctor of Philosophy
in Department of Electrical and Electronic Engineering

April 2004, Hong Kong
Authorization

I hereby declare that I am the sole author of the thesis.

I authorize the Hong Kong University of Science and Technology to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize the Hong Kong University of Science and Technology to reproduce the thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Siu Ping CHAN
End-to-End Streaming Protocols with QoS Control for Secure IP Multimedia Communications

by

Siu Ping CHAN

This is to certify that I have examined the above PhD thesis and have found that it is completed and satisfactory in all respects, and that any and all revisions required by the thesis examination committee have been made.

Prof. Yifan HAN (BICH), (Chairman)

Prof. Ted Chi-Wah KOK (ELEC), (Thesis Supervisor)

Prof. Danny H.-K. TSANG (ELEC)

Prof. Albert K. WONG (ELEC)

Prof. Lionel M. NI (COMP)

Prof. Ming-Ting SUN (External Thesis Examiner)

Prof. Khaled BEN LETAIJEF (ELEC), (Head of Department)

Department of Electrical and Electronic Engineering
April 2004, Hong Kong
Acknowledgements

First of all, I would like to thank Professor Ted Chi-Wah KOK. It is my greatest fortune to have him as my advisor. I appreciate so much his insightful inspiration and constant encouragement. He lets me know what exactly the research is and lets me be interested in it. Without him, my graduate study would not be meaningful.

I also want to thank Professors Danny H.-K. TSANG, Albert K. WONG, Lionel M. NI, Ming-Ting SUN, and Khaled BEN LETAIFF. I appreciate their interest in and comments on my work, and their time serving in my examination committees.

I would like to thank and remember all the friends in our research group, for those enjoyable moments which they have been so generous to share with me. To name all of them: Alton Kam-Fai CHAN, Carrson C. FUNG, Hua YU, Ivan Ming-Yan CHAN, Kevin Ying-Man LAW, Min LI, Tim Kam-Tim WOO, Victor Man-Wai KWAN, Ning YAO, and Zhijin WANG.

I wish to express my deepest gratitude to my parents for their disciplines and encouragement throughout my whole life. I would also like to thank my brother and sister for their support in my life. I am always proud to have such a happy and warm family.

Finally and most of all, I am grateful to my dearest, Yan. I would like to thank for all her love and encouragement. She always gives me power to overcome all the difficulties in my study. I thank her for sharing with me good times and bad times. If there is anything that I can give her, it would be my deepest love for her.
# Contents

<table>
<thead>
<tr>
<th>Title Page</th>
<th>i</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authorization Page</td>
<td>ii</td>
</tr>
<tr>
<td>Signature Page</td>
<td>iii</td>
</tr>
<tr>
<td>Acknowledgements</td>
<td>iv</td>
</tr>
<tr>
<td>Table of Contents</td>
<td>v</td>
</tr>
<tr>
<td>Abbreviations and Notations</td>
<td>xi</td>
</tr>
<tr>
<td>List of Figures</td>
<td>xv</td>
</tr>
<tr>
<td>List of Tables</td>
<td>xxi</td>
</tr>
<tr>
<td>Abstract</td>
<td>xxii</td>
</tr>
</tbody>
</table>

1 Introduction  

1.1 Streaming Applications in the Internet: an Overview  

1.1.1 A Taxonomy for Streaming Applications  

1.2 Research Problems
1.2.1 Network Constraints for Streaming System 3

1.3 Solutions 12

1.3.1 Quality Adaptation for Streaming Applications 12

1.3.2 Error Control for Streaming Applications 13

1.3.3 Congestion Control for Streaming Applications 14

1.3.4 Security and Privacy for Streaming Applications 15

1.3.5 Generic Architecture for Multimedia Streaming 16

1.3.6 Streaming Applications and Protocols 17

1.4 Contributions 18

1.5 Dissertation Overview 22

1.6 Publications 24

2 Related Work 27

2.1 Multimedia Quality Adaptation in the Internet 27

2.1.1 Multimedia Bitrate Adaptation 28

2.1.2 Layered Encoding Scheme 29

2.2 End-to-end Congestion Control in the Internet 29

2.2.1 Server-based Congestion Control 29

2.2.2 Client-based Congestion Control 30

2.2.3 TCP Congestion Control and Avoidance 31

2.2.4 TCP-friendliness 33

2.2.5 TCP-friendly Congestion Control with AIMD mechanism 34

2.3 Gateway-assisted Congestion Control in the Internet 38

2.3.1 DropTail 38
2.3.2 Random Early Detection (RED) and its Variants . . . . 39

2.4 Security for Streaming Applications . . . . . . . . . . . . . 39

2.4.1 Simple Key Distribution Scheme . . . . . . . . . . . . . 40

2.4.2 Logical Key Hierarchical Scheme . . . . . . . . . . . . . 41

2.4.3 Subgrouping Scheme . . . . . . . . . . . . . . . . . . . . . 41

3 The End-to-end Architecture . . . . . . . . . . . . . . . . . 43

3.1 Design Principles . . . . . . . . . . . . . . . . . . . . . . . . . 43

3.1.1 Being Adaptive . . . . . . . . . . . . . . . . . . . . . . . . 43

3.1.2 Being Fast to Start . . . . . . . . . . . . . . . . . . . . . . 44

3.1.3 Being Friendly . . . . . . . . . . . . . . . . . . . . . . . . 44

3.1.4 Being Resilient . . . . . . . . . . . . . . . . . . . . . . . . 45

3.1.5 Being Scalable . . . . . . . . . . . . . . . . . . . . . . . . 45

3.1.6 Being Useful . . . . . . . . . . . . . . . . . . . . . . . . . 46

3.1.7 Being Secure . . . . . . . . . . . . . . . . . . . . . . . . . 46

3.2 Design Parameters . . . . . . . . . . . . . . . . . . . . . . . . 47

3.2.1 Flow Control . . . . . . . . . . . . . . . . . . . . . . . . . 47

3.2.2 Multimedia Quality Adaptation . . . . . . . . . . . . . . 47

3.2.3 Loss Recovery . . . . . . . . . . . . . . . . . . . . . . . . 48

3.2.4 End-to-end Congestion Control . . . . . . . . . . . . . . 49

3.2.5 Gateway-assisted Congestion Control . . . . . . . . . . 51

3.2.6 Key Distribution for Secure Multimedia Multicast . . . 52

3.3 The Proposed End-to-end Architecture with QoS Control . . 52
4 Client-based Streaming Protocol 55

4.1 Basic Flow Control .......................... 55

4.1.1 Server-client Communications ................. 55

4.1.2 Packetization ................................ 57

4.1.3 Buffer Control ............................... 59

4.2 Multimedia Bitrate Adaptation .................. 61

4.2.1 Decision of Adaptation ........................ 61

4.2.2 Timeout Mechanism ............................ 63

4.2.3 Adaptive Buffer Design ......................... 63

4.2.4 Markov Analysis of Multimedia Bitrate Adaptation 64

4.2.5 Simulation using NS2 ........................... 68

4.2.6 Results and Discussion ......................... 69

4.3 Loss Packet Recovery .......................... 72

4.3.1 Loss Packet Recovery Mechanism ............... 72

4.3.2 Nonuniform Packet Arrival Mechanism ........... 75

4.3.3 Simulation ..................................... 75

5 Client-based Congestion Control 81

5.1 Client-based Congestion Control ................. 81

5.1.1 Decision of Rate Adaptation .................... 82

5.1.2 Timeout Mechanism ............................. 83

5.1.3 Combination with Multimedia Bitrate Adaptation ... 83

5.1.4 Resource Allocation by Priority Weighting ........ 84

5.1.5 Scalability .................................... 85
6.2.5 Results and Discussion ........................................ 129

7 Security and Privacy for Multimedia Streaming .............. 136
  7.1 Re-keying Cost .................................................. 138
  7.2 Minimum Redundancy Tree (MRT) ............................ 141
    7.2.1 Key Tree Generation ....................................... 142
    7.2.2 Membership Changes ...................................... 143
    7.2.3 Key Tree Update ........................................... 147
    7.2.4 Update Communication Overhead for MRT .............. 149
  7.3 Multiple Minimum Redundancy Tree (Multiple MRTs) ....... 154
    7.3.1 Membership Changes ...................................... 158

8 Conclusions and Future Work ..................................... 160
  8.1 Conclusions .................................................... 160
  8.2 Future Work ................................................... 163

Bibliography ............................................................... 167
# Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACK</td>
<td>Acknowledgement</td>
</tr>
<tr>
<td>AIMD</td>
<td>Additive Increase Multiplicative Decrease</td>
</tr>
<tr>
<td>DiffServ</td>
<td>Differentiated Service</td>
</tr>
<tr>
<td>EWMA</td>
<td>Exponentially Weighted Moving Average</td>
</tr>
<tr>
<td>FTP</td>
<td>File Transfer Protocol</td>
</tr>
<tr>
<td>FIFO</td>
<td>First In First Out</td>
</tr>
<tr>
<td>GC</td>
<td>Group Controller</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hypertext Transfer Protocol</td>
</tr>
<tr>
<td>IETF</td>
<td>Internet Engineering Task Force</td>
</tr>
<tr>
<td>JD</td>
<td>Jitter Detection</td>
</tr>
<tr>
<td>KEK</td>
<td>Key Encryption Key</td>
</tr>
<tr>
<td>LKH</td>
<td>Logical Key Hierarchical</td>
</tr>
<tr>
<td>LDA+</td>
<td>Loss-Delay Based Adaptation</td>
</tr>
<tr>
<td>MRT</td>
<td>Minimum Redundancy Tree</td>
</tr>
<tr>
<td>NS2</td>
<td>Network Simulator 2</td>
</tr>
<tr>
<td>PD</td>
<td>Preferential Dropping</td>
</tr>
<tr>
<td>PSNR</td>
<td>Peak Signal to Noise Ratio</td>
</tr>
<tr>
<td>QoS</td>
<td>Quality-of-Service</td>
</tr>
<tr>
<td>RED</td>
<td>Random Early Detection</td>
</tr>
<tr>
<td>RAP</td>
<td>Rate Adaptation Protocol</td>
</tr>
<tr>
<td>RTCP</td>
<td>Real-time Control Protocol</td>
</tr>
<tr>
<td>RTP</td>
<td>Real-time Transport Protocol</td>
</tr>
<tr>
<td>RLM</td>
<td>Receiver-driven Layered Multicast</td>
</tr>
<tr>
<td>RTT</td>
<td>Round Trip Time</td>
</tr>
<tr>
<td>SEK</td>
<td>Session Encryption Key</td>
</tr>
<tr>
<td>SMTP</td>
<td>Simple Mail Transfer Protocol</td>
</tr>
<tr>
<td>TFRC</td>
<td>TCP-friendly Rate Control</td>
</tr>
<tr>
<td>TCP</td>
<td>Transmission Control Protocol</td>
</tr>
<tr>
<td>UDP</td>
<td>User Datagram Protocol</td>
</tr>
</tbody>
</table>
Notations

\( T_i \)  
Transmission time of Packet i from the server to client
\( PL_i \)  
Playout time of Packet i
\( A_i \)  
Arrival time of Packet i
\( EA_i \)  
Expected arrival time of Packet i
\( J_i \)  
Delay jitter of Packet i
\( T_d \)  
Link transmission delay
\( P_d \)  
Link propagation delay
\( E_d \)  
End-to-end delay
\( S_d \)  
Startup delay
\( W \)  
Waiting time in the queue
\( \tau \)  
Service time in the queue
\( DSeq \)  
Data sequence number
\( PSeq \)  
Play sequence number
\( CSeq \)  
Control sequence number of the control packet
\( SSeq \)  
Server control sequence number counter
\( RSeq \)  
Largest received data sequence number
\( Plist \)  
A list to store the received data sequence number
\( Rlist \)  
A list to store the requested resent data sequence number
\( P(t) \)  
Stored Playtime in the client’s buffer
\( C(t) \)  
Consumption rate
\( CP(t) \)  
Current playout time
\( b(t) \)  
Playtime difference between the original B and the new B in encoding the multimedia streams
\( B \)  
Multimedia Bitrate
\( R \)  
Sending rate
\( \Delta \)  
Multimedia bitrate increment
\( \delta \)  
Sending rate increment
\( ReceivedPackets \)  
Number of packet received by the client
\( FrameSize \)  
Multimedia payload size
\( PacketSize \)  
Multimedia packet size
\( MAX \)  
Adaptive maximum threshold in the client’s buffer
\( MIN \)  
Adaptive minimum threshold in the client’s buffer
\( a \) Raising step in the Client-based Congestion Control
\( b \) Decreasing factor in the Client-based Congestion Control
\( w \) Weight factor for resource allocation in the Client-based Congestion Control
\( B_{\text{max}} \) Maximum buffer size
\( P_{\text{th}} \) Stored playtime threshold value
\( MBA_{\text{th}} \) Predefined threshold time in the Multimedia Bitrate Adaptation
\( \text{EndofMedia} \) Close session packet
\( \text{TIMEOUT} \) Timeout value
\( T_c \) Timer in the client for the Timeout mechanism in the Multimedia Bitrate Adaptation
\( T_s \) Timer in the server for the Timeout mechanism in the Multimedia Bitrate Adaptation
\( T_d \) Timer in the Loss Packet Recovery
\( T_r \) Recent Timer in the Loss Packet Recovery
\( \text{Dropout} \) A counter in the client to measure the multimedia dropout rate
\( TIL \) Time interval length used in the measurement of the NS2 simulations
\( \text{delay} \) Delay encountered by a streaming packet in the JD
\( CBO \) Current buffer occupancy in the JD
\( LC \) Output link occupancy in the JD
\( \text{ave.} \text{delay} \) Average delay encountered by a streaming packet in the JD
\( \text{jitter} \) Delay jitter encountered by a streaming packet in the JD
\( v \) End-to-end delay jitter counter
\( \text{bound} \) Sub-region boundary inside the enqueue region in the JD
\( \text{fixedLth} \) Enqueue region size in the JD
\( \text{threshold} \) Dropping threshold in the JD
\( A \) Delay jitter allowance value in the JD
\( \hat{D} \) Sample mean delay
\( N_s \) Number of sample packets
\( P(\text{drop}) \) Probability of dropping a packet in the JD
\( \sigma_d \) Standard deviation of delay of received multimedia packets
\( J_{\text{max}} \) Maximum allowable delay jitter
\( \alpha_j \) Variance of \( J_{\text{max}} \)
\( \text{UsefulThreshold} \) A threshold to determine whether the received multimedia packet is useful for playback or not
\( N_i \) \( N_i \) layers multimedia traffic
\( \gamma_i \) Layer \( i \) traffic arrival rate
\( \beta_i \) Dropping fraction for layer \( i \) traffic
\( Q_{max} \)  Maximum queue limit
\( \lambda' \)  Total arrival rate of the layer traffic into the queue
\( \mu \)  Service rate of the router
\( p \)  Traffic intensity
\( N_q \)  Average number of packets in the queue
\( W \)  Average waiting time for each packet in the queue
\( \bar{i} \)  Average number of keys needed to be updated for each member
\( \bar{h} \)  Average tree height for each member
\( P_i \)  Probability of leaving group for Member i
\( P_{S_{min}} \)  Sum of probability of leaving subgroup for all members in the subgroup \( S_i \)
\( H_l \)  Entropy of leaving group for members
\( H_l(max) \)  Maximum Entropy of leaving group for members
\( d \)  A key tree with \( d \) branches per node
\( N_t \)  Number of joining requests within a key tree update interval
\( N_o \)  Number of leaving requests within a key tree update interval
\( N_e \)  Number of empty branches in the tree within a key tree update interval
\( N \)  Multicast group size
\( M \)  Subgroup size
\( G \)  Number of subgroups
\( S_i \)  Subgroup i
\( T_i \)  Expected session time for the member \( M_i \)
\( a_c \)  Cost constraint
\( a_o \)  A predefined constant by the pricing scheme of the multicast session
\( a_f \)  A predefined constant by the pricing scheme of the multicast session
\( F_s \)  Subscription fee for each member per unit time to join the multicast session
\( GCS \)  Group controller storage
\( \bar{O} \)  Average update communication overhead for membership change
\( \bar{O}_{l} \)  Expected total update communication overhead for the member leaving events within \( \Delta t \)
\( \bar{O}_{r} \)  Average MRT reconstruction overhead for each member at \( \Delta t \)
\( \bar{O}_{min} \)  Average update communication overhead for minimum update situation in Multiple MRTs
\( \bar{O}_{total,\Delta t} \)  Total expected update communication overhead for each member for each key tree update interval
\( \bar{O}_{total,t} \)  Total expected update communication overhead for each member per unit time
\( \Delta t \)  Key tree update interval for MRT
List of Figures

1.1 End-to-end Delay for a Streaming Packet $i$ ............... 3
1.2 Types of Delay encountered for a Streaming Packet $i$ ........ 6
1.3 Startup Delay for the Streaming System .................. 8
1.4 Delay Jitter for the Streaming System ..................... 9
1.5 Network Topology with Bottleneck Link ...................... 11
1.6 Delay-Constrained Retransmission for the Streaming System ... 14
1.7 End-to-end Architecture for Multimedia Streaming over the Internet 16

2.1 Server-based Congestion Control Mechanism .................. 31
2.2 Client-based Congestion Control Mechanism .................. 32
2.3 AIMD Congestion Control for Multimedia Streaming ............ 34
2.4 Rate Adaptation Protocol (RAP) ............................ 36
2.5 Loss-Delay Based Adaptation (LDA+) .......................... 37

3.1 The proposed End-to-end Architecture with QoS Control for Multimediat Streaming over the Internet .............. 53

4.1 Basic Flow Control Mechanism .............................. 57
4.2 Generic Multimedia Data Packet Format ....................... 57
4.3 Buffer Control Mechanism ........................................ 59
4.4 Multimedia Bitrate Adaptation Mechanism .................. 60
4.5 Multimedia Bitrate Adaptation by Timeout Mechanism ..... 64
4.6 3-state Markov Chain of Multimedia Bitrate Adaptation at a particular time $t$ ........................................ 65
4.7 Multimedia Bitrate $B$ at each time instant for different MAX, $\alpha > \beta$ 67
4.8 Multimedia Bitrate $B$ at each time instant for different MIN, $\alpha > \beta$ 67
4.9 Simulation Topology by NS2 ................................. 68
4.10 Stored Playtime $P(t)$ inside the Client’s Buffer ............ 70
4.11 Multimedia Bitrate $B$ of the Sources, $TIL=0.1s$, (Upper) With Multimedia Bitrate Adaptation, (Bottom) Without Multimedia Bitrate Adaptation ................................. 73
4.12 Loss Packet Recovery Mechanism .............................. 74
4.13 Nonuniform Packet Arrival Mechanism ...................... 76
4.14 Packet Receive Rate for Channel Bandwidth: 56Kbps, 112Kbps, 224Kbps and 448Kbps ................................. 76
4.15 Packet Receive Rate in Lossy Channel with Loss Probability between 0.1-0.9 ........................................ 77
4.16 Packet Receive Rate against Loss Probability ............ 78
4.17 Packet Loss Rate at 56Kbps with Loss Percentage of 25% 79
4.18 Packet Loss Rate at 56Kbps with Loss Percentage of 25% and Bursty Period at 50s-70s ................................. 79
5.1 Client-based Congestion Control Mechanism ................ 82

xvi
6.6  (a) Average Throughput of TCP against Different Packet Sizes,  
      (b) Average Throughput of UDP against Different Packet Sizes  . 113

6.7  (a) Average Throughput of UDP Packets at each Router using the 
      proposed JD Scheme 2, (b) Ratio of Average Throughput between 
      UDP and TCP ................................................................. 113

6.8  Average Delay Jitter for some received UDP Packets using Dif- 
      ferent Queue Management Schemes (a) Original Network with 2 
      TCP Traffic and 2 UDP Traffic, (b) Modified Network with 2 TCP 
      Traffic and 4 UDP Traffic .............................................. 115

6.9  (a) Useful Goodput of UDP Traffic against Different Packet Sizes 
      (Useful Threshold = 0.1s), (b) Useful Goodput of UDP Traffic 
      against Different Useful Threshold (Packet Size = 900 bytes) . . . 117

6.10 (a) Percentage of Useless UDP Packets against Different Packet 
       Sizes (Useful Threshold = 0.1s), (b) Percentage of Useless UDP 
       Packets against Different Useful Threshold (Packet Size = 900 bytes)117

6.11 Audio Quality Comparison between Different Queue Management 
      Schemes ................................................................. 118

6.12 Details of Modified JD scheme for Layered Multimedia Multicast 
      Traffic ................................................................. 121

6.13 Queuing Model of Modified JD for supporting Layered Multimedia 
      Multicast Traffic .......................................................... 122
6.14 (Upper) Average Number of Packets in the Queue against Arrival Rate, $\gamma$, (Bottom) Average Waiting Time in the Queue against Arrival Rate, $\gamma$, Increment Step for $J_{max}=0.5$  

6.15 Upper: Average Number of Packets in the Queue against Traffic Intensity, $p$, Bottom: Average Waiting Time in the Queue against Traffic Intensity, $p$ Increment Step for $J_{max}=0.5$  

6.16 Average Waiting Time in the Queue against Maximum Delay Jitter Allowance, $J_{max}$  

6.17 Simulation Topology for Modified JD Scheme with Layered Multimedia Multicast Traffic  

6.18 Throughput in the Link between R12 and Client Node of Layered Multicast 1, (Upper) by RED, (Bottom) by Modified JD  

6.19 Throughput in the Bottleneck Link, (Upper) by RED, (Bottom) by Modified JD  

6.20 Quality Comparison of Layered Multimedia Multicast Traffic using RED and Modified JD, UsefulThreshold=0.1s  

7.1 The Resultant Key Tree by LKH Approach, with $d = 2$  

7.2 The Resultant Key Tree by MRT Approach, with $d = 2$  

7.3 The Simultaneous Join/Leave Group, Case 1: $N_i = N_o + N_e$  

7.4 The Simultaneous Join/Leave Group, Case 2: $N_i < N_o + N_e$  

7.5 The Simultaneous Join/Leave Group, Case 3: $N_i > N_o + N_e$  

7.6 Member Joining and Leaving Events within each Key Tree Update Interval  

xix
List of Tables

5.1 A Comparison of Average Sending Rate of Multiple Multimedia Streams .................................. 94

7.1 The Key Assignment for each Member $M_i$ .................................................. 143

7.2 A Worst Case Analysis of Different Key Management Schemes ........................................ 159

7.3 A Comparison of Different Costs of the LKH and Multiple MRTs Approaches. .......................... 159
End-to-End Streaming Protocols with QoS Control for Secure IP Multimedia Communications

by

Siu Ping CHAN

Department of Electrical and Electronic Engineering
The Hong Kong University of Science and Technology

Abstract

In recent years there has been a rapid increase in the deployment of multimedia streaming applications such as audio and video broadcasting. However, since the best-effort Internet is an unreliable network with a high packet loss rate and nonuniform packet arrival, it does not provide any QoS control. This is crucial to sustain real-time multimedia traffic.

To resolve the problem, a multimedia bitrate adaptation flow control for streaming multimedia data over the Internet is proposed. It constantly maintains the buffer at a prescribed capacity, even with bursty network loss, by adapting the multimedia bitrate at the streaming encoder. Simulation results showed that the proposed system with multimedia bitrate adaptation can maintain a higher buffer fill-up rate and larger amount of stored playtime, even in bursty loss period, compared to systems without such an adaptation scheme. A loss packet recovery mechanism and a nonuniform packet arrival mechanism are also proposed to provide error recovery for the system and to deal with the out-of-sequence packet
arrival problems. Moreover, a client-based congestion control algorithm able to resolve network congestion problems by adapting the sending rate of the server is presented. Simulation results showed that the proposed client-based congestion control maintains a degree of TCP-friendliness compared to that of the TFRC scheme used in the TCP-friendly congestion control for multimedia traffic. It also provides better resource allocation among different multimedia traffic using a simple weight factor scheme.

Next, a novel gateway-assisted congestion control mechanism called “Jitter Detection” (JD) is described. This improves the QoS in multimedia networking by detecting and discarding useless packets that have accumulated a large enough delay jitter. The JD scheme helps to maintain a high bandwidth for packets within the delay jitter tolerance of the multimedia traffic. The JD can further be used to stream layered multimedia multicast traffic over the Internet in order to preserve the base layer traffic with the best-effort when passing through the gateways. Simulation results have shown that the proposed JD scheme can effectively lower the average received packet delay jitter and increase the goodput of the received packets while maintaining the same TCP-friendliness compared to those using RED and DropTail schemes. The results have also shown that the modified JD scheme can provide better quality in terms of PSNR than that of using RED for layered multimedia multicast traffic.

Lastly, a “Minimum Redundancy Tree” (MRT) is presented for key distribution in secure multimedia multicast, in order to reduce the update communication overhead for re-keying. The MRT is optimal in terms of minimum re-keying costs.
in that it keeps the minimum average number of keys needed to be updated for each member and maintains the minimum average tree height for each member.

The tree update procedure maintains the optimality of the key tree after the re-keying. Analytical analysis of the proposed algorithms is presented and the key tree update interval under constrained network resources is computed. By combining MRT and subgrouping, multiple MRTs can be generated such that the member storage, GC storage and update communication overhead can further be minimized compared to those of other key management schemes.
Chapter 1

Introduction

1.1 Streaming Applications in the Internet: an Overview

Recently, there has been a rapid increase in the deployment of multimedia streaming applications over the Internet. The applications for multimedia streaming involve real-time audio and video playback in which stored, or real-time captured, multimedia content is streamed from a server to a client upon request. Examples include continuous media servers, digital libraries, distance learning systems, shopping and entertainment services. Real-time multimedia streaming as the name implies, has a timing constraint, such that the multimedia data are played continuously. If the multimedia data does not arrive in time, the playback is paused. This is annoying to the audience. Thus, multimedia streaming requires isochronous processing and “Quality-of-Service” (QoS) from the end-to-end point of view. However, today’s Internet does not attempt to guarantee an upper bound on the end-to-end delay, or a lower bound on the available network bandwidth.
As a result, the quality of the delivered service to real-time applications is unpredictable, and difficult to control. Thus, there are many challenges when designing mechanisms and protocols for multimedia streaming over the Internet. To address these challenges, extensive research was conducted in the areas of protocol design, and application layer QoS control including flow control, quality adaptation, congestion control, and error control. The lack of support for QoS has not prevented the rapid growth of real-time streaming applications. This growth is expected to continue, and multimedia traffic will form a higher portion of the Internet load. Thus, the overall behavior of these applications will have a significant impact on the other Internet traffic. These are the reasons why we focused in our research on multimedia networking. Our plan was to design and develop a robust end-to-end multimedia streaming architecture which could compensate for the lack of QoS support of the current Internet for multimedia streaming.

1.1.1 A Taxonomy for Streaming Applications

The real-time transport of live multimedia, or stored multimedia, is the predominant part of real-time multimedia. There are two modes used to transmit the stored multimedia over the Internet, namely, the download mode and the streaming mode (i.e. multimedia streaming). In the download mode, the user downloads an entire multimedia file and then the multimedia file is played by the player. However, full file transfer in the download mode usually suffers from a long and, perhaps, an unacceptable transfer time. In contrast, in the streaming mode, the multimedia content does not need to be fully downloaded, but is played while
parts of the content are being received and decoded. In this dissertation, our concern is mainly the streaming mode which is applicable to both finite and infinite size multimedia data, where downloading an entire multimedia stream is not possible in the infinite multimedia size case.

1.2 Research Problems

1.2.1 Network Constraints for Streaming System

In general, a streaming system has several network constraints such as end-to-end delay, startup delay, delay jitter, bandwidth and packet loss. QoS control mechanisms such as quality adaptation, congestion control and error control are carried out according to current network conditions so as to assure the quality of the delivered streams. In this section we discuss those network constraints and address the corresponding QoS control mechanisms.

![Diagram showing end-to-end delay](image)

PLi = **Playout time of Packet i**
Ai = **Arrival time of Packet i**
Ti = **Transmission time of Packet i**

Figure 1.1: End-to-end Delay for a Streaming Packet i
End-to-end Delay

Multimedia streaming requires a bounded end-to-end delay so that multimedia data packets can arrive at the client in time to be decoded and played. Figure 1.1 shows the definition of end-to-end delay for a streaming system. \( T_i \) is the transmission time of Packet \( i \) from the server. \( PL_i \) is the playout time of Packet \( i \) in the player. \( A_i \) is the arrival time of Packet \( i \). If Packet \( i \) arrives in time, i.e. it arrives before its playout time \( PL_i \), it is used for the playback. However, if Packet \( i \) does not arrive in time, i.e. it arrives later than \( PL_i \), it cannot be used for the playback, and this causes a multimedia dropout problem. The outcome of the multimedia dropout problem is a degradation of the multimedia playback because the streaming buffer does not have enough playtime in which the content can be played. Furthermore, if the multimedia packet arrives too slowly and beyond a delay bound, then it will not be used in real-time playback. As a result, such packets are rendered useless even though they have successfully arrived at the client. Such multimedia data packets are a waste of network resources, and create a congested network.

We will now discuss several types of delay that contribute to the end-to-end delay for a streaming packet transferring from the server to the client. Figure 1.2 shows the types of delay that the streaming packets encounter. Figure 1.2 shows the routing path for a streaming packet. The link between Router 1 and Router 2 has the minimum bandwidth along the routing path. This is called the "bottleneck link".

1. Link Transmission Delay and Link Propagation Delay: Link Transmission
Delay $T_d$ for a streaming packet is the time that it takes for the packet to transmit over a transmission link. The Link Transmission Delay $T_d$ depends on the packet size $PacketSize$ of the streaming packet and the link capacity $LinkCapacity$, i.e.

$$T_d = \frac{PacketSize}{LinkCapacity}. \quad (1.1)$$

Therefore, if the packet size is fixed and the link capacity is reduced in the bottleneck link, the Link Transmission Delay $T_d$ is increased. The Link Propagation Delay $P_d$ is the time it takes for one bit of the packet to transfer over a transmission link. The Link Propagation Delay $P_d$ depends on the propagation speed. This is equal to the speed of light $c$ and the length of link $l$, i.e.

$$P_d = \frac{l}{c}. \quad (1.2)$$

As shown in Figure 1.2, all the links have the same length. So, the Link Propagation Delay $P_d$ is the same.

2. Waiting Time and Service Time: The Waiting Time $W$, or the Queuing Time, is the time a packet waits in the queue inside the switch before it gets the service. The Waiting Time $W$ depends on the number of packets in the queue. The Service Time $\tau$ is the time it takes for the switch to process a packet, and this depends on the processing power of the switch. Thus, the Total Delay $S_d$ for a packet staying inside the switch is the sum of the Waiting Time $W$ and the Service Time $\tau$, i.e.

$$S_d = W + \tau. \quad (1.3)$$
Figure 1.2: Types of Delay encountered for a Streaming Packet $i$
3. End-to-End Delay: End-to-end delay $E_d$ is the time it takes for the server to transmit one packet to the client over the packet-switch network. The end-to-end Delay $E_d$ depends on the Link Transmission Delay $T_d$, Link Propagation Delay $P_d$, the Total Delay $S_d$ for a packet staying inside the switch (the sum of Waiting Time $W$ and Service Time $\tau$). Moreover, the end-to-end Delay $E_d$ also depends on the number of network nodes and the number of links along the routing path for a multimedia packet. As discussed before, multimedia streaming requires bounded end-to-end delay so that multimedia data packets can arrive at the client in time to be decoded and displayed.

To remedy the problem of end-to-end delay variations, data buffer of a reasonable size should be allocated within the client’s system in order to avoid a deleterious effect on the streaming performance.

**Startup Delay**

This is the time it takes for a multimedia player to buffer up enough multimedia content and start the playback to the users. Figure 1.3 shows the startup delay for two streaming systems. It shows the stored playtime $P(t)$ in the client’s buffer. There are two streaming systems with different buffer fill-up rates. It can be shown that the streaming system with the higher buffer fill-up rate has a smaller startup delay. The startup delay for a streaming system is prompted to be as small as possible because users do not want to wait a long time in order to enjoy the multimedia sessions. However, typically, the arrival of packets is based on
network conditions. For a fixed available network bandwidth, the buffer fill-up rate is always higher when using a lower multimedia bitrate bitstream than when using a higher multimedia bitrate bitstream. A higher buffer fill-up rate means that the presentation of the multimedia content is faster, and that the startup delay is lower. Also, it can provide an ability for the streaming system to recover from an inadequate playtime situation and to prevent the multimedia dropout. As a result, for a fixed available network bandwidth, it is preferable to transmit a relatively lower quality bitstream or a lower multimedia bitrate bitstream. In fact, there is a need for a flow control mechanism which can ensure a higher buffer fill-up rate for the client’s buffer.
Delay Jitter

As discussed before, multimedia streaming requires a bounded end-to-end delay in order to provide a continuous and smooth multimedia playback to users. However, the end-to-end delay varies with the network conditions. Therefore it is unpredictable and difficult to control. If the end-to-end delay is not bounded, it causes a delay jitter problem. Figure 1.4 shows the packet arrival time in the client’s buffer. $PL_i$ is the playout time of Packet $i$, and $A_i$ is the arrival time of Packet $i$ to the client’s buffer. $EA_i$ is the expected arrival time of Packet $i$. This is linearly related to the end-to-end delay of a streaming packet transferred from the server to the client. The delay jitter is defined as the difference between the arrival time $A_i$ and the expected arrival time $EA_i$, i.e.

$$J_i = |A_i - EA_i|.$$  

(1.4)
The delay jitter problem complicates the synchronization problem between packets from a single media stream, or between packets from different media streams. Too much accumulated delay jitter when the multimedia stream traverses the network renders the stream useless when received by the clients and this leads to a degradation in the QoS. This is because it is difficult to re-adjust the timing relationships between multimedia packets from the same, or several, media streams so as to ensure a synchronized playback of information [19]. Removing delay jitter requires collecting packets and holding them long enough to allow the slowest packets to arrive in time to be played in the correct sequence. This causes additional playout delay. The two conflicting goals of minimizing delay and removing delay jitter have engendered various schemes capable of adapting a delay jitter buffer size that matches the time-varying requirement of the network delay jitter removal. This adaptation has the explicit goal of minimizing the size and delay of the delay jitter buffer, while at the same time preventing the buffer underflow caused by delay jitter. The other way to tackle the delay jitter problem is to use gateway-assisted congestion control. This can help to detect, and discard, those packets which have accumulated too much delay jitter in order to resolve network congestion.

**Bandwidth**

To ensure an acceptable quality of playback, a streaming application, typically, has a bandwidth requirement. The multimedia playback can only be achieved when there is sufficient available bandwidth for this streaming application in the
network. However, the current Internet does not provide bandwidth reservation to support this requirement. The situation worsens when there is a bottleneck link along the routing path of the streaming packet, as discussed. Figure 1.5 shows the typical "dump bell" network topology with a bottleneck link. Figure 1.5 shows that there is a multimedia traffic being transferred over the network from the server to the client. But, there is also another network traffic sharing the same network. When the packets pass through the router of the bottleneck link, they queue up inside the buffer of the router. A congestion occurs in the bottleneck link, and this overflows the buffer of the router. This causes packets to be dropped. Usually, packet loss is due to transmission error and packet dropping in the routers. The packet loss causes degradation in the quality of the multimedia playback. Therefore, it is desirable for multimedia streaming
applications to employ congestion control and quality adaptation so as to avoid
the congestion which happens in the heavily loaded network. For multimedia
streaming, the congestion control takes the form of rate control: that is, the
sending rate is adapted according to the available bandwidth in the network. The
quality adaptation ensures a suitable quality of bitstream according to current
network conditions.

1.3 Solutions

1.3.1 Quality Adaptation for Streaming Applications

Quality adaptation adjusts the quality of the multimedia streams with long term
changes in bandwidth. There are two attractive approaches which can be used to
adjust the quality of a multimedia stream on-the-fly, i.e., (1) Adaptive Encoding
Scheme and (2) Layered Encoding Scheme. In the adaptive encoding approach,
the resolution of encoding scheme is adjusted by changing the quantization factor
of the encoder based on the network feedback.

In the layered approach, the stream is encoded in a hierarchical fashion [3].
As the available network bandwidth changes, the server adjusts and encodes the
number of layers accordingly. The use of this approach can change the quality
over a wide range. It is also suitable for communications between multiple clients.
The layered approach usually has the decoding constraint where a particular
enhancement layer can only be decoded if the base layer and all the lower quality
enhancement layers have received by the clients.
1.3.2 Error Control for Streaming Applications

Error control is used for recovery of lost packets. Although multimedia streaming applications require quality rather than reliability, it is necessary to keep the packet loss rate below a certain quality threshold. The error control mechanism is different from that of traditional reliable “Transmission Control Protocol” (TCP) communications, in that, for multimedia streaming, the retransmission of a packet is only done if the retransmitted packet arrives no later than its delay bound. The server should allocate a portion of the available network bandwidth for the retransmission of lost packets. The higher the loss rate, the larger the bandwidth requirement for the recovery. The recovery bandwidth is used in retransmission [4], or to add redundancy, e.g., Forward Error Correcting (FEC) [4]. The retransmission has a lower bandwidth requirement, and can be performed selectively, but it results in a higher delay. In contrast, adding redundancy requires a higher average bandwidth requirement at the expense of a shorter recovery delay. In multimedia streaming, delay-constrained retransmission is the most suitable choice for providing loss packet recovery to unicast multimedia traffic [3].

Figure 1.6 shows the basic idea of the delay-constrained retransmission for the multimedia traffic. Suppose that the server sends out three streaming packets to the client. If Packet 2 is lost, the client detects this and asks for the retransmission of Packet 2 from the server. The retransmission of Packet 2 is only carried out when the expected arrival time is smaller than that of the playout time of Packet 2 in the client.
1.3.3 Congestion Control for Streaming Applications

Congestion control is used to resolve the network congestion problem and determines the most suitable sending rate for the multimedia streams based on the network statistics. A dominant portion of today’s Internet traffic consists of a variety of TCP-based flows. Thus, TCP-friendly behavior is important. In the absence of any resource management mechanism in the Internet, “Additive Increase Multiplicative Decrease” (AIMD) [12] seems to be the most promising, and most well-understood, rate adaptation algorithm. The goal of AIMD is to hunt for a fair share of the network bandwidth. While it can rapidly adapt to any change in network traffic, it inherently oscillates around an optimal rate in equilibrium. An alternative approach is to use an analytical model of TCP behavior
for rate adjustment [12]. This approach exhibits smoother variations in bandwidth. However, it is unable to quickly adapt to sudden changes in background traffic.

1.3.4 Security and Privacy for Streaming Applications

Multimedia streaming applications usually involve one-to-many communications such as multimedia conferencing, interactive group game and video-on-demand system, etc. IP multicast is an efficient communication mechanism for group-oriented applications. Bandwidth is saved because the source traffic is sent on a multicast tree that spans all members of the multicast group. However, the IP multicast is not a secure network protocol because it does not provide any provision for restricting the delivery of data to a specified set of clients. To achieve a secure delivery, the server encrypts messages with a “Session Encryption Key” (SEK) previously distributed to all the valid group members, so that members can decrypt the messages accordingly. When there are changes in group membership (members joining or leaving), the “Group Controller” (GC) distributes a new set of SEK to those group members associated with the membership changes in order to protect the security of past, present, and future communications. This is known as re-keying. Therefore, a robust and efficient key management scheme is needed for the re-keying process so that the re-keying costs, including member storage, Group Controller (GC) storage, and update communication overhead, can be minimized.
1.3.5 Generic Architecture for Multimedia Streaming

Figure 1.7 depicts a generic end-to-end architecture for multimedia streaming over the Internet. The quality adaptation and error control modules are responsible for providing the best quality of multimedia content and the resilient ability of the streaming system, respectively. The adaptation of multimedia quality is done by changing the multimedia bitrate $B$, or the encoded bitrate, of the multimedia stream in the encoder. Moreover, the congestion control module regulates the sending rate $R$, or the transmission rate, of the multimedia traffic based on the packet loss statistics from the error control module. In fact, this end-to-end architecture is generic for any existing streaming system. Any modifications to the module can be plugged directly into this architecture without affecting the other modules. Our research focused was to design and develop those QoS control
modules to ensure a better performance in managing the multimedia streams than those existing solutions.

1.3.6 Streaming Applications and Protocols

A robust streaming protocol is required to achieve a continuous playback of multimedia content over the Internet with short delay after the starting of the downloading of the multimedia content. The streaming protocol provides services such as transport, and QoS control mechanisms including quality adaptation, congestion control and error control. The streaming protocol is built on the top of the network level protocol and the transport level protocol.

The multimedia streaming protocol is based on the IP network and, in particular, "User Datagram Protocol" (UDP) is mainly used, despite of some streaming applications still using TCP. Like TCP, UDP is a transport layer protocol. But, unlike TCP, UDP is a connectionless transport protocol. UDP does not guarantee a reliable transmission and an in-order arrival of packets. Under UDP, there is no guarantee that a packet will arrive at its destination. The UDP packets may get lost in the network when there is a lot of traffic. For example, the packets may get discarded because of a gateway buffer overflow. Therefore, UDP is not suitable for data packet transfer where guaranteed delivery is important. UDP, however, is the ideal transport layer protocol for streaming applications in which the priority is to transfer the packets from the source to the destination. Unlike TCP, UDP does not contribute any additional delay, which is the result of the retransmission of lost packets. Therefore, UDP does not lead to have an
unbounded end-to-end delay for the streaming packets.

As UDP does not guarantee packet delivery, the client needs to rely on “Real-time Transport Protocol” (RTP) to detect packet loss. RTP is an Internet standard protocol designed to provide end-to-end transport functions for supporting real-time application [6]. “Real-time Control Protocol” (RTCP) is a companion protocol with RTP and is designed to provide QoS feedback to the participants of a RTP session. In summary, RTP is a data transfer protocol while RTCP is a control protocol.

1.4 Contributions

We have discussed the existing research challenges for multimedia streaming and the corresponding solutions. In this dissertation, we have proposed many robust and novel solutions to each of the research problems of multimedia streaming. In the following the major contributions are listed.

1. End-to-end Streaming Architecture

We have designed a streaming-friendly end-to-end architecture that combines flow control, quality adaptation, error control, end-to-end and gateway-assisted congestion control and security designs for any streaming application in the Internet. We believe such an architecture can be used as a generic architecture for multimedia streaming applications.

2. Client-based Flow Control

In this dissertation, a multimedia-on-demand system that provides real-time
playback of multimedia data over UDP/IP is presented. We designed an adaptive buffer to overcome the multimedia dropout problems. A multimedia bitrate adaptation flow control, which constantly maintains the buffer at a predefined capacity, is discussed. The multimedia bitrate adaptation algorithm is considered to be an end-to-end mechanism in the application level. It constantly maintains the buffer at a prescribed capacity, even with bursty packet loss in the network, by adapting the multimedia bitrate $B$ at the streaming encoder. Together with the proposed client-based congestion control mechanism presented below, the network congestion problems can be reduced while maintaining a certain degree of TCP-friendliness. Simulation results obtained using “Network Simulator 2” (NS2) [21] showed that a higher buffer fill-up rate and a larger amount of stored playtime $P(t)$ can be achieved during bursty loss period compared to those of systems without such a flow control mechanism. This demonstrated the efficiency of the proposed system.

3. Client-based Congestion Control

A client-based congestion control algorithm, thought of as being located in a lower level than the client-based flow control mechanism, is also presented. It works together with the flow control mechanism above to resolve network congestion problems while maintaining a certain degree of TCP-friendliness, compared to that of the TFRC scheme used in TCP-friendly congestion control for the multimedia streaming. It can be done by adapting the sending rate $R$ of the server. Simulation results obtained from NS2 [21]
have shown that better resource allocation can be obtained by using a simple weight factors scheme, and an overall increase in the average sending rate. Hence, better quality of streaming media is observed.

4. Multimedia Streaming Gateway with Jitter Detection

A novel active buffer management scheme, called “Jitter Detection” (JD) for gateway-assisted congestion control to stream multimedia traffic in packet-switched networks, is presented. The quality of multimedia presentation can be greatly degraded due to end-to-end delay variation, or delay jitter, when transported over a packet-switched network. Packets received by the client are rendered useless if they have accumulated a large enough delay jitter. The proposed JD scheme improves the QoS in multimedia networking by detecting and discarding packets that have accumulated a large enough delay jitter. As a result, a high bandwidth is maintained for packets that are within a multimedia stream’s delay jitter tolerance. Simulation results showed that the proposed JD scheme can effectively lower the average received packet delay jitter. It can increase the goodput of the received packets compared to that of “Random Early Detection” (RED) [15], and DropTail used in traditional gateway-assisted congestion control schemes. Furthermore, simulation results have also revealed that the proposed JD scheme can maintain the same TCP-friendliness compared to that of RED, and DropTail, used for multimedia streams.

A modified JD scheme for gateway-assisted congestion control is used to stream layered multimedia multicast traffic over the Internet, in order to preserve the base layer traffic with the best effort when passing through the gateway. The priority of dropping decisions is given to higher level layers (enhancement layers) multicast traffic. Such a feature is not included in popular schemes such as RED and DropTail. Queuing analysis was performed to analytically evaluate the efficiency of the proposed scheme. Simulation results obtained using NS2 [21] are presented and those show that the proposed scheme can provide better quality than that using RED and DropTail for gateway-assisted congestion control.

6. Minimum Redundancy Key Management Scheme for Secure Multimedia Multicast

Multimedia streaming traffic is carried within a secured multimedia multicast system in which the multimedia data is only available to a group of valid group members. A robust key management scheme is often used in this kind of situation to manage and securely distribute keys to valid group members. However, when the membership of such a group changes, the key has to be updated in order to protect the security of the multicast session. This costs a significant update communication overhead.

In this dissertation we present an algorithm that generates a “Minimum Redundancy Tree” (MRT) for key distribution in secure multimedia multicast. The use of this algorithm alleviates the problem. The MRT is optimal in
terms of minimum re-keying costs as it keeps the minimum average number of keys needed to be updated for each member, and maintains the minimum average tree height for each member. The tree update procedure maintains an optimality of the key tree after the re-keying. Analytical analysis of the proposed algorithms is presented and the key tree update interval under constrained network resources is also computed in this dissertation. By combining MRT with subgrouping, multiple MRTs can be generated, such that the member storage, GC storage and update communication overhead can be further minimized compared to that of other key management schemes.

1.5 Dissertation Overview

This dissertation is organized as follows:

In Chapter 2, the related work is reviewed and some of the differences between the previous work and the work presented in this dissertation are discussed.

A high level architectural view for the design of streaming applications in the Internet os provided in Chapter 3. Addressing design principles for Internet applications leads us to identify multimedia quality adaptation, congestion control including both end-to-end and gateway-assisted approaches, error control and security issues for any streaming applications in the Internet. We briefly explore the design for each one of these components and select an appropriate mechanism from its corresponding space. Then we discussed the composition of all
the key components into a coherent architecture and describe the interaction between these components. We also argue that the architecture can be viewed as a generic architecture for streaming applications as long as the different modules are properly integrated.

In Chapter 4 we present the client-based flow control mechanism for a server-client multimedia streaming system. Multimedia bitrate adaptation, which provides an overall quality improvement for the multimedia streams, is also addressed.

The client-based congestion control mechanism used for end-to-end congestion control purposes in the multimedia streaming traffic is presented in Chapter 5. It can also provide resource allocation between different multimedia streams using a simple weight factor scheme. The details of the mechanisms will be addressed in this chapter.

In Chapter 6 we discuss the proposed JD which can detect and discard the multimedia packets that have already accumulated a large amount of delay jitter value when being transported through the network. The details of the JD will be investigated in this chapter. We also discuss the further extension of the mechanism for application in layered multimedia multicast traffic.

The proposed MRT for a key management scheme for secure multimedia multicast is addressed in Chapter 7. The details of MRT, including MRT generation, and tree update when there are membership changes, are discussed. Advantages of using MRT compared to advantages of using other traditional key management schemes, will be addressed. The extension of MRT to Multiple MRTs can further
minimize the re-keying cost, and this will be discussed in detail in this chapter.

In Chapter 8 a conclusion is given and future plans are discussed.

1.6 Publications

Accepted Journal Publication


Journal Publications under review


24
Accepted Conference Publications


**Submitted Conference Publications**


Chapter 2

Related Work

2.1 Multimedia Quality Adaptation in the Internet

Multimedia streaming applications require real-time playback, i.e., the multimedia content is played on-the-fly while the client is still receiving new content. As a result, the multimedia content must be received faster than the rate at which the content on the client is being played. The real-time playback of multimedia content can be achieved by compressing the multimedia at a ratio that is sufficient to transmit the data at least fast enough to permit the client system to generate continuous playing during receipt of the compressed data. If this does not occur, a multimedia dropout problem, due to an inadequate amount of stored playtime inside the client’s buffer, is observed. It is a phenomena wherein the multimedia playback terminates for some noticeable time period, and resumes after this delay.
Since the Internet is a shared environment, the available bandwidth for streaming in a particular instant depends on the number of services being provided during that time period. Therefore, the multimedia system should be scalable, such that the output multimedia bitrate or encoded bitrate of the compressed multimedia data can be adjusted to cope with network bandwidth fluctuations. If the multimedia data is transmitted at too high a bitrate, a longer than real-time is required in order to complete the transmission of the multimedia data across the network. This results in multimedia dropout. On the other hand, if the multimedia data is transmitted at too low a bitrate, the playback quality suffers. Scalable multimedia encoding systems can be roughly classified into the following categories according to the granularity of the multimedia quality adaptation.

2.1.1 Multimedia Bitrate Adaptation

To cope with the time-varying bandwidth of the Internet, the compression system encodes the multimedia content to several different multimedia bitrate values, or even encodes the multimedia content with a desired multimedia bitrate determined on-the-fly [3]. Such a multimedia bitrate adaptation scheme has an advantage of ensuring optimal quality with a specified bitrate. However, it is a time-consuming and computationally expensive procedure. Furthermore, the extra bitrate, if impossible, is difficult to appreciate. Instead, the multimedia bitrate adaptation is usually given with a degree of granularity. In other words, the multimedia bitrate can be adjusted in terms of $\pm k\Delta B$ where $\Delta B$ is the multimedia bitrate adaptation step size, and $k$ is a positive integer.
2.1.2 Layered Encoding Scheme

The layered encoding system compresses the multimedia content and generates multiple substreams as output. One of the compressed substreams, the “base” layer bitstream, can be independently decoded and used for playback with coarse quality multimedia content. Other compressed substreams are “enhancement” layer substreams. These can only be decoded together with the “base” layer and lower “enhancement” layer substreams. The “enhancement” layers improve the quality of the multimedia playback. The more times the “enhancement” layers can be decoded, the better the quality of the multimedia playback. Although the layered encoding scheme can be employed in both unicast and multicast, it is today mainly used in multicast. The “Receiver-driven Layered Multicast” (RLM) by [22], a well-known quality adaptation, uses a layered encoding scheme for multicast multimedia communications. In RLM, the client, based on its own network available bandwidth, determines how many layered multicast groups to join. The more layered multicast groups to join, the better the multimedia quality.

2.2 End-to-end Congestion Control in the Internet

2.2.1 Server-based Congestion Control

Server-based congestion control is initiated by the server according to current network situations. Several parameters such as end-to-end delay, delay jitter,
bandwidth and packet loss rate are focused on in the server-based congestion control mechanism. The server uses statistical results from the client feedback report to calculate the required network information, as is described in [55, 56, 57]. For example, the “Round Trip Time” (RTT) and the delay jitter are calculated using the timestamp of the feedback report. The sequence number of each packet is used for packet loss detection. The estimated RTT is also used for estimating the available network bandwidth. Server-based congestion control has an advantage in that it forms a closed-loop network monitoring system. So the obtained network statistics are more accurate. A more precise decision in regard to the sending rate of multimedia traffic can be obtained to suit current network conditions. Also, the server-based streaming mechanism is well-developed and is supported by RTP/RTCP. However, if the feedback information is lost or delayed due to network congestion, the server is not be able to react immediately, and this may cause the network to become even more congested.

### 2.2.2 Client-based Congestion Control

In client-based congestion control, clients directly collect and analyze the network statistics and send the response commands back to the server to resolve any congestion detected. The client can request congestion control from the server by monitoring the packet loss rate. As discussed in Section 2.1.2, RLM [22] is one example of client-based rate control in multicast communications. The clients, based on their available network bandwidth, decide which layered multicast groups to join. Clients in RLM do not send back the response command to
the server. Advantages of using client-based congestion control are that it can ensure a fast response to network congestion, and that it is scalable with regard to multiple clients. It requires less feedback traffic than that needed in server-based congestion control, and causes less burden to the network. The design of a client-based congestion control is simple as it can be built on top of RTP/RTCP in the network. However, the network statistics for client-based congestion control are less accurate than those in server-based congestion control as these are obtained using an open-loop network monitoring process.

2.2.3 TCP Congestion Control and Avoidance

TCP is a connection-oriented unicast protocol that offers reliable data transfer as well as flow and congestion control. TCP maintains a congestion window that
controls the number of unacknowledged data packets in the network. Sending data consumes slots in the window of the server, and the server sends packets only when free slots are available. When an "Acknowledgement" (ACK) for those packets is received, the window is shifted so that the acknowledged packets leave the window, and the same number of free slots becomes available. On startup, TCP performs slowstart, in which the rate doubles during each RTT to quickly gain its fair share of the bandwidth. In a steady state, TCP uses an AIMD mechanism so as to detect additional bandwidth and to react to congestion. When there is no indication of loss, TCP increases the congestion window by one slot per RTT. When there is packet loss, indicated by a *TIMEOUT*, the congestion window is reduced to one slot and TCP re-enters the slowstart phase. Packet loss
indicated by three duplicate ACKs results in a window reduction to half that of its previous size.

2.2.4 TCP-friendliness

Nowadays, most of the traffic in the Internet uses TCP-based protocols [12] such as “Hypertext Transfer Protocol” (HTTP), “Simple Mail Transfer Protocol” (SMTP), or “File Transfer Protocol” (FTP). However, because multimedia streaming applications do not integrate TCP-compatible congestion control mechanisms; they treat competing TCP traffic in an unfair manner. When there is network congestion, all TCP traffic reduce their sending rate in order to resolve the congestion. The multimedia traffic continue to send at their original rate. This eventually causes starvation of the TCP traffic. Therefore, it is desirable to define appropriate rate adaptation rules and mechanisms for multimedia traffic that are compatible with the rate adaptation mechanism of TCP. The rate adaptation, or the congestion control mechanisms, for multimedia traffic should be TCP-friendly so as to ensure a fair distribution of bandwidth between TCP traffic and multimedia traffic.

Multimedia traffic are said to be TCP-friendly when the long term throughput does not exceed the throughput of a co-existent TCP connection under the same conditions [13]. Unicast traffic is considered TCP-friendly when it does not reduce the long-term throughput of any co-existent TCP traffic more than another TCP traffic on the same path would under the same network conditions. Multicast traffic is said to be TCP-friendly when, for each server-client pair, the multicast
traffic has the property of being unicast TCP-friendly.

2.2.5 TCP-friendly Congestion Control with AIMD mechanism

TCP-friendly congestion control for multimedia streaming usually achieves TCP-friendliness by dynamically adapting the sending rate according to a network feedback mechanism that indicates congestion. The most common TCP-friendly congestion control mechanisms for multimedia streaming use the idea of AIMD which is similar to TCP congestion control. This results in a rate that displays the typical sawtooth-like behavior of TCP congestion control, as shown in Figure 2.3. Figure 2.3 shows the change in the sending rate $R$ for a multimedia traffic
when an AIMD congestion control mechanism is used. When there is no packet loss, or when the packet loss rate is smaller than, or equal to, a certain threshold, i.e., $Loss \leq Th$, the sending rate $R$ is linearly increased. However, when there is packet loss, or when the packet loss rate is greater than a certain threshold, i.e., $Loss > Th$, the sending rate $R$ is decreased multiplicatively, usually to one half of its original value.

Several TCP-friendly congestion control mechanisms using AIMD are widely adopted in streaming applications. These are discussed in details as follows:

1. The “Rate Adaptation Protocol” (RAP) presented in [9] is a simple AIMD scheme for unicast flows. Each data packet is acknowledged by the client. The ACKs are used to detect packet losses and to calculate the RTT. When RAP experiences congestion, it halves the sending rate. In periods without congestion, the sending rate increases by one predefined $PacketSize$ per RTT. The mechanism is similar to that of the TCP AIMD congestion control algorithm. As a result, it can maintain certain TCP-friendliness. Figure 2.4 details the RAP congestion control algorithm. The decision on rate change is made once per RTT.

2. The “Loss-Delay Based Adaptation” (LDA+) [7], another AIMD variant, is based on the feedback report by RTCP. Figure 2.5 details the LDA+ congestion control algorithm. The increasing and decreasing factors of AIMD mechanism within LDA+ are dynamically adjusted according to network conditions. If there is congestion, the sending rate $R$ is changed to $R = \max(R' \ast \sqrt{loss}, R_{tcp})$ where is $R'$ is the previous value of $R$, $loss$ is
the estimated packet loss rate, and $R_{tcp}$ is the sending rate of other TCP connections sharing the same link. This is determined using a predefined analytical model. The decreasing factor is based on the number of packets lost during congestion. If there is no congestion, the sending rate $R$ increases from $R$ to $R' + A$, where $A$ is the increase value.

3. The “TCP-friendly Rate Control Protocol” (TFRC) [13], specific to unicast communication, can be adapted to multicast streaming with some modifications. The TFRC increases the sending rate $R$ of the multimedia traffic to a fair share of the bandwidth. The increment of the sending rate $R$ ceases when there is packet loss detected. In cases of packet losses, TFRC behaves like TCP congestion control and the sending rate is reduced by a specific value, usually to one half of its original value, like TCP congestion.
control mechanism as discussed in Section 2.2.3. The client updates its parameters once for each RTT and sends a feedback report to the server. The server then computes a new fair rate from these parameters and adjusts the sending rate $R$ accordingly. A major advantage of TFRC is that it can achieve a relatively stable sending rate $R$ while still provide a sufficiently fast response to other competing TCP traffic.
2.3 Gateway-assisted Congestion Control in the Internet

In the previous section we discussed some end-to-end TCP-friendly congestion control mechanisms for multimedia traffic that help to resolve network congestion problems with certain TCP-friendliness to other TCP traffic. However, it should be noted that congestion problems are caused by the bottleneck links between gateways. Therefore, we developed a gateway-assisted congestion detection and resolution protocol for multimedia streams that ensures a fair allocation of network resources to “useful” multimedia traffic. The performance of end-to-end congestion control is expected to be greatly improved with the deployment of the gateway-assisted congestion control mechanism. The following active queue management schemes are widely employed in gateway-assisted congestion control for both TCP traffic and multimedia traffic.

2.3.1 DropTail

“First In First Out” (FIFO) queuing with the DropTail policy is widely used in the current Internet gateways in which incoming packets are scheduled in a FIFO manner and newly arrived packets are discarded when the gateway buffer is full. The DropTail is simple, scalable, and is easy to implement. However, it leads to a global synchronization problem when reducing the sending rate of TCP traffic by discarding newly arrived TCP packets, especially in bursty conditions.
2.3.2 Random Early Detection (RED) and its Variants

To address the problems incurred when using DropTail, RED was proposed by the “Internet Engineering Task Force” (IETF). RED is an active queue management scheme in which the incoming packets are dropped randomly with a probability that is related to the current queue length and a queue-size threshold value. Typically, RED does not provide any protection for TCP flows when there are also UDP traffic sharing the queue. UDP traffic typically lowers the throughput of TCP flows. The application of RED to achieve TCP-friendly UDP connection has been a popular research subject in RED. In [16], several variations of RED, and even packet size, are taken into consideration in order to achieve better loss differentiation and fairness. In [17], average delay instead of average queue length is used to calculate the dropping probability in RED. In [18], “Preferential Dropping” (PD) is proposed to identify individual mis-behaving high bandwidth flows such that the rate of these flows are compressed to certain target values.

2.4 Security for Streaming Applications

As discussed before, to achieve secure multicast for multimedia streaming, the server encrypts the messages with a SEK that has been distributed to all the valid group members so that the members can decrypt the messages accordingly. If there are any membership changes in the group, a re-keying process is needed in order to protect the security of past, present, and future communications. The re-keying process should be done securely and efficiently so that it does not increase
the member joining and leaving latencies that are caused by the transmission of control messages and new keys. The re-keying process updates the SEK by encrypting the new SEK using a set of keys known as the “Key Encryption Key” (KEK) before distributing the encrypted SEK. A key management scheme has to be implemented to manage and securely distribute the KEK to valid group members. The key management needs to ensure that only valid group members have access to the messages encrypted with the new SEK. Newly-joined group members should not be able to access messages encrypted with previous SEKs, and members who have left the multicast group should not be able to access multicast messages after they have left.

2.4.1 Simple Key Distribution Scheme

A simple key management scheme assigns unique and independent KEK, $KEK_i$, to each member $M_i$. The SEK is then encrypted using the KEK and is distributed to each member. As a result, each member needs to store two keys, the KEK, $KEK_i$, which is unique to each member, and the SEK which is common to all members in the multicast session. When a new member joins the multicast session, the key management assigns a new KEK to that member. A new SEK is then generated and distributed to all group members after encryption with the KEKs. When a member leaves the multicast session, the key management generates a new SEK and distributes this to the remaining members after encryption with their KEKs. This is the so-called “1 affect N” effect, and the re-keying complexity is of $O(N)$ where $N$ is the number of members in the session. The
GC needs to store the SEK and the seed of a random number generator for each KEK. Therefore, the storage complexity of GC is 2.

2.4.2 Logical Key Hierarchical Scheme

To reduce the complexity of the re-keying process, a “Logical Key Hierarchical” (LKH) scheme to manage the keys is proposed in [28]. A similar method, in which key graphs are used, is proposed in [29]. In the hierarchical tree approach with the degree of the tree being \( d \), one SEK is used for all the group members. At the same time, each member remembers multiple keys. The member needs to store \( 2 + \log_d N \) keys, and these are the Root Key, SEK, and KEKs. All the keys are generated by the GC, and it organizes all the keys in a hierarchical structure. When there is a membership change, instead of sending \( N \) messages to change the SEK, only \( O(\log_d N) \) re-keying messages are needed. However, the GC has a key storage complexity of \( O(N) \) for this key management scheme.

2.4.3 Subgrouping Scheme

Another interesting idea to resolve the key management problem is to apply a divide-and-conquer approach to divide all the members into independent subgroups. The key management scheme Iolus proposed in [30] is a well-known algorithm of this type. Iolus organizes the multicast group into independent subgroups. Each subgroup can be further divided into lower level subgroups. There is no global SEK. Instead, each subgroup maintains its own SEK. When there are membership changes in the subgroup, the SEK of the associated subgroup is
changed, but those of other subgroups are not affected. The member needs to store two keys. This is smaller than those needed in a logical key hierarchy key management scheme. The key update communication overhead is $O(M)$ where $M$ is the size of the subgroup. The GC needs to store $O(N/M)$ keys for this key management scheme. However, the drawback is an increase in the data latency. This occurs because the subgroup GC needs to re-encrypt each message from the GC using the subgroup SEK before sending the encrypted messages to its members.
Chapter 3

The End-to-end Architecture

3.1 Design Principles

In the shared best-effort Internet, there are several principles in regard to the design of robust and secure streaming applications that must be followed.

3.1.1 Being Adaptive

With the best-effort service model in the Internet, there is neither an upper bound for delay nor a lower bound for the available bandwidth being allocated to individual services. Therefore, the quality of multimedia playback varies with time. The streaming applications should be able to adjust the quality of the delivered stream and, consequently, its consumption rate, in regard to long term changes in the available bandwidth, and operate in various network conditions. The multimedia bitrate $B$ or the encoded bitrate should be adjustable according to both the network condition and the client’s buffer condition in terms of the stored playtime $P(t)$. The purposes of multimedia bitrate adaptation are to provide
better quality playback and to maintain a steady amount of stored playtime $P(t)$ inside the client’s buffer so as to prevent a multimedia dropout problem, and to provide a smooth multimedia playback to the users.

### 3.1.2 Being Fast to Start

As discussed before, there is a startup delay in that the multimedia player must buffer up enough multimedia content before the start of playback to the users. The startup delay for a streaming system is prompted to be as small as possible because users do not want to wait a long time before being able to enjoy the multimedia sessions. There is a need for the flow control mechanism to be able to work in an end-to-end architecture which can achieve a higher buffer fill-up rate (smaller startup delay) for the client’s buffer, and to maintain the long term quality of the streaming media by adapting multimedia bitrate of the bitstream.

### 3.1.3 Being Friendly

As a dominant portion of today’s Internet traffic consists of a variety of TCP traffic, multimedia traffic should be TCP-friendly so as not to suppress the other TCP traffic. The TCP-friendly congestion control mechanism should be deployed in the end-to-end architecture for congestion control in multimedia streaming. And, in order to achieve TCP-friendliness, the AIMD congestion control scheme is the most suitable choice, as discussed, because the TCP congestion control mechanism is also in the AIMD form. Furthermore, the gateway-assisted congestion control mechanism should also be employed in order to detect and discard high
bandwidth traffic so that a fair share of the bandwidth can be achieved. The
gateway-assisted congestion control mechanism should also be TCP-friendly.

3.1.4 Being Resilient

Packets are randomly lost in the network because of congestion. Although stream-
ing applications can tolerate some loss, this also degrades the quality of the de-
livered streams. To maintain a reasonable quality, streaming applications need a
way to recover from most losses before their playout time. As discussed before,
the best way to achieve error control in multimedia streaming, is by using the
delay-constrained retransmission so that the retransmitted streaming packets can
still reach their delay bound and be used to play out.

3.1.5 Being Scalable

Multimedia streaming applications such as multimedia conferencing, interactive
group game and video-on-demand system, etc, usually operate in a one-to-many
communications model. However, current streaming protocols including flow
control, multimedia bitrate adaptation and congestion control are mainly im-
plemented in server-based systems. Therefore, if the number of clients continues
to increase, there will be a heavy load for both the server and the network. The
client-based streaming protocol is more scalable and is, therefore, more suitable
for multicast multimedia communications compared to the server-based stream-
ing protocol.
3.1.6 Being Useful

The quality of multimedia playback is degraded due to end-to-end delay variation, or delay jitter, when transported over the network. Packets received by the client are rendered useless if they have accumulated a large amount of delay jitter. Too much delay jitter also degrades the performance of the streaming buffer in the client. Therefore, a novel active buffer management scheme is needed in the gateways. This will improve the quality of service in multimedia networking by detecting and discarding packets that accumulate a large enough delay jitter. The aim is to maintain a high bandwidth for “useful” packets within the multimedia stream’s delay jitter tolerance.

3.1.7 Being Secure

As discussed, multimedia streaming applications usually involve multicast, and there is a need to provide security for multimedia content being sent to a specific multicast group. To achieve a secure delivery, the server encrypts the messages with the SEK previously distributed to all the valid group members so that the members can decrypt the messages accordingly. When there are changes in group membership (because of members joining or leaving), the GC distributes a new set of SEK to those group members associated with the membership changes in order to protect the security of past, present and future communications. Therefore, there is a need to employ an efficient key management scheme in the end-to-end architecture that can minimize member joining and leaving latencies, update communication overhead, and key storage for both the GC and members in the
3.2 Design Parameters

Before we describe the proposed architecture for multimedia streaming, the design space for individual key components and our specific design choices, are explored.

3.2.1 Flow Control

Due to the fact that the Internet only provides best-effort service, the current Internet does not provide any flow control for real-time multimedia traffic. Therefore, data buffer of a reasonable size should be allocated in the multimedia streaming client system so as to avoid a deleterious effect on network performance. As discussed, a robust flow control mechanism, which runs concurrently with the multimedia-on-demand system to monitor and regulate the flow of multimedia data to the buffer, is needed. This flow regulation system would constantly maintain the buffer at steady level so that, in the event of network delay, the client system would continue to play the multimedia data stored in the buffer until new multimedia data begins to arrive.

3.2.2 Multimedia Quality Adaptation

The flow control mechanism is used for monitoring and regulating the flow of multimedia data to the buffer. Further quality improvement for the multimedia playback can be achieved by employing a multimedia bitrate adaptation scheme
inside the flow control mechanism. The current quality adaptation and congestion control mechanisms for multimedia streaming do not consider the availability of the client’s buffer nor the amount of stored playtime $P(t)$. When the client’s buffer is full, even if the channel is available, increasing the sending rate $R$ results in the received packets in the client being discarded.

We designed a client-based flow control system with multimedia bitrate adaptation for multimedia-on-demand that provides the real-time playback of multimedia data transferred over the Internet. In this system the client is responsible for all the flow control decisions. Because of the client-based nature, only a small packet that contains the command signal is needed to be sent from the client to the server. As a result, it can quickly respond to and adapt, the multimedia bitrate $B$ of the coder output. Furthermore, the client can also expect, and be assured of the best quality bitstream for his or her network environment. The multimedia bitrate adaptation protocol provides multimedia quality adaptation for congestion controlled multimedia data streaming. The proposed system with bitrate adaptation maintains a high buffer fill up rate and a large amount of stored playtime $P(t)$, even in bursty loss periods, compared to that in which multimedia bitrate adaptation is not used.

### 3.2.3 Loss Recovery

Packet loss is inevitable in the Internet and leads to degradation of the quality of playback. The condition is even worse when there is a bursty loss during network congestion. Although real-time multimedia has a loss effect, the current
best effort Internet does not provide any loss guarantee. Thus, it is desirable that a multimedia stream be robust to packet loss. Furthermore, nonuniform packet arrival is a problem that a real-time multimedia system needs to deal with. To overcome these network difficulties, data buffer of a reasonable size should be allocated in the client system for error recovery and the reordering of streaming packets. We designed a loss packet recovery mechanism able to provide error recovery for the system, and to minimize the effect of packet loss in bursty conditions. To deal with the out-of-sequence packet arrival problem, the proposed nonuniform packet arrival protocol helps provide a smooth multimedia playback for users.

3.2.4 End-to-end Congestion Control

Several server-based congestion control mechanisms for multimedia streaming are proposed in [7, 9, 10, 11, 13]. They perform AIMD rate control in the server, similar to that of TCP. According to [12], those congestion control mechanisms are rated-based congestion control mechanisms. Those mechanisms achieve TCP-friendliness by dynamically adapting the sending rate $R$ in cases of network congestion. The client is required to send feedback information to the server and this is used to decide the kind of rate control operations which should be performed. The drawback is that if the feedback packets are lost or delayed due to network congestion, the performance of the system is severely degraded.
In order to provide scalability for the streaming system, a client-based TCP-friendly congestion control is proposed. Like the client-based flow control mechanism, only a small sized command signal is needed to be sent from the client to the server. So, it can quickly respond to, and resolve network congestion, compared to those server-based mechanisms.

The client-based congestion control mechanism is added to the system which is located in a lower level than the multimedia bitrate adaptation flow control. When the client detects that the packet is lost due to congestion, it sends a command packet back to the server requesting a lowering of the sending rate $R$ of the transmission of the packets. When there is no network congestion, the client requests the server to increase the sending rate for streaming. The client-based congestion control is in the form of the AIMD algorithm, similar to those proposed in [7, 9, 10, 11, 13], but the network congestion condition is determined by monitoring the packets loss in the client. When there is network congestion, the proposed congestion control mechanism shows an effective reaction by reducing the sending rate $R$ of the data while maintaining a certain degree of TCP-friendliness.

In addition, a resource allocation algorithm which uses a simple priority weighting scheme for multiple streaming traffic, is implemented in the proposed client-based congestion control mechanism. The reason for using this is that users who pay more to the service providers, should get more network resources for better quality of playback. The proposed resource allocation algorithm allows multimedia streams with high priority more resources. Thus, they can pass through
the bottleneck link more quickly than those with a low priority. Moreover, the proposed priority weighting scheme can be applied to layer-encoded multimedia streams so as to obtain further quality improvement.

3.2.5 Gateway-assisted Congestion Control

Traditional active buffer management schemes like DropTail and RED were originally designed for TCP traffic in the gateways. The gateways discard packets in the tail of the queue, or randomly discard high bandwidth traffic, in order to perform congestion control and resolve network congestion problems. However, those schemes do not work well with multimedia traffic. Those schemes do not consider the timing relationship of the multimedia packets or the “usefulness” of the streaming packets, i.e., they do not detect and discard streaming packets with too much accumulated delay jitter and those which exceed the useful threshold.

The proposed gateway-assisted congestion control mechanism, JD, designed for application to each streaming packet, uses the delay jitter counter to select the discard threshold. In our study, we further investigated the notion that a measure of how far, or close, the packet is from its destination is taken into consideration for the discard decision. The aims are to maintain a high bandwidth for “useful” packets within the multimedia stream’s delay jitter tolerance and a streaming packet that has accumulated a large delay jitter value should not be dropped if it is close to the client.
3.2.6 Key Distribution for Secure Multimedia Multicast

We present algorithms that generate the proposed MRT for a key distribution with a given probability of leaving group for each member. The MRT, optimal in minimizing the re-keying costs, keeps the minimum average number of keys $\bar{l}$ needed to be updated for each member, and the minimum average tree height $\bar{h}$ for each member. Members who leave more frequently (have a higher probability of leaving group $P_l$) have a smaller number of keys $l_i$ assigned than that of members who stay longer in the group. Also, the members who stay the longest in the group have the same, or the smallest variance, in the number of keys assigned. Furthermore, the tree update procedure maintains the optimality of the key tree after re-keying. An analytical analysis of the proposed MRT algorithm was performed, and the key tree update interval under certain cost constraint was computed. Finally, after combining the MRT with subgrouping, a multiple MRTs scheme, which further minimizes member storage, GC storage and average update communication overhead compared to that of other key management schemes, was proposed.

3.3 The Proposed End-to-end Architecture with QoS Control

Figure 3.1 depicts the proposed end-to-end architecture with QoS control for multimedia streaming over the Internet. The proposed multimedia bitrate adaptation and loss packet recovery modules are responsible for providing the best
Figure 3.1: The proposed End-to-end Architecture with QoS Control for Multimedia Streaming over the Internet

quality of multimedia content and the resilient ability of the streaming system, respectively. Both modules monitor the streaming buffer and make the corresponding QoS control decisions. The proposed client-based congestion control mechanism regulates the sending rate $R$ of the multimedia traffic based on the loss statistics from the loss packet recovery module. Moreover, the proposed gateway-assisted congestion control, JD, is involved in detecting and discarding those streaming packets with a large accumulated delay jitter. Furthermore, the end-to-end unicast architecture can be extended into the multicast architecture.

The proposed minimum redundancy key management scheme, MRT, for secure multimedia multicast should be applied to provide security and privacy of the multimedia content for those valid group members. In fact, this end-to-end
architecture can be applied to any streaming system. Any modifications to the QoS control mechanisms can be directly plugged into this architecture without affecting the other mechanisms.
Chapter 4

Client-based Streaming Protocol

4.1 Basic Flow Control

The aim of the flow control mechanism is to monitor and regulate the flow of the streaming packets into the client’s buffer.

4.1.1 Server-client Communications

Without loss of generality, we considered the most simple multimedia streaming application, that the multimedia server waits for a request from the client before initiating a connection using the TCP/IP flow regulation. The server responds by transmitting a block of initial information using TCP/IP. This informs the client about the availability of the requested multimedia stream and the available bit rate, frame-size and other information required to decompress and start the playback of the multimedia bitstream. The server starts data streaming using the UDP/IP protocol right after sending the initial information packet. Multimedia streaming involves transferring the streaming packets from the server to the client
within a certain end-to-end delay bound. So UDP is mainly used instead of TCP, despite of some streaming applications still using TCP. UDP is considered to be the ideal transport layer protocol for streaming applications because it is a “connectionless” protocol. Unlike TCP, when a UDP packet is dropped, the server keeps sending information, causing only a brief glitch instead of a huge gap of silence in the real-time playback of the multimedia stream.

Figure 4.1 shows the details of the flow control mechanism between the server and client. The server maintains a data sequence number $DSeq$ counter, and a $DSeq$ is assigned to each data packet. The client maintains a play sequence number $PSeq$ counter, so that out-of-sequence data packet arrival is allowed only if the $PSeq$ of the packet that has been decompressed and played is smaller than the received $DSeq$. If it is not, the out-of-sequence arrival data packets are discarded.

Moreover, a $Plist$ is maintained. This is used to store and record the received streaming data packets. These information is useful for updating the stored playtime $P(t)$ to a reasonable level so as to avoid multimedia dropout and low quality multimedia playback. The $Plist$ is also used to detect lost packets and the out-of-sequence arrival of packets. This will be discussed later.

Figure 4.1 details the flow control using these two counters ($DSeq$ and $PSeq$) plus two time dependent variables ($P(t)$ and $C(t)$) over UDP/IP. The stored bitstream is played when a stored playtime $P(t)$ in the client’s buffer is greater than a given startup delay time $S_d$. The playback and the communication session cease when the client receives an $EndOfMedia$ packet.
4.1.2 Packetization

Shown in Figure 4.2 is a data packet format of the streamed media. To simplify the discussion, the data packet format is based on the RTP/UDP/IP. We will discuss those fields in the header that are needed to provide the proposed multimedia bitrate flow control mechanism. Those fields in the packet header are designed to provide design flexibility to the streaming applications. Different streaming protocols may contain different fields in their packet formats. However, there are some fields that all the streaming protocols should have.
Time Stamping

In a streaming packet format, there should be a field called Timestamp. Timestamp is used to synchronize different media streams. Synchronization between media streams should be done by the streaming applications. Timestamp is also used to calculate the RTT and delay jitter for network condition predictions.

Sequence Numbering

Since packets arriving at the client may be out of sequence (UDP does not deliver packets in sequence), the streaming protocol should employ sequence numbering to place the incoming streaming packets in the correct order. The sequence number is also used for packet loss detection.

Multimedia Bitrate Identification

Some streaming protocols may contain a field that indicates the current multimedia bitrate of the streaming encoder. Each packet is identified by a particular multimedia bitrate suitable for the clients to decode. This is used for the deployment of our bitrate adaptation flow control design, and for congestion control purposes embedded in a streaming protocol.

Frame Size / Packet Size

Frame size shows the size of the payload carried by the multimedia data packet. The frame size or packet size is used to calculate the additional received playtime contained in this received multimedia data packet.
4.1.3 Buffer Control

When the multimedia bitrate $B$ of the compressed multimedia stream is lower than the sending rate of the network, the client sends a control sequence to request the server to suspend data streaming when the buffer is full, i.e. $P(t) > B_{\text{max}}$. Streaming is resumed when the buffer content falls below a pre-assigned threshold $P_{\text{th}}$, i.e. $P(t) < P_{\text{th}}$. The details of the buffer control are shown in Figure 4.3.

Due to the high packet loss rate of the Internet, the effective sending rate of the network is substantially lower. This, in turn, constrained by the multimedia bitrate $B$ of the streaming multimedia, is much lower than the channel capacity. As a result, the channel bandwidth is wasted in much of the time. The situation is worse when the server ceases data streaming due to the client’s buffer overflow. Multimedia bitrate adaptation is therefore proposed.
\begin{figure}
\begin{center}
\begin{tabular}{|l|l|}
\hline
\textbf{Client} & \textbf{Server} \\
\hline
If (P(t) > MAX) \{ & If (CSeq > SSeq) \{ \\
If \( T_0 = 0 \) & SSeq = CSeq \\
Start Timer \( T_0 \) & If (request higher multimedia \\
Else if (\( T_0 > \text{MBA}_a \)) \{ & bitrate stream) \{ \\
CSeq ++ & B = B + \Delta \\
Request higher multimedia & \} \\
bitrate stream & \} \\
\( T_0 = 0 \) & \} \\
\} & Else if (request lower multimedia \\
\} & bitrate stream) \{ \\
\hline
Else if (P(t) < MIN) \{ & \quad B = B - \Delta \\
If (T_0 = 0 \text{ or } T_0 > \text{MBA}_a) \{ & \} \\
CSeq ++ & \} \\
Request lower multimedia & \} \\
bitrate stream & \} \\
\hline
T_0 = 0, \text{Start Timer } T_0 & \} \\
\} & \} \\
\hline
\end{tabular}
\end{center}
\caption{Multimedia Bitrate Adaptation Mechanism}
\end{figure}
4.2 Multimedia Bitrate Adaptation

In general, data buffer is allocated for each multimedia system. It is used to store sufficient playtime $P(t)$ for continuous multimedia presentation. In our system, we define

$$P(t) = P'(t) + \text{ReceivedPackets} \times \text{FrameSize}/B - C(t), \quad (4.1)$$

where $P'(t)$ is the current stored playtime in the client’s buffer, ReceivedPackets equals the number of packets received by the client in one TIMEOUT, FrameSize is the multimedia payload size, $C(t)$ is the consumption rate of the buffer and $B$ is the multimedia bitrate of the encoded stream. The proposed multimedia bitrate adaptation algorithm, detailed in Figure 4.4, assumes the availability of adaptive encoding technology [1, 2, 8]. The adaptive encoding adjusts the resolution of the encoding scheme by changing the quantization factors of the encoder. Thus, it generates multimedia bitstreams with different multimedia bitrate values $B$. The availability of the buffer is considered in the adaptation scheme. This is missing in existing streaming rate control protocols such as [7, 9, 10, 11, 13].

4.2.1 Decision of Adaptation

Figure 4.4 details the proposed multimedia bitrate adaptation. Under normal operation, the data packets are transmitted at sending rate $R$ faster than real-time. Thus, the stored playtime $P(t)$ reaches, and then overtakes, the maximum threshold $MAX$. If the $P(t)$ remains higher than $MAX$ for a predetermined
period of time $MBA_{th}$, the client system requests the server to encode the multimedia data with higher quality using multimedia bitrate of $B = B + \Delta$. Here, $\Delta$ is the multimedia bitrate increment which depends on the encoding scheme used. If layered encoding scheme is used, $\Delta$ is the multimedia bitrate difference between different layers. If adaptive encoding scheme is used, $\Delta$ is the specified value which is determined by the encoder in the server. A longer time is needed for the transmission of a high quality data since more data needs to be transmitted at the same sending rate $R$. Furthermore, high quality data has a high consumption rate. As a result, $P(t)$ drops at a faster rate.

Once $P(t)$ falls beneath a minimum threshold $MIN$ for $MBA_{th}$ second, a low quality data packet request is issued. This requests the encoder to encode the multimedia data of a multimedia bitrate of $B = B - \Delta$. This process is repeated so that high quality data is mixed with normal quality data. The result is an overall playback quality improvement.

Typically, more than one block is transferred over the Internet in a given time period. Once a multimedia bitrate adaptation request is issued, the data packets still in the network are received, and are stored in the buffer before the bitrate adapted packets are sent out. The buffer overflow, or underflow, condition remains for a certain period of time, until previously received packets are consumed. Having a multimedia bitrate adaptation time threshold $MBA_{th}$ helps to minimize unnecessary multimedia bitrate changes requests. Furthermore, a control sequence number $CSeq$ is assigned to each control packet, and a server control sequence number counter $SSeq$ is maintained in the server in order to
store the largest $CSeq$ number received so far. The server ignores any received
control packets with $CSeq < SSeq$.

4.2.2 Timeout Mechanism

The multimedia bitrate adaptation flow control shown in Figure 4.4 is only ef-
tective when the stored playtime $P(t)$ is greater than, or smaller than, the two
threshold values $MAX$ and $MIN$, respectively. A preventive multimedia bitrate
adaptation, with two timers, $T_c$ and $T_s$ located in the client and the server, re-
spectively, is shown in Figure 4.5. These measure the time between the reception
of streaming data packets in the client and $ACK$ in the server is proposed. The
multimedia bitrate $B$ is lowered when $T_c$ or $T_s$ is greater than the $TIMEOUT$
constant. Because streaming low multimedia bitrate $B$ packets help the system
to recover from prolonged network congestion, the $T_c$ and $T_s$ reset to 0 whenever
the client receives a valid data packet, or when the server receives an $ACK$,
respectively. $ACK$ is sent by the client after receiving $\delta$ packets. This minimizes
server-client communications.

4.2.3 Adaptive Buffer Design

To obtain the best quality service, the buffer size is made adaptive. The $MAX$
and $MIN$ parameters which govern the multimedia bitrate adaptation threshold
are adjusted after each multimedia bitrate adaptation so as to minimize the num-
bers of unnecessary adaptation, which results in too much streaming data quality
fluctuation. The buffer adaptation maintains the stored playtime difference between two threshold at a constant level, i.e. $MAX - MIN = constant$, so as not to increase the dropout rate after adaptation. Furthermore, $MAX$ is increased to $MAX + b(t)$, where $b(t)$ is the playtime difference between data packets before and after a higher multimedia bitrate $B$ request in multimedia bitrate adaptation. Similarly, $MIN$ is decreased to $MIN - b(t)$ when a low multimedia bitrate $B$ request is issued.

4.2.4 Markov Analysis of Multimedia Bitrate Adaptation

At a particular time $t$, the multimedia bitrate adaptation can be modeled as a 3-state Markov Chain as shown in Figure 4.6. If the initial state probabilities $P_B = P_{B-\Delta} = P_{B+\Delta} = p$, then the state probabilities, $P[B]$, $P[B-\Delta]$ and $P[B+\Delta]$ are given by the following equations,
Figure 4.6: 3-state Markov Chain of Multimedia Bitrate Adaptation at a particular time \( t \)

\[
P[B + \Delta] = P_B\alpha + P_{B+\Delta}(1 - \beta) = p + p(\alpha - \beta),
\]

\[
P[B - \Delta] = P_{B-\Delta}(1 - \alpha) + P_B\beta = p - p(\alpha - \beta),
\]

\[
P[B] = P_B(1 - \alpha - \beta) + P_{B-\Delta}\alpha + P_{B+\Delta}\beta = p, \tag{4.2}
\]

where \( \alpha \) and \( \beta \) are the transition probabilities, which is determined by the streaming applications. If \( \alpha > \beta \), there is a higher probability in increasing the multimedia bitrate \( B \), which is more realistic in current Internet situations and more suitable for existing streaming applications. Therefore, the corresponding state probabilities are given by,

\[
P[B + \Delta] = p + p(\alpha - \beta),
\]

\[
P[B - \Delta] = p - p(\alpha - \beta),
\]

\[
P[B] = p, \tag{4.3}
\]

As a result, we can calculate the multimedia bitrate \( B \) at \( k^{th} \) state,

\[
B_k = (B_{k-1} + \Delta)P_{k-1}[B + \Delta] + (B_{k-1} - \Delta)P_{k-1}[B - \Delta] + B_{k-1}P_{k-1}[B]. \tag{4.4}
\]
The relationship between the multimedia bitrate \( B \) and the two threshold values, \( MAX \) and \( MIN \) were analyzed for the case \( \alpha > \beta \) when the available channel bandwidth is 300Kbps, the TIMEOUT period is 20s, the multimedia bitrate increment \( \Delta \) is 6.6Kbps, the initial \( B \) is 132Kbps, and the initial state probability \( p = 1/3 \). The simulations also revealed the effects on the change of \( B \) under different \( P[Loss] \) values.

The results in Figure 4.7 show the relationship between \( B \) and \( MAX \). It is shown that if the \( MAX \) decreases, the \( B \) will be increased because there is a higher probability of increasing the \( B \). Also, if the \( P[Loss] \) decreases, that is the case when the network is not congested, the \( B \) will then be increased as shown in the Figure 4.7. It implies the multimedia bitrate adaptation can provide multimedia quality improvement when there is sufficient available network bandwidth, as the \( B \) is directly related to the quality of the multimedia streams.

On the other hand, the results in Figure 4.8 show that if the \( MIN \) increases, the \( B \) will be decreased because there is a higher probability in decreasing the \( B \). Also, if the \( P[Loss] \) increases, that is the case when the network is congested or during the bursty loss period, the \( B \) will then be decreased as shown in Figure 4.8. It implies that the multimedia bitrate adaptation can provide a higher buffer fill-up rate for the streaming system by lowering the \( B \). It can also prevent the multimedia dropout during the bursty loss period. The results also show that by varying the two threshold values \( MAX \) and \( MIN \), it can actually control the change of the multimedia bitrate \( B \). It gives an good analytical reason for the design of adaptive \( MAX \) and \( MIN \) as discussed in Section 4.2.3.
Figure 4.7: Multimedia Bitrate $B$ at each time instant for different MAX, $\alpha > \beta$

Figure 4.8: Multimedia Bitrate $B$ at each time instant for different MIN, $\alpha > \beta$

4.2.5 Simulation using NS2

The NS2 [21] was used to simulate the proposed algorithms with simulation topology as shown in Figure 4.9. Only one multimedia source is sent with an initial rate of 1.5Mbps (Streaming traffic with the proposed multimedia bitrate adaptation), and three TCP sources, all with a rate of 1.5Mbps (FTP traffic) were sent. They were all connected to the bottleneck router R1 by a link with 5Mbps max capacity and a propagation delay of 3ms. The bottleneck link had a maximum capacity of 1.5Mbps, and a propagation delay of 10ms. The RED was used in the bottleneck routers (R1 and R2) for gateway-assisted congestion control of the TCP and UDP packets. The effectiveness of the multimedia bitrate adaptation in response to bursty traffic was observed by imposing a bursty UDP traffic with a constant bitrate UDP traffic (CBR) at 1Mbps during the simulation time periods 3s-4s and 6s-7s. In the NS2, the CBR traffic is constant in its sending rate $R$, but its multimedia bitrate $B$ can be varied. Throughout the simulation, all traffic had the same packet size of 1000 bytes and the time interval length $TIL$ for throughput calculation was set at 0.1s. Several simulations were performed
to investigate the stored playtime $P(t)$ in the client’s buffer and TCP-friendliness of the multimedia traffic. We also performed the same set of simulations using a constant bitrate UDP traffic (CBR) at 1.5Mbps to replace the multimedia traffic. For simplicity, there were only five levels of multimedia bitrate $B$ for the operation of the multimedia bitrate adaptation in these simulations. We compared the results obtained using multimedia bitrate adaptation to those obtained without the use of multimedia bitrate adaptation. The results clearly demonstrated the performance of the proposed system.

### 4.2.6 Results and Discussion

**Stored Playtime and Buffer Fill-up Rate**

The stored playtime $P(t)$ in the client buffer with multimedia bitrate adaptation over a bursty network (bursty conditions during the periods of 3s-4s and 6s-7s) is shown in Figure 4.10. The two red lines in Figure 4.10 depict the MAX and MIN parameters. The playtime stored in the client buffer under a normal channel without bursty conditions is presented in Figure 4.10 for the startup phase of the streaming session. The result shows that a fast buffer fill-up rate in bursty periods is achieved when the proposed bitrate adaptation protocol is used. The cooperation between the multimedia bitrate and the buffer adaptations effectively enhance the buffer fill-up rate during a high loss period. The client’s buffer maintains a large amount of stored playtime $P(t)$ even in the bursty channel. The multimedia bitrate adaptation lowers the multimedia bitrate $B$ of the bitstream and, hence, increases the stored playtime $P(t)$ in the buffer to a level at which
dropout is avoided. Recall that when the multimedia bitrate $B$ from the encoder is small, the result is a large amount of stored playtime $P(t)$. Furthermore, maximum buffering can be achieved in a short time. It is clear that the quality of the multimedia content is lowered due to a lower multimedia bitrate $B$. However, for a real-time multimedia-on-demand system, it is important to maintain a continuous playback which has an acceptable quality. A relatively higher quality bitstream, but with high dropout rate, will be more annoying to users. The fast buffer fill-up rate ensured through the use of the proposed protocol results in smaller startup delay with acceptable quality multimedia playback compared to that obtained without multimedia bitrate adaptation flow control mechanisms. A smaller startup delay means less hold time, and a faster presentation of the
streamed media to the user. Also, a fast buffer fill-up rate reduces the multimedia dropout rate as the buffer can maintain a sufficient amount of playtime to be presented.

**Multimedia Bitrate of the Streaming Media**

The multimedia bitrate $B$ of the streaming media using the proposed multimedia bitrate adaptation is shown in Figure 4.11. The initial multimedia bitrate $B$ of the streaming media is set to 1.5Mbps. The multimedia bitrate is specified in bits/TIL and the time interval length $TIL$ used in this simulation was 0.1s. For example, if the multimedia bitrate $B$ is measured as 150Kbits/TIL, then its multimedia bitrate $B$ will be 1.5Mbps. The same simulation is done in Figure 4.11, the streaming traffic was replaced by a constant bitrate UDP traffic (CBR) with no multimedia bitrate adaptation mechanism. The same flow control mechanism and parameter settings were used in this CBR traffic, but without bitrate adaptation. Therefore, the multimedia bitrate $B$ of the CBR traffic could not be varied according to the network and client’s buffer situations. From Figure 4.11, it can be observed that the multimedia bitrate $B$ of the multimedia traffic remains constant before 3s. However, when there is CBR bursty traffic, the multimedia bitrate $B$ decreases. This is because the client requests the server to reduce its multimedia bitrate $B$ of the streaming encoder due to network congestion, even without applying any congestion control mechanism. The results show that the proposed multimedia bitrate adaptation can be applied to a real network situation such that the streaming coder varies the multimedia bitrate $B$
according to the both current network and client’s buffer situations. Moreover, the multimedia traffic does not suppress the other TCP traffic as happens in systems with no multimedia bitrate adaptation, as shown in Figure 4.11. It can be observed that the multimedia bitrate $B$ is directly proportional to the sending rate $R$. Moreover, they are actually located in two different levels in the layered structure in the streaming architecture. This is evidence that the proposed multimedia bitrate adaptation algorithm works well with the proposed congestion control algorithm. The proposed client-based congestion control mechanism is discussed in Chapter 5.

4.3 Loss Packet Recovery

As the current best-effort Internet does not provide any loss guarantee, real-time multimedia streaming has a loss problem. Thus, it is desirable that a multimedia stream be robust to packet loss. Furthermore, nonuniform packet arrival is a problem that a real-time multimedia system needs to deal with.

4.3.1 Loss Packet Recovery Mechanism

To overcome these network difficulties, a data buffer of a reasonable size should be allocated in the client system so to avoid deleterious effects on network performance. Therefore, an error control protocol should be incorporated to provide error recovery for the system. The streaming system protocols should be designed to minimize the effect of packet lost in bursty channels, the out-of-sequence packet
Figure 4.11: Multimedia Bitrate $B$ of the Sources, $TIL=0.1s$, (Upper) With Multimedia Bitrate Adaptation, (Bottom) Without Multimedia Bitrate Adaptation
arrival problem, and provide a smooth media playback for the users. A robust loss packet recovery mechanism is proposed, and this is detailed in Figure 4.12. A timer $T_d$ is maintained in the client system so that the lost packet resent request is made periodically whenever $T_d > T_r$ (the predefined resent constant). Due to network delay, it is possible that the resent packets might arrive later than the current playtime $CP(t)$. This wastes channel bandwidth. To avoid network congestion, only those lost packets with a playtime greater than $1.5T_x$ should be requested for retransmission. Changing the playtime overhead to a higher value conserves the bandwidth, but increases the packet lost rate and, hence, the dropout in playback. A list, $Plist$, is used to store the received packet numbers
and a counter, $RSeq$, is used to store the largest received packet number. Another list, $Rlist$, stores the packet numbers in the resent request. To prevent bandwidth wastage due to multiple resent requests in a congested network, the lost packet number in the previous resent request is not considered in the current resent request but in the subsequent request.

### 4.3.2 Nonuniform Packet Arrival Mechanism

The buffered packets are played according to their sequence number to overcome out-of-sequence arrival. Such a scheme introduces a glitch in playback in lossy networks. A counter, Dropout, in the client measures the dropout rate in the unit of packets. If a packet arrives in order i.e. $DSeq < RSeq$, then Dropout is decreased, as shown in Figure 4.13. If it does not arrive in order, if $DSeq > RSeq+1$, then nonuniform packet arrival occurs and Dropout is increased. When Dropout reaches $u$, a predefined constant, this indicates that the system has a serious multimedia dropout problem. As a result, multimedia bitrate adaptation should be requested in order to quickly restore the stored playtime $P(t)$ into the buffer.

### 4.3.3 Simulation

Simulations were performed to demonstrate the effectiveness of the proposed loss packet recovery mechanism. The simulation results, in regard to the relationship between the packet receive rate and the time for different channel bandwidth, are shown in Figure 4.14. The packet receive rate is an alternative measure of the
If (DSeq < SSeq)
{  Dropout--
  If (Dropout < 0)
  {  Dropout = 0  }
}

Else if (DSeq > RSeq + 1)
{  Dropout = Dropout
    + DSeq - RSeq - 1
  If (Dropout > u)
  {  CSeq++
      Request lower multimedia
      bitrate stream
      Dropout = 0
  }
}

If (CSeq > SSeq)
{  SSeq = CSeq
  if (request lower multimedia
      bitrate stream)
  {  B = B - Δ
  }
}

Figure 4.13: Nonuniform Packet Arrival Mechanism

Figure 4.14: Packet Receive Rate for Channel Bandwidth: 56Kbps, 112Kbps, 224Kbps and 448Kbps
successful packet transmission rate of the system. The packet receive rate in four different channel bandwidth, 56Kbps, 112Kbps, 224Kbps and 448Kbps, is shown in Figure 4.14. It can be observed that the packet receive rate is exponentially related to time, and increases with the channel bandwidth. We conjecture that a successful packet transmission rate will follow the same trend. Furthermore, the simulation results also show the effect on the packet receive rate with, and without, the proposed loss packet recovery mechanism. The result in Figure 4.15 clearly indicates that the packet receive rate of the system with the loss packet recovery mechanism (in blue lines) is higher than that without the loss packet recovery mechanism (in red lines). The channel bandwidth is set at 56Kbps and the loss probability ranges from 0.1 to 0.9. Figure 4.16 shows the packet receive rate in a 132Kbps channel with different loss probability. It can be observed
that the packet receive rate with the loss packet recovery mechanism is always higher than that without the loss packet recovery mechanism in different noisy channel conditions. The results in Figure 4.15 and 4.16 show the effectiveness of the proposed loss packet recovery mechanism in improving the successful packet transmission rate in noisy channels.

Figure 4.17 shows the relationship between the loss percentage, and the time and the channel bandwidth at 56Kbps. The loss packet recovery mechanism is shown to achieve a lower loss percentage than that of system that does not have the loss packet recovery mechanism. Moreover, a smaller fluctuation in the range of the loss percentage is observed in the system with a loss packet recovery mechanism. To maintain a smooth multimedia playback, a steady loss percentage of the system is desired. Under a bursty period, the consecutive packet loss always
Figure 4.17: Packet Loss Rate at 56Kbps with Loss Percentage of 25%

Figure 4.18: Packet Loss Rate at 56Kbps with Loss Percentage of 25% and Bursty Period at 50s-70s
leads to a high loss percentage in the streaming system, as shown in Figure 4.18, in which the bursty period occurs from 50s to 70s. The simulation result shows that the loss packet recovery mechanism helps the system to withstand a sudden change of loss percentage during the bursty period. If there is no loss packet recovery, the maximum loss percentage has a large fluctuation than that with a loss packet recovery during the bursty period.
Chapter 5

Client-based Congestion Control

5.1 Client-based Congestion Control

The proposed client-based congestion control mechanism helps to reduce packet loss and delay, enhance resource sharing between traffic, and provide a certain degree of TCP-friendliness. As discussed, client-based means that the decision about the congestion control mechanism is made by the client, and that the action of congestion control mechanism is also initiated by the client. Figure 5.1 details the proposed client-based congestion control mechanism. The proposed client-based congestion control mechanism considers the changes in the sending rate $R$ of the server according to the available network bandwidth. Here, the sending rate $R$ is the rate for the streaming packets to be transmitted over the network. This is different from the multimedia bitrate $B$, as discussed in multimedia bitrate adaptation in Chapter 4. The multimedia bitrate $B$ is the encoded bitrate of the multimedia content from the encoder. Therefore, the multimedia bitrate $B$ is located on the application layer in the OSI model while the sending rate $R$ is
For every RTT period
If (loss >= 1 || Timeout) {
    CSeq ++
    Request lower sending rate R
    set loss = 0
}
Else if (loss == 0) {
    CSeq ++
    Request higher sending rate R
}

If (CSeq > SSeq) {
    SSeq = CSeq
    If (request lower sending rate) {
        R = R * (1 - b) * w
    } Else if (request higher sending rate) {
        R = R' + s * a
    }
}

where b: decreasing factor ,
    w: weight factor ,
a: raising step

Figure 5.1: Client-based Congestion Control Mechanism

on the transport layer. To make the multimedia traffic TCP-friendly, the client-based congestion control applies the ideas of the TCP AIMD congestion control mechanism. The server starts with a sending rate of R. The sending rate R changes according to the available channel bandwidth.

5.1.1 Decision of Rate Adaptation

The client is responsible for monitoring network congestion, and for determining the adaptation of the sending rate R. Congestion monitoring is achieved via a simple periodic (for every RTT second) packet loss monitoring routine, in which a single packet loss in each period is regarded as network congestion. When congestion is detected, the client requests the server to reduce the sending rate
by

\[ R = R \times (1 - b) \times w, \]  \tag{5.1}

where \( b \) is the decreasing factor and \( w \) is the weight factor. The weight factor \( w \) is used to provide resource allocation. This is discussed in the following section. If no packet loss is detected, the client requests the server to increase the sending rate \( R \) by

\[ R = R + \delta \times a, \]  \tag{5.2}

where \( a \) is the raising step size. It is clear that the adaptation of the sending rate \( R \) is in the form of AIMD. This is similar to that of TCP. The objective is to make the rate adaptation mechanism TCP-friendly, just like those in [7, 9, 13].

### 5.1.2 Timeout Mechanism

In addition to the packet loss, the proposed client-based congestion control mechanism is also initiated by the \textit{TIMEOUT}. This is the same as in multimedia bitrate adaptation. If the server does not receive any command from the client for a \textit{TIMEOUT} period due to congestion in the feedback path, it automatically reduces the sending rate \( R \). This is similar to that of multimedia bitrate adaptation by \textit{TIMEOUT}, as indicated in Figure 4.5.

### 5.1.3 Combination with Multimedia Bitrate Adaptation

The sending rate \( R \) is the transmission rate of the multimedia information and is governed by the proposed congestion control. But the multimedia bitrate \( B \),
as discussed before, is the multimedia bitrate of the multimedia content from
the encoder. In the traditional Internet model, the multimedia bitrate $B$ is on
the application layer, the resultant multimedia bitrate after multimedia bitrate
adaptation while the sending rate $R$ is on the transport layer, and the transmis-
sion rate of the multimedia information is controlled by the congestion control
mechanism. Therefore, the proposed multimedia bitrate adaptation mechanism
actually runs on the top of the proposed client-based congestion control mecha-
nism. The multimedia encoder encodes the multimedia content with multimedia
bitrate $B$ and the encoded multimedia information is sent out at sending rate
$R$. In the proposed streaming architecture, both $B$ and $R$ can be varied using
the proposed multimedia bitrate adaptation and client-based congestion control
mechanisms, respectively.

5.1.4 Resource Allocation by Priority Weighting

In the proposed client-based congestion control mechanism, as shown in Figure
5.1, there is a weight factor, $w$, for providing resource allocation among different
multimedia traffic. Today, users who pay more should be able to get more network
resources for better quality multimedia playback. The weight factor $w$ is linearly
related to the price that the user paid, and the priority levels of the multimedia
traffic like video traffic are more important than those of audio traffic,

$$w = f(Price, Priority).$$  \hspace{1cm} (5.3)
5.1.5 Scalability

Multimedia streaming applications usually involve one-to-many communications. However, existing congestion control mechanisms [7, 9, 13] are mainly server-based systems. Therefore, if the number of clients continuously increases, there is a heavy burden on both the server and the network. The client-based congestion control is more scalable, and is, thus, more suitable for multicast multimedia communications compared to the server-based congestion control mechanisms [7, 9, 13].

5.2 Simulation

The NS2 [21] was used to simulate the proposed client-based congestion control mechanism. The effectiveness of the proposed client-based congestion control mechanism, cooperating with the proposed multimedia bitrate adaptation as discussed in Chapter 4, was investigated in one of the simulations. A comparison of the proposed client-based congestion control mechanism, and the well-known TFRC [13] mechanism is also given in the following sections. Another simulation was performed to look into the efficiency and the effectiveness of the proposed algorithm in dealing with multiple multimedia streams.
5.3 Results and Discussion

5.3.1 Combination of Multimedia Bitrate Adaptation and
Client-based Congestion Control mechanisms

We simulated the proposed congestion control mechanism with the proposed multimedia bitrate adaptation scheme presented in Chapter 4. The simulation topology used, as shown in Figure 5.2, was the same as that used to obtain the information presented in Chapter 4. Again, there were one multimedia source sending at an initial rate of 1.5Mbps (Streaming traffic with the proposed multimedia bitrate adaptation and congestion control) and three TCP sources with a rate of 1.5Mbps (FTP traffic). They were all connected to the bottleneck router R1 by a link with 5Mbps maximum capacity and a propagation delay of 3ms. The bottleneck link had a maximum capacity of 1.5Mbps and a propagation delay of 10ms. The RED was used in the bottleneck routers (R1 and R2) for the gateway-assisted congestion control of the TCP and UDP packets.
Figure 5.3: Throughput of the Traffic in the Bottleneck Link, $TIL = 0.1s$, (Upper) With Multimedia Bitrate Adaptation and Client-based Congestion Control, (Bottom) Without Multimedia Bitrate Adaptation and Client-based Congestion Control
The throughput of all the traffic in the bottleneck link was simulated. The throughput of the streaming media, when the proposed mechanisms is used, is shown in Figure 5.3. The throughput was also measured in bits/TIL and the time interval length $TIL$ in this simulation was 0.1s.

The results, similar to those presented in Chapter 4 show that the throughput of the multimedia traffic in the bottleneck link goes up and down due to the effect of both multimedia bitrate adaptation and congestion control mechanisms. Again, the multimedia traffic does not suppress the other TCP traffic. This happens in systems without multimedia bitrate adaptation and congestion control, as shown in Figure 5.3. Also, the results show that better TCP-friendliness can be achieved using both multimedia bitrate adaptation and congestion control mechanisms.

As discussed before, the multimedia bitrate $B$ is directly proportional to the sending rate $R$. If $B$ is varied, $R$ also changes. However, the resultant sending rate $R$ to the channel, controlled by the congestion control mechanism, is based on the available network bandwidth. The important result, shown in Figure 5.3, is that the streaming traffic does not suppress the bursty CBR traffic. This shows the fairness of sharing the bandwidth with other UDP traffic. The proposed system does not forcibly suppress other TCP traffic, unlike the system without both multimedia bitrate adaptation and congestion control mechanisms, as shown in Figure 5.3.
5.3.2 Comparison with TCP-friendly Rate Control (TFRC)

Another simulation using the topology shown in Figure 5.4 was done to compare the performance of the proposed client-based congestion control with that of TFRC traffic by [13]. We used two multimedia sources (one with the proposed Congestion Control mechanism and one with the TFRC mechanism) and two TCP sources with rates of 1.5Mbps (FTP traffic). The bottleneck link had a maximum capacity of 3.5Mbps and a propagation delay of 10ms. All traffic started their transmissions at 0s. In Figure 5.5, a comparison of the throughput of the proposed client-based congestion control mechanism, and that of the TFRC mechanism in the bottleneck link is presented. The multimedia stream with the proposed client-based congestion control mechanism varies its sending rate $R$ when there is congestion. The TFRC mechanism performs in the same way, and fairly shares the link capacity with other TCP traffic. In Figure 5.5, it is
Figure 5.5: Throughput of the Traffic in the Bottleneck Link, $w=1$, $a=1$, $b=0.5$, (Upper) $TIL=0.1s$, (Bottom) $TIL=1s$

90
shown that the proposed client-based congestion control can ensure a performance similar to that of TFRC in regard to sharing the channel capacity with other TCP traffic. In this simulation, the decreasing factor $b$ was set at 0.5, and the client-based congestion control mechanism reduced its sending rate by half once it detected a loss.

5.3.3 Client-based Congestion Control with Multiple Streaming Traffic

A different simulation, one in which the topology shown in Figure 5.6 was used, was performed to investigate the effect of congestion control in regard to managing multiple streaming traffic. We used three multimedia sources (Streaming traffic with the proposed Bitrate Adaptation and Congestion Control mechanisms) and two TCP sources with rates of 1.5Mbps (FTP traffic). The same network topology as that in the first set of simulations was used, but the bottleneck link had a maximum capacity of 3.5Mbps, and a propagation delay of 10ms. Resource
allocation can be achieved by applying different priority weights, \( w \), into different streams. The traffic that has a higher priority obtains more resources for passing through the bottleneck link.

Figure 5.7 shows the effectiveness of the congestion control mechanism in the bottleneck link. Each multimedia stream varies its sending rate \( R \) when congestion happens, and fairly shares the link capacity. Moreover, it does not compress the TCP traffic. In this simulation, the weight \( w \) was set at 1. Therefore, each multimedia stream shared the same ratio of network resources. The decreasing factor \( b \) was set at 0.5, and the system reduced its sending rate by half once it detected a loss. Figure 5.8 shows results from the same simulation done but with different priority weight values, where \( w = 1, 1.5, 2 \), for different multimedia streams. As indicated in Figure 5.1, the sending rate \( R \) is reduced to \( R = R \times (1 - b) \times w \). Therefore, the weight factor \( w \) scales the ratio of the sending rate \( R \) between different multimedia streams. The results, given in Figure 5.8, show that each stream shares the capacity of the available channel bandwidth according to the ratio of its weight, \( w \).

Figure 5.9 and Figure 5.10 show the change of throughput in the bottleneck link when decreasing factors \( b = 0.25 \) and 0.75, respectively, are used. The results show that when a large decreasing factor \( b = 0.75 \) is used, the allocation of the channel resource between streams takes longer to reach a steady state compared to when a smaller decreasing factor, \( b = 0.25 \), is used. This is because a larger decreasing factor \( b \) results in a larger fluctuation of the sending rate \( R \), as can be observed in Figure 5.10. As a result, more time is needed for the sending rate
$R$ to become steady. However, the use of a large $b$ gives a fair share of resources between multimedia streams, no traffic can fully utilize the network resources itself.

As shown in Figure 5.10, each stream shares the channel resources according to the ratio of its assigned weight, $w$. The results in Figure 5.9 suggest that the stream with weight $w = 2$ always utilizes the resources itself. The result indicates poor resource allocation.

Furthermore, the average sending rate $R$ for each multimedia stream was calculated. These are listed in Table 5.1. For a real-time multimedia-on-demand system, the sending rate $R$ of the source is directly proportional to the multimedia bitrate $B$ of the encoder. Therefore, the implication is that the higher the sending rate $R$, the higher the multimedia bitrate $B$ and, hence, the higher the quality of the bitstream. The average sending rate $R$ of the multimedia traffic with resource allocation when different weight factors are applied is shown to be higher than that when resource allocation, i.e. $w = 1$ for all the weight factors, is not used. The results show that using resource allocation in different multimedia streams does improve the average streaming quality. Similar results were criticized in [5], in which the importance of resource allocation in improving the quality of multimedia content even across heavily congested connections, was stressed.

The layered encoded video, as discussed, can provide further quality adaptation for congestion controlled streaming media, as stated in [8]. In this simulation, multimedia stream 1 (MM1) was assumed to be the base layer, “BL”, encoded
Table 5.1: A Comparison of Average Sending Rate of Multiple Multimedia Streams

<table>
<thead>
<tr>
<th>Layer-Encoded</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Video 1</td>
<td>BL</td>
<td>EL1</td>
<td>EL2</td>
<td>1</td>
<td>0.25</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1.5</td>
<td>1</td>
<td>1.8</td>
<td>1.13</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>0.72</td>
<td>0.78</td>
<td></td>
</tr>
<tr>
<td>Layer-Encoded</td>
<td>BL</td>
<td>EL1</td>
<td>EL2</td>
<td>1</td>
<td>0.5</td>
</tr>
<tr>
<td>Video 2</td>
<td>2</td>
<td>1.5</td>
<td>1</td>
<td>1.32</td>
<td>1.07</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0.98</td>
<td>0.7</td>
<td>0.98</td>
<td></td>
</tr>
<tr>
<td>Layer-Encoded</td>
<td>BL</td>
<td>EL1</td>
<td>EL2</td>
<td>1</td>
<td>0.75</td>
</tr>
<tr>
<td>Video 3</td>
<td>2</td>
<td>1.5</td>
<td>1</td>
<td>1.24</td>
<td>1.01</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0.92</td>
<td>0.69</td>
<td>0.92</td>
<td></td>
</tr>
<tr>
<td>Layer-Encoded</td>
<td>BL</td>
<td>EL1</td>
<td>EL2</td>
<td>1</td>
<td>0.5</td>
</tr>
<tr>
<td>Video 4</td>
<td>1</td>
<td>1</td>
<td>0.96</td>
<td>0.96</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>0.94</td>
<td>0.94</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>0.93</td>
<td>0.93</td>
<td></td>
</tr>
</tbody>
</table>

BL : Base Layer encoded stream
EL : Enhancement Layer encoded stream

video, while multimedia streams 2 (MM2) and 3 (MM3) were the enhancement layer encoded video “EL1” and “EL2”, respectively. The base layer “BL” encoded video stream was always set to a large $w$ value i.e. $w = 2$, while other enhancement layer encoded streams had smaller $w$ values i.e. $w = 1.5$ and 1, respectively. The results are listed in Table 5.1. Shown is the fact that a system with resource allocation achieves a higher average sending rate than that of a system without resource allocation, i.e. when $w = 1$ is used for all the priority weights. A system with resource allocation improves the streaming quality when the network is congested.
Figure 5.7: Throughput of the Traffic in the Bottleneck Link, $w=1$, $a=1$, $b=0.5$, (Upper) $TIL=0.1s$, (Bottom) $TIL=4s$
Figure 5.8: Throughput of the Traffic in the Bottleneck Link, Different Weights $w=1, 1.5, 2$, $a=1, b=0.5$, (Upper) $TIL=0.1s$, (Bottom) $TIL=4s$
Figure 5.9: Throughput of the Traffic in the Bottleneck Link, Different Weights w=1, 1.5, 2, a=1, b=0.25, (Upper) TIL=0.1s, (Bottom) TIL=4s
Figure 5.10: Throughput of the Traffic in the Bottleneck Link, Different Weights w=1, 1.5, 2, a=1, b=0.75, (Upper) TIL=0.1s, (Bottom) TIL=4s
Chapter 6

Gateway-assisted Congestion Control

6.1 Gateway-assisted Congestion Control

When a multimedia stream is transported over packet-switched networks, it assumes a constant network delay in order to maintain the timing relationship between the packets. It is difficult, however, if not impossible, to maintain a constant network delay. As a result, multimedia traffic usually experiences impairment due to network delay variation, or delay jitter, and this may lead to a degradation in the QoS.

The situation worsens when there are a large number of delay jitter corrupted multimedia packets (multimedia packets that have accumulated delay jitter larger than that of the client’s delay jitter tolerance) in the network. In this case, although the delay jitter corrupted multimedia packets are rendered useless when
received by the client, they continue to consume network bandwidth and, thus, increase the congestion of the network for multimedia packet delivery. This kind of congestion cannot be solved using traditional active buffer management schemes such as RED [15] or DropTail in gateway-assisted packet-switched network congestion control.

It is assumed that multimedia streams are transported by UDP traffic that share the network with TCP traffic. The multimedia packets that have accumulated excessive delay jitter are useless to the client. Therefore, it is a waste of network resources to continue to forward these packets in the network. An efficient gateway-assisted congestion control for streaming traffic can be implemented to detect and discard multimedia packets that violate the delay jitter tolerance. The efficiency of such gateway-assisted congestion control algorithm can be improved by considering the multimedia packet’s “closeness” to its destination in the discard decision process.

JD is proposed to improve the QoS by detecting and discarding multimedia packets that have accumulated a large delay jitter, so as to maintain a high bandwidth for packets that stay within the multimedia stream’s delay jitter tolerance.

JD assumes the gateway is able to classify a packet as a TCP, or as multimedia streaming packet. For example, a packet may be recognized as a streaming packet based on the encoding in the protocol field in the IP version 4 packet header. To keep track of the accumulated delay jitter in the multimedia UDP packets, a small delay jitter counter is included in the network layer (i.e. IP) header. This
delay jitter counter, in each packet’s header, is updated in each gateway by the addition of the estimated delay jitter experienced by the multimedia packet in the current gateway. The updated delay jitter is quantized into four values and stored in the delay jitter counter. The delay jitter counter is implemented in the Type-of-Service field in the IP version 4 packet header. The Type-of-Service field itself contains (from left to right) a three-bit Precedence field, three flags, D, T, and R, and two unused bits. These two unused bits are used to represent all the four quantized delay jitter values.

A simple discard algorithm is applied to each streaming packet, in which the delay jitter counter is used to select the discard threshold. For the purposes of this study, we investigated the notion that a measure of how far, or close, the packet is from its destination should be taken into consideration for the discard decision. The idea here is that a streaming packet that has accumulated a large delay jitter value should not be dropped if it is close to the client. It was assumed that the TTL field in the IP packet header could be used as a measure of this “residual distance”. To use the TTL field for this purpose requires the initialization of the TTL field [20]. Instead of using a fixed default “safe value” for all destinations, the server of a streaming packet needs to set a TTL value that reflects the actual hop-count to the client. More generally speaking, this initial TTL value should be set according to the end-to-end delay budget, instead of the hop count, because the delay on each hop is a function of the link bandwidth, and this is non-uniform. To use the TTL, servers and clients are required to exchange hop-count/delay-budget information during media session set-up, and when the hop-count changes due
Figure 6.1: System Block Diagram for the proposed Jitter Detection Gateway to routing updates. A safety margin may be needed in the initial TTL value for protection against various scenarios that cause unnecessary discarding of packets.

6.1.1 Classification of Input Traffic into FIFO output queue

In order to prevent the "locked out" situation of non-aggressive TCP traffic when there is high bandwidth multimedia UDP traffic across the channel, and in order to achieve a certain degree of TCP-friendliness of the gateway, we propose that a classification of TCP and UDP traffic be done before putting them into the FIFO queue. By classifying the input traffic into different classes using the classifier, different and suitable queue management schemes for managing the traffic can be applied. In our research, it was assumed that there were only two traffic classes, TCP, and UDP. We assumed that the UDP traffic was a constant bitrate multimedia traffic and that TCP traffic were some variable bitrate services like FTP service. In fact, we further classified the service classes according to the delay requirement constraints, as indicated in “Differentiated Service” (DiffServ). For different delay jitter sensitive classes, we simply applied different values of parameters to achieve the QoS requirement. Figure 6.1 shows the system block diagram. To classify the input traffic, the classifier simply looks for the source
and destination addresses inside the packet header of the incoming packet. TCP packets are subjected to the original RED algorithm. For UDP packets, or other delay jitter sensitive traffic packets, our proposed JD with delay jitter concerned is applied, as indicated in Figure 6.1. Although the packets are classified, there is still only one output queue of packets, and this is shared by all traffic classes. All packets are enqueued and dequeued in a FIFO manner. By doing this, the design of the router or gateway is simplified. And, certain “better-than-best-effort” services are provided.

6.1.2 Delay Jitter Aware QoS for Multimedia Traffic

Figure 6.1 illustrates the operation of a gateway in which two types of traffic classes are assumed: UDP and TCP. The UDP traffic carry constant sending rate multimedia traffic, and TCP traffic carries variable bitrate services such as FTP. Input traffic to the gateway is classified into TCP, or UDP streaming traffic. TCP traffic is subjected to the RED scheme, while the UDP streaming traffic is subjected to the JD scheme before being enqueued into the output queue. To simplify this discussion, the output queue is assumed to be FIFO. Also, the packet size of the streaming traffic is assumed to be fixed. In reality, multimedia traffic, such as MPEG-2 VBR audio traffic, is usually chopped down to a constant packet size to fit into the network environment. In that case, a simple multimedia frame (either video frame or audio frame) requires a different number of packets in VBR multimedia bitrate condition. Further note that losing some of the packets still allows a partial decoding of the multimedia data (e.g. in MPEG audio and video
coder). This justifies our assumption that the queue management scheme can consider each packet separately.

6.1.3 Jitter Detection (JD) Scheme 1

Figure 6.2 shows the pseudo code of the JD scheme 1 used in the gateway congestion control. The time delay, delay, is encountered by a streaming packet in the gateway. delay can be estimated from the current buffer occupancy CBO of the output queue together with the estimated available output link capacity LC, such that

\[ delay = CBO \times \text{Packet.Size} / LC. \]  \hspace{1cm} (6.1)

The average delay ave_delay experienced by the incoming packets is estimated as an “Exponentially Weighted Moving Average” (EWMA) of the delay.

\[ ave\_delay = ave\_delay \times (1 - w_d) + delay \times w_d. \]  \hspace{1cm} (6.2)

By means of EWMA, short term increases in the average delay can be prevented by the smoothing effect of the weighted average. The delay jitter experienced by the multimedia streaming packet is given by

\[ jitter = delay - ave\_delay. \]  \hspace{1cm} (6.3)

The end-to-end delay jitter counter v that tracks the accumulated delay jitter of the multimedia packets is defined in Figure 6.2 and 6.3. v is updated by quantizing the accumulated delay jitter stored by v plus the estimated delay jitter experience by the multimedia packets in the gateway into one of the four
Initialization: delay=0, ave_delay=0, v=1

For each classified packet arrived:
  1) Estimate the output link capacity: LC
  2) Update the current buffer occupancy: CBO
  3) Calculate the delay:
     \[ \text{delay} = \text{CBO} \times \text{Packet\_Size} / \text{LC} \]
  4) Calculate the average delay:
     \[ \text{ave\_delay} = (1 - wd) \times \text{ave\_delay} + wd \times \text{delay} \]
  5) Calculate the bound:
     \[ \text{bound} = \text{fixed\_th} / 6; \]

Set the current delay jitter allowance value A:
  if (v=1) \{ A = -0.5 \times \text{fixed\_th} + 1 \times \text{bound} \}
  if (v=2) \{ A = -0.5 \times \text{fixed\_th} + 2 \times \text{bound} \}
  if (v=3) \{ A = \text{bound} \}
  if (v=4) \{ A = 2 \times \text{bound} \}

Decision of dropping packets:
  \[
  \text{jitter} = \text{delay} - \text{ave\_delay} \\
  \text{threshold} = \text{jitter} + A \\
  \text{if} (-0.5 \times \text{fixed\_th} \leq \text{threshold} \leq 0.5 \times \text{fixed\_th}) \\
  \{ \text{queue this packet} \}
  \text{else} \{ \text{discard this packet} \}
  \\
  \text{Update the delay jitter allowance counter v:} \\
  \text{if threshold in Region i ; 1 \leq i \leq 4} \\
  \{ v=i \}
  \text{else threshold in Discard Region} \\
  \{ \text{Discard this packet; v keeps unchanged} \}

Variables
LC: Estimated available output link capacity (bps)
CBO: Current Buffer Occupancy
delay: Estimated packet delay (s)
ave_delay: Estimated average packet delay in queue (s)
v: delay jitter allowance counter
threshold: dropping threshold
bound: quantization step size
Fixed Parameters
wd: delay weight
Packet\_Size: received packet size (byte)
fixed\_th: fixed threshold

Figure 6.2: JD Scheme 1 for Delay Jitter Sensitive Traffic
regions, as in Figure 6.2. We use \( \text{bound} \), which equals to \( \text{fixed.th}/6 \), to define the region, and each region has a width of \( 1.5 \times \text{bound} \), as shown in Figure 6.3.

Finally, the dropping threshold \( \text{threshold} \) is determined by

\[
\text{threshold} = \text{jitter} - \frac{\text{fixed.th}}{2} \left(1 + \frac{(v + \left| \frac{v}{3} \right|)}{3}\right).
\]  

(6.4)

The multimedia packet is sent to the output queue if, and only if,

\[
-\frac{\text{fixed.th}}{2} \leq \text{threshold} \leq \frac{\text{fixed.th}}{2}.
\]  

(6.5)

The proposed JD scheme deals with the delay jitter control problem when the channel of concern, or the client’s buffer size, is limited. In order to ensure that the client buffer never underflows (or overflows), the inflow rate must match that of the outflow rate. This is achieved via feedback control, or by dropping those packets that arrive early. Discarding those packets with a large accumulated jitter
helps to better control the inflow rate into the client’s buffer.

### 6.1.4 Jitter Detection (JD) Scheme 2

The *fixed.th* in the JD scheme 1 determines the allowable delay jitter tolerance for each multimedia packet. It must be chosen with great care because the delay jitter accumulated at each gateway is nonuniformly distributed. If a tight *fixed.th* is used, it is highly probable that the multimedia packets will be dropped early in the gateway path. On the other hand, if a loose *fixed.th* is used, it is highly probable that the multimedia packets received by the client will be useless due to delay jitter tolerance violation. An optimal performance can be achieved only if different *fixed.th* are used at the gateway path. In particular, as the multimedia packets enter the gateway that is closest to the client, looser, i.e. bigger, values of *fixed.th* should be used.

As discussed in Section 6.1, such *fixed.th* adaptation can be achieved by extracting the hop count knowledge carried in the TTL field. To simplify this discussion, we assume that the value in the TTL field in the packet header is the remaining hop count. JD scheme 1 and JD scheme 2 are, therefore, differed by the additional residual hop count information. It should be noted that the actual residual distance estimation, as discussed in Section 6.1, is difficult to obtain. The assumption that the TTL field carries the residual hop count allows an understanding that a gain in performance is achievable with the estimated residual distance information. The pseudo code of the modified JD scheme 2 is shown in Figure 6.4. The TTL value is first set to equal that of the hop count
Setting the fixed threshold value:
(a) Determine TTL value of packet
   Set $\sigma$ according to specific TTL value
   for $\text{TTL} = \{ n, n-1, n-2, \ldots, 2, 1 \}$
   such that $\sigma_n < \sigma_{n-1} < \sigma_{n-2} < \ldots < \sigma_2 < \sigma_1$
(b) $\text{fixed}_\text{th} = \text{fixed}_\text{th} + \sigma_n$

Calculate the bound:
$\text{bound} = \text{fixed}_\text{th} / 6$

Set the current delay jitter allowance value $A$:

$\therefore$

$\text{same as JD Scheme 1}$

$\therefore$

Figure 6.4: JD Scheme 2 with TTL Constraint

at the server, and is decremented by one every time the packet passes through
a gateway. The $\text{fixed}_\text{th}$ is adapted by adding $\sigma_i$ with $i$ being the TTL value,
such that the adapted $\text{fixed}_\text{th}$ is given by $\text{fixed}_\text{th} + \sigma_i$. $\sigma_i$ should be chosen so
that $\sigma_n < \sigma_{n-1} < \sigma_{n-2} < \ldots < \sigma_2 < \sigma_1$, where $n$ is the total hop count in the
gateway path. The rest of the JD algorithm remains the same.

6.1.5 Performance Bound and Analysis of Dropping Decision

The performance of the Jitter Detection can be examined by measuring the delay
jitter value of the received packet using the sample mean delay $\bar{D}$ due to the use
of the finite number of sample packets $N_s$. By using the Chebyshev inequality,
an upper bound on the probability of dropping packet $P(\text{drop})$ and the initial
value of \( fixed.th \) can be derived. Assume that the delay values of the received multimedia packets \( D_i \) are independent and are identically distributed with a standard deviation of \( \sigma_d \). The sample mean delay \( \bar{D} = \frac{1}{N_s} \sum_{i=1}^{N_s} D_i \).

The Chebyshev inequality can be applied to obtain a loose bound that helps to find the relationship between the probability of dropping packets, \( P(\text{drop}) \) in the estimation of delay jitter of the received packets using \( D_i \) and \( \bar{D} \), the number of sample packets \( N_s \), and the dropping threshold \( fixed.th \). The dropping threshold \( fixed.th \) is defined as \( fixed.th = 2\alpha_j J_{\text{max}} \) where \( J_{\text{max}} \) is the maximum delay jitter value allowed for the desired application and \( \alpha_j \in (0, 1) \).

\[
P(\text{drop}) = P\{ |D_i - \bar{D}| \geq \frac{fixed.th}{2} \} \leq \frac{4\sigma_d^2}{fixed.th^2}. \tag{6.6}
\]

\[
P(\text{drop}) = P\{ (\bar{D} \geq D_i + \frac{fixed.th}{2}) \text{or} (\bar{D} \leq D_i - \frac{fixed.th}{2}) \}
\leq \frac{4\sigma_d^2}{fixed.th^2}. \tag{6.7}
\]

\[
P(\text{drop}) = P\{ (\bar{D} \geq D_i + \alpha_j J_{\text{max}}) \text{or} (\bar{D} \leq D_i - \alpha_j J_{\text{max}}) \}
\leq \frac{\sigma_d^2}{\alpha_j^2 J_{\text{max}}^2}. \tag{6.8}
\]

Therefore, given the number of sample packets \( N_s \) to estimate the sample mean delay \( \bar{D} \) during the initial state, it can be used to define the values of the parameter \( fixed.th \), \( J_{\text{max}} \) and \( P(\text{drop}) \). Also, eq.(6.8) provides a performance bound of the dropping probability \( P(\text{drop}) \) with \( P(\text{drop}) \leq \frac{\sigma_d^2}{\alpha_j^2 J_{\text{max}}^2} \).

This packet dropping decision model of JD schemes demonstrates that there is a good analytical reason for changing the \( fixed.th \) according to the “closeness” to the destination in order to vary the probability of the dropping packets. It has been analytically shown that the larger the \( fixed.th \), the lower the probability of
dropping the packets. This explains why the JD Scheme 2 allows more packets to pass through the gateway when compared to that of JD Scheme 1.

6.1.6 Simulation

Simulations were performed using NS2 [21] to investigate the performance of the proposed algorithms. Figure 6.5 shows the simulation topology. There are two TCP sources (FTP traffic) and two UDP sources (Multimedia traffic i.e. MPEG-2 audio coder with encoded/multimedia bitrate 0.8Mbps) connecting to the streaming gateway R1 by links with a maximum capacity of 10Mbps and a propagation delay of 3ms. The MPEG-2 audio sequence test sample was encoded in variable multimedia bitrate (VBR) format and was carried by constant channel bitrate (CBR) UDP traffic in this simulation. For simplicity, we use UDP packets to represent the audio streams in the following discussion. Figure 6.5 details the settings of the traffic through the network. The packets go through a chain of twelve gateways (R1-R12) before arriving at the client. Those gateways (R1-R12) are connected to each other by links with a maximum capacity of 1.5Mbps and a propagation delay of 20ms. The bottleneck of such a network topology is in the chain of those twelve gateways. RED and JD are implemented in the gateways for congestion control of the TCP and UDP packets, respectively. Figure 6.5 also details the parameter settings of the simulations. Several simulations were performed in order to investigate the TCP-friendliness of the system, the delay jitter, and the useful throughput of the multimedia traffic. Two sets of additional simulations were performed using the DropTail and the RED congestion control.
schemes on the multimedia traffic for performance comparison with that of the proposed JD schemes.

6.1.7 Results and Discussion

TCP-friendliness

It is well-known that UDP does not respond to packet loss. Greedy high bandwidth UDP traffic typically lowers the throughput of TCP flows. Our proposal was to classify TCP and UDP traffic and apply different buffer management schemes to the two sets of traffic before putting them into the FIFO queue. It
was important that the proposed JD schemes had a certain TCP-friendliness so that the throughput of the TCP flows was not lowered. The simulation results, provided in Figure 6.6a, show the average throughput of TCP traffic against different packet sizes under different buffer management schemes for the UDP traffic and RED for the TCP traffic. It can be observed that the proposed JD scheme 1 and 2 achieve the same level of TCP throughput compared to those of other buffer management schemes. Similarly, in Figure 6.6b, the average throughput of UDP traffic against different packet sizes, from the same simulation as that in Figure 6.6a, is presented. The average throughput of UDP in the proposed schemes decreases as the packet size increases. Also, it is lower than that of the DropTail and RED algorithms. This is because UDP packets with a large accumulated delay jitter are discarded. Figure 6.7a shows the throughput of UDP traffic when it passes through each gateway (R1-R12). The result shows the effect of the TTL counter in JD Scheme 2. A lower packet drop rate is observed when the gateway is “closer” to the client. The TCP-friendliness of the proposed gateway-assisted buffer management scheme is directly related to the ratio of the UDP throughput to the TCP throughput. In Figure 6.7b the degree of TCP-friendliness with different packet sizes for different gateway-assisted buffer management schemes, is shown. It can be observed that the proposed JD scheme achieves a level of TCP-friendliness similar to that of other simulated buffer management schemes.
Figure 6.6: (a) Average Throughput of TCP against Different Packet Sizes, (b) Average Throughput of UDP against Different Packet Sizes

Figure 6.7: (a) Average Throughput of UDP Packets at each Router using the proposed JD Scheme 2, (b) Ratio of Average Throughput between UDP and TCP

113
Delay Jitter of Multimedia Traffic

Simulations were performed to investigate the delay jitter of the multimedia traffic. The curve in Figure 6.8a shows the average delay jitter for some received UDP packets when different active queue management schemes, defined as the average value of delay jitter from $J_0$ to $J_1$, are used. The same delay jitter pattern is found in all the schemes. The periodic pattern found in all the delay jitter figures is mainly due to the congestion control mechanism of the TCP traffic. The results show that our schemes ensure a lower average delay jitter than is obtained when using DropTail and RED, as shown in Figure 6.8a.

To investigate the scalability and stability of the JD mechanisms, we further simulated the same network topology with two more identical UDP traffic (MPEG-2 audio stream) than used in the previous simulation. This allowed us to investigate the performance of the proposed JD algorithms under a heavy traffic network situation. The simulation results, given in Figure 6.8b, show that even under heavy traffic conditions, the JD schemes still ensure a lower average delay jitter than that obtained using DropTail and RED.

Useful Goodput of Multimedia Traffic

Multimedia UDP traffic is delay jitter sensitive. When the delay jitter carried by the packet is larger than a tolerance threshold, it is regarded as useless even if it successfully arrives at the client. Congestion occurs when there is not enough available network bandwidth. Forwarding a packet that arrives too early, or too late, reduces the available bandwidth of the channel and, thus, reduces the delay
Figure 6.8: Average Delay Jitter for some received UDP Packets using Different Queue Management Schemes (a) Original Network with 2 TCP Traffic and 2 UDP Traffic, (b) Modified Network with 2 TCP Traffic and 4 UDP Traffic

jitter tolerance of the gateways along the path. The reduced delay jitter tolerance results in more packets arriving at the client with too much delay jitter. This makes them useless, or causes them to be dropped when JD is applied.

Therefore, it is the “useful” goodput of the multimedia traffic that is of concern, not the throughput. “Useful” means that the multimedia packet arrives at the client within the delay jitter tolerance, i.e. $J_t < UsefulThreshold$, as shown in Figure 1.4. The “useful” multimedia packet can meet its scheduled playout position and be played out to the user. In the simulation, the $UsefulThreshold$ was set to equal to, or smaller than, one half of the playout interval, as shown in Figure 1.4. The $UsefulThreshold$ determines whether the received multimedia packets are “useful” or not, while the given $J_{max}$ determines whether the packet in the queue should be discarded or not. They are actually different in meaning and, at the same time, they are directly proportional to each other in value.

The range of useful threshold values used in the simulation were in the range of 0.05s to 0.1s. Such settings are useful for audio streaming such as MPEG-2 with
an audio bit rate from 6kbps to 1Mbps, as in [19]. This means that the client’s buffer only needs to store up to 0.1Mbits of data in order to compensate for the effect of delay jitter of the multimedia packets. Simulations were performed to investigate the average useful goodput of multimedia UDP traffic against different packet sizes and different useful thresholds. Results are shown in Figure 6.9a and 6.9b, respectively. The results show that our schemes can achieve a higher average useful goodput of multimedia UDP traffic than those using DropTail and RED. Moreover, the percentage of useless packets in those received packets versus different packet sizes, and different useful thresholds, were also investigated. The simulation results are shown in Figure 6.10a and 6.10b, respectively. Again, the results show that our schemes can maintain a low percentage of useless packets in those received packets when compared to that obtained by using DropTail and RED.

Notice that the simulation results obtained using NS2 have a random variation due to the nature of the Monte-Carlo simulation employed in NS2. As a result, there may be some inconsistencies in the individual simulations. However, the overall trend, clearly observable in Figure 6.9 and 6.10, agrees well with the theoretical development. For one or two particular experimental setups, the results vary a little from the predicted theoretical results. Since the variation is small, we consider the simulation results to be acceptable, and have reported them in this dissertation. The variation in the simulation results is more obvious in large packet size because the useful goodput is small. As a result, the percentage of variation increases, despite the magnitude of the simulation variation remains the
Figure 6.9: (a) Useful Goodput of UDP Traffic against Different Packet Sizes (Useful Threshold = 0.1s), (b) Useful Goodput of UDP Traffic against Different Useful Threshold (Packet Size = 900 bytes)

Figure 6.10: (a) Percentage of Useless UDP Packets against Different Packet Sizes (Useful Threshold = 0.1s), (b) Percentage of Useless UDP Packets against Different Useful Threshold (Packet Size = 900 bytes)

Audio Quality Comparison between different Queue Management Schemes

The results given in Figure 6.6b show that the proposed JD schemes for multimedia streams such as audio achieves a relatively lower received throughput of multimedia UDP traffic than that obtained using RED and DropTail. However, if the “useful” received multimedia packets are considered, there are more “useless” packets when RED or DropTail is used than when the proposed modified JD is
Figure 6.11: Audio Quality Comparison between Different Queue Management Schemes

employed. The important results, given in Figure 6.11, show that the proposed JD schemes can achieve a better quality for MPEG-2 audio stream than that using RED and DropTail within a given useful threshold. It shows that the JD scheme has up to about 0.2 – 0.3dB advantage in PSNR over RED, and up to about 0.5dB advantage over DropTail. The results were computed and reported as EWMA for every window size of one second. As a result, a smaller fluctuation in PSNR is observed in the audio playback obtained by the simulation.
6.2 Gateway-assisted Congestion Control for Layered Multimedia Multicast

6.2.1 Layered Multimedia Multicast

Multicast with a layered coding scheme has been shown to be an effective solution for improving the quality of multimedia streaming through bitrate adaptation. Layered multicast has been a hot topic in the networking community as well as the coding community [22, 23, 24, 25, 27]. The well-known layered multimedia multicast algorithm, RLM [22], sends each video layer over a separate multicast group. The client periodically joins a higher layer’s group to explore the available bandwidth. If packet loss is detected, the client leaves the group. However, the RLM algorithm is not TCP-friendly [23, 25, 26]. This can be improved by using equation-based rate control on the client side [25, 26]. Another attractive layered multicast approach is prioritized transmission [24, 27]. In this approach, the server assigns different priorities to each layer according to their level of importance. During congestion, the routers will first drop the low priority packets, i.e. the enhancement layer packets. In this thesis, we further investigated the application of jitter detection for layered multimedia streaming aiming to improve the multicasting QoS. An analytical analysis, using queuing theory, was also presented to show the efficiency of the proposed scheme.
6.2.2 Jitter Detection (JD) for Layered Multimedia Multicast Traffic

Figure 6.12 shows the pseudo code of the modified JD scheme for layered multimedia multicast traffic. The delay, \( \text{ave}_{\text{delay}} \), jitter, \( v \) and threshold are estimated and maintained in a similar way as that in the JD algorithm described in Section 6.1.3. The modified JD for layered multimedia multicast preserves the base layer traffic with best effort when passing through a gateway with JD congestion control. It can identify the level of priority for each packet in layered multimedia multicast traffic. The priority of the dropping decision is given to those higher level layer (enhancement layer) multicast packets. These are not considered in the original JD algorithm.

6.2.3 Analytical Analysis

As illustrated in Figure 6.13, there are \( N_i \) UDP flows with arrival rate \( \gamma_i \) from the packet classifier that go through the JD mechanism. For simplicity, it is assumed that all the packets are layered multimedia traffic from one source, and that there is no other TCP traffic passing through the gateway. The arrival rate of the “base” layer traffic is \( \gamma_1 \), while other “enhancement” layer traffic have the arrival rate \( \gamma_2, \gamma_3, ..., \) and \( \gamma_{N_i} \), respectively. According to the JD scheme discussed in Section 6.2.2, the \( N_i^{th} \) flow has the highest level of priority will be dropped first when there is network congestion, while the “base” layer, 1\text{st} flow, is the last category to be dropped. Therefore, we set \( N_i \) different dropping fraction
Initialization: delay=0, ave_delay=0, v=1

For each classified packet arrived:
(1) Estimate the output link capacity: LC
(2) Update the current buffer occupancy: CBO
(3) Calculate the delay:
   delay = CBO*PacketSize/ LC
(4) Calculate the average delay:
   ave_delay = (1 - wd) * ave_delay + wd * delay
(5) Calculate the bound:
   bound = fixed_th / 6;

Set the current delay jitter allowance value A:
   if (v=1) { A = -0.5*fixed_th + 1*bound }
   if (v=2) { A = -0.5*fixed_th + 2*bound }
   if (v=3) { A = 1*bound; }
   if (v=4) { A = 2*bound; }

Decision of dropping packets:
   jitter = delay - ave_delay
   threshold = jitter + A
   if (-0.5*fixed_th <= threshold <= 0.5*fixed_th)
      { queue this packet }
   else { if (base layer packet)
      { discard the highest level x enhancement packet;
        if (only base layer packet in the queue)
        { discard this packet }
      }
      if (level x enhancement packet)
      { discard level y enhancement packet, where y>=x }
   }

   where 1 <= x,y <= N_l , N_l is the number of level
   in layered video

Update the delay jitter allowance counter v:
   if threshold in Region i ; 1 <= i <= 4
      { v=i }
   else threshold in Discard Region
      { Discard this packet;
        v keeps unchanged }

Variables
LC : Estimated available output link capacity (bps)
CBO : Current Buffer Occupancy
delay : Estimated packet delay (s)
ave_delay : Estimated average packet delay in queue (s)
v : delay jitter allowance counter
threshold : dropping threshold
bound: quantization step size

Fixed Parameters
wd : delay weight
PacketSize : received packet size (byte)
fixed_th: fixed threshold

Figure 6.12: Details of Modified JD scheme for Layered Multimedia Multicast Traffic
Figure 6.13: Queuing Model of Modified JD for supporting Layered Multimedia Multicast Traffic

\( \beta_i \) with \( \beta_1, \beta_2, \ldots, \beta_{N_i} \) for the \( N_i \) different flows respectively in this analysis.

Let

\[
\beta_i = P(\text{drop}) \times (1/N_i)^{N_i-i} \quad \text{for } i = 1, 2, \ldots, N_i - 1, N_i.
\]  

(6.9)

The packet sent to the Output Queue after JD scheme can be modeled as a \( M/M/1/K \) Markov Chain, where \( K \) being the maximum queue limit \( Q_{\text{max}} \). Let

\[
\lambda_k = \begin{cases} 
\gamma_1(1-\beta_1) + \gamma_2(1-\beta_2) + \cdots + \gamma_{N_i}(1-\beta_{N_i}) & \text{for } k = 0, 1, 2, \ldots, K - 1, \\
0 & \text{otherwise},
\end{cases}
\]

(6.10)

\[
\mu_k = \begin{cases} 
\mu & \text{for } k = 1, 2, \ldots, K, \\
0 & \text{otherwise},
\end{cases}
\]

(6.11)

where \( \lambda_k \) is the arrival rate of the output queue at the output of the JD operation and \( \mu_k \) is the service rate of the queue, which equals the \( \text{LinkCapacity/PacketSize} \).
Furthermore, let $p$ be the traffic intensity, which is given by

$$p = \frac{\lambda'}{\mu} = (\gamma_1(1 - \beta_1) + \gamma_2(1 - \beta_2) + \cdots + \gamma_{N_i}(1 - \beta_{N_i}))/\mu.$$  \hspace{1cm} (6.12)

Let $P_k$ be the probability of having $k$ packets in the output queue. Under steady state analysis, $\mu P_k = \lambda' P_{k-1}$,

$$\Rightarrow \quad P_k = \lambda'/\mu P_{k-1} = P_0 \prod_{i=0}^{k-1} \frac{\lambda'/\mu}{\mu}, \quad k \leq K,$$  \hspace{1cm} (6.13)

As $P_0 = (1 - \lambda'/\mu)/(1 - (\lambda'/\mu)^{K+1})$,

$$P_k = \begin{cases} 
(\lambda'/\mu)^k(1 - \lambda'/\mu)/(1 - (\lambda'/\mu)^{K+1}) & k \leq K, \\
0 & \text{otherwise,}
\end{cases} \hspace{1cm} (6.15)$$

$$P_k = \begin{cases} 
p^k(1 - p)/(1 - p^{K+1}) & k \leq K, \\
0 & \text{otherwise,}
\end{cases} \hspace{1cm} (6.16)$$

The average number of packets $\bar{N}_q$ in the queue is

$$\bar{N}_q = \sum_{k=0}^{K} kP_k$$

$$= \sum_{k=0}^{K} k(\lambda'/\mu)^k(1 - \lambda'/\mu)/(1 - (\lambda'/\mu)^{K+1})$$

$$= \sum_{k=0}^{K} k(p^k)(1 - p)/(1 - p^{K+1}).$$  \hspace{1cm} (6.17)

Using Little’s theorem, the average waiting time $\bar{W}$ of each packet in the queue equals

$$\bar{W} = \frac{Q}{\lambda'}$$

$$= \frac{1}{\lambda'} \sum_{k=0}^{K} k(\lambda'/\mu)^k(1 - \lambda'/\mu)/(1 - (\lambda'/\mu)^{K+1})$$

$$= \frac{1}{\lambda'} \sum_{k=0}^{K} k(p^k)(1 - p)/(1 - p^{K+1}).$$  \hspace{1cm} (6.18)

Several simulations were performed using the above equations to investigate the
Figure 6.14: (Upper) Average Number of Packets in the Queue against Arrival Rate, $\gamma$, (Bottom) Average Waiting Time in the Queue against Arrival Rate, $\gamma$, Increment Step for $J_{max}=0.5$
Figure 6.15: Upper: Average Number of Packets in the Queue against Traffic Intensity, $p$, Bottom: Average Waiting Time in the Queue against Traffic Intensity, $p$ Increment Step for $J_{\text{max}}=0.5$
effect of the maximum delay jitter allowance $J_{\text{max}}$, traffic intensity $p$, and the packet arrival rate of each of the flows to the average number of packets, and the average waiting time in the queue. For simplicity, we assumed the same arrival rate for all the flows so that $\gamma_i = \gamma$, which varies from 0.1Mbps to 0.6Mbps. The packet size was 900bytes for all the flows and the link capacity was 1.5Mbps. Thus, the service rate, $\mu$, was 1.5Mbps/900bytes. The maximum delay jitter allowance $J_{\text{max}}$ varies between 1s to 5s, with 0.5s increment. The corresponding maximum buffer size is set to be 3Mbits, which is a reasonable size for most layered multimedia applications. Figure 6.14 plotted the average number of packets and the average waiting time in the queue under different arrival rate $\gamma$. Since the probability of dropping packet $P(\text{drop})$ is affected by the maximum delay jitter allowance $J_{\text{max}}$. Therefore, varying $J_{\text{max}}$ will affect the average number of packets and average waiting time in the queue. In all the simulated cases, the JD scheme which provides a tight control on the delay jitter allowance will achieve a lower average number of packets and average waiting time in the queue than that achieved when no control scheme is applied. Here, the no control scheme simply means that no packets are dropped until a full buffer condition occurs. This is equivalent to the DropTail scheme.

Similarly, Figure 6.15 plotted the average number of packets and the average waiting time in the queue versus different traffic intensity $p$. The results show that the proposed modified JD scheme can achieve a lower average number of packets and average waiting time in the queue with different $J_{\text{max}}$ than that obtained with the DropTail scheme. Furthermore, it can be observed that the
average number of packets and the average waiting time in the queue with larger $J_{\text{max}}$ approaches that of the DropTail scheme. The larger the $J_{\text{max}}$, the larger the average number of packets and the larger the average waiting time in the queue. Because the larger the $J_{\text{max}}$, the larger the fixed_th value and this decreases the $P(\text{drop})$ as discussed in Section 6.1.5. This results in fewer number of packets being dropped by the JD schemes, and hence increases the average number of packets and the average waiting time in the queue.

This analysis also allows us to choose a suitable value of $J_{\text{max}}$ in order to achieve certain average delay jitter bound for the layered multimedia streams, which is related to the average waiting time in the queue. In the actual implementation of the proposed JD scheme, it is crucial to set an appropriate value to the dropping threshold fixed_th. The fixed_th should be set to $2\alpha J_{\text{max}}$, as discussed in Section 6.1.5. As a result, deciding upon the appropriate $J_{\text{max}}$ is critical. Figure 6.16 plotted the average waiting time in the queue versus different $J_{\text{max}}$. It can be observed that the most “suitable” values are in the range of 3s to 3.5s in this simulation. This is because the average waiting time in the queue begins to take off when $J_{\text{max}}$ is larger than 3.5s.

### 6.2.4 Simulation

Network simulations were performed using NS2 [21] to investigate the performance of the proposed scheme. Figure 6.17 shows the simulation topology. There are two TCP sources (FTP traffic of 1Mbps) and two UDP sources (one for the
Figure 6.16: Average Waiting Time in the Queue against Maximum Delay Jitter Allowance, $J_{max}$
1Mbps base layer of the layered multicast, and the other is the 0.5Mbps enhancement layer traffic), and a 1Mbps multicast traffic connected to the streaming gateway R1 by the links with 10Mbps capacity and 3ms propagation delay. Figure 6.17 details the settings of the traffic through the network. The packets go through a chain of twelve gateways (R1-R12) before arriving at the client. These gateways (R1-R12) are connected to each other by links with a 1.5Mbps capacity and a 20ms propagation delay. The bottleneck of such a network topology is in the chain of those twelve gateways. RED and JD were implemented in the gateways for congestion control of the TCP and Multicast packets respectively. Figure 6.17 details the parameter settings of the simulations. Several simulations were performed to investigate the TCP-friendliness of the system, the useful throughput, and the quality comparison of the multimedia traffic. Similar simulations were performed using the RED congestion control scheme as a replacement for the JD scheme for multimedia traffic. The simulation results are useful for performance comparison in terms of TCP-friendliness and the quality of multimedia streaming using different schemes.

6.2.5 Results and Discussion

TCP-friendliness

It is well known that UDP does not respond to packet loss. Greedy high bandwidth UDP traffic typically lowers the throughput of TCP flows. Our proposal was to classify TCP and UDP traffic and apply different buffer management schemes to the two traffic before they were enqueued to the FIFO queue. It was
Simulation Settings:
Simulation Time = 30s
Time-To-Live = 14
Queue limit in each node = 25 packets
Packet Sizes = 900 bytes

(a) RED parameter settings:
max_threshold = 15 packets
min_threshold = 5 packets
wq = 0.002
maxp = 0.1

(b) JD parameter settings:
fixed_th = 5.52s
wd = 0.02

Figure 6.17: Simulation Topology for Modified JD Scheme with Layered Multimedia Multicast Traffic
shown in Section 6.1.7 that the JD schemes have a certain TCP-friendliness in that they do not lower the throughput of the TCP flows. The simulation results given in Figure 6.19 show the average throughput of all traffic passing through the bottleneck link under different buffer management schemes. It can be observed that the proposed modified JD scheme for layered multicast can achieve the same level of TCP throughput as that of other buffer management schemes like RED. However, the throughput of multimedia traffic is lower than that obtained when RED is used. This is because UDP packets with a large accumulated delay jitter are discarded. It can be observed that the proposed modified JD scheme for layered multicast can achieve a TCP-friendliness similar to that of other simulated buffer management schemes such as RED.

**Quality Comparison of Received Layered Multimedia Multicast Traffic**

Multimedia UDP traffic is delay jitter sensitive. When the delay jitter carried by the packet is larger than a tolerance threshold, it is regarded as useless, even if it successfully arrives the client. As a result, it is the “useful” goodput of the multimedia traffic that is of concern, not the throughput. “Useful” refers to the fact that the multimedia packet is received by the client within the delay jitter tolerance, i.e. $J_i < UsefulThreshold$, as shown in Figure 1.4. In the simulation, $UsefulThreshold$ was set to equal to, or smaller than, one half of the playout interval. Here, $UsefulThreshold$ was used to determine the usefulness of the received multimedia packets, while $J_{max}$ was used to determine whether the packet in the queue should be discarded or not. They are actually different in
Figure 6.18: Throughput in the Link between R12 and Client Node of Layered Multicast 1, (Upper) by RED, (Bottom) by Modified JD
Figure 6.19: Throughput in the Bottleneck Link, (Upper) by RED, (Bottom) by Modified JD
meaning, but, at the same time, they are directly proportional to each other in value. The range of useful threshold values used in the simulation was in the range of 0.05s to 0.1s. Such a setting is useful for video streaming such as MPEG with video bitrate from 6Kbps to 4Mbps, as in [19]. This means that the client’s buffer only needs to store up to 0.4Mbits of data in order to compensate for the delay jitter effect of the multimedia packets.

The results in Figure 6.18 show that the proposed modified JD scheme for layered multicast achieves a relatively lower received throughput of multimedia UDP traffic than that obtained when using RED. Figure 6.20 shows that the quality of the received layered multimedia multicast traffic when using RED is higher than that obtained through the use of the proposed modified JD. However, if the “useful” received multimedia packets are considered, there are more “useless” packets when RED is used than obtained using in the proposed modified
JD. The important results in Figure 6.20 show that the proposed modified JD for layered multicast can achieve a better quality than that obtained using RED within a given useful threshold.
Chapter 7

Security and Privacy for Multimedia Streaming

Multimedia streaming usually involves multiple clients communications. IP multicast is an efficient communication mechanism for group-oriented applications such as video conferencing, Internet group game and video-on-demand system, etc. IP multicast saves bandwidth by sending the source traffic on a multicast tree that spans all the members of the group. However, IP multicast does not provide any provisions to restrict the delivery of data to a specified set of clients, therefore it is not a secure network protocol. To achieve secure delivery, the server encrypts the messages with a SEK previously distributed to all the valid group members, so that the members can decrypt the messages accordingly. When there are changes in group membership (members joining or leaving), the GC distributes a new set of SEK to those group members affected by the membership changes in order to protect the security of past, present and future communications. This process is known as re-keying. The confidentiality requirements can be translated into two
major key distribution rules [31]:

1. Forward confidentiality: users that have left the group do not have access to any future key. This ensures that a member cannot decrypt data broadcasted in the multicast session after the member left the multimedia session.

2. Backward confidentiality: a new user joining the group does not have access to any old keys. This ensures that a newly joined member cannot decrypt data broadcasted in the multicast session before the member joined the multimedia session.

These two key distribution rules not only require the re-keying process be done securely, it also has to be done efficiently, so that it does not increase the member joining and leaving latencies caused by the transmission of control messages and new keys. The re-keying process updates the SEK by encrypting the new SEK using a set of keys known as the KEK and distributes the encrypted SEK. Therefore, a key management scheme is required to manage and securely distribute the KEK to valid group members.

The key management scheme needs to ensure that only valid group members have access to the messages encrypted with the new SEK. Newly joined group members should not be able to access messages encrypted with previous SEKs, and members who have left the multicast group should not be able to access multicast messages after they leave.
The advantages and disadvantages of existing key management schemes, including the simple key management scheme, LKH [28], and Iolus [30] have been discussed in Section 2.5. In this section, we discuss the proposed MRT, which is based on the LKH approach. The MRT is optimal in minimizing the average number of keys \( \tilde{l} \) needed to be updated for each member, and the average tree height \( \tilde{h} \) for each member. Both parameters are linearly related to the average update communication overhead \( \tilde{O} \) of the re-keying process.

### 7.1 Re-keying Cost

The performance of a particular key management scheme is characterized by two parameters; the average number of key \( \tilde{l} \) needed to be updated for each member, and the average tree height \( \tilde{h} \) for each member. The probability of leaving group \( P_i \) is applied in [32] to measure how frequently the member \( M_i \) wants to leave the group. Since a new set of keys (\( l_i \) keys) is assigned to the member \( M_i \) for every time when he joined the group, therefore, the average number of key \( \tilde{l} \) needed to be updated for each member can be computed as

\[
\tilde{l} = \sum_{i=0}^{N} P_i l_i,
\]

where \( N \) is the multicast group size.

It is shown in [32] that \( \tilde{l} \) is bounded by

\[
H_l \leq \tilde{l} < H_l + 1,
\]
where

\[ H_t = - \sum_{i=0}^{N} P_i \log_2 P_i, \quad (7.3) \]

for a key tree with \( d \) branches per node. It is also stated in [32] that a member with probability of leaving group \( P_i \) should be assigned with

\[ l_t^* = - \lfloor \log_2 P_i \rfloor \]

(7.4) keys, excluding the root key and the SEK. In that case, \( \bar{l} \) achieves its minimum value for the given key tree. In fact, \( \bar{l} \) is linearly related to the average update communication overhead \( \bar{O} \). After the generation of a key tree, the average update communication overhead \( \bar{O} \) is linearly related to the average tree height \( \bar{h} \) of each member. The probability of leaving group allows the computation of \( \bar{h} \) as

\[ \bar{h} = \sum_{i=0}^{N} P_i h_i, \quad (7.5) \]

where \( h_i \) is the tree height for member \( M_i \) related to the root node.

The analysis provided in [32] allows the analytical evaluation of the performance of a given key tree and key management scheme. However, [32] does not provide any key tree generation algorithm that generates an optimal key tree with minimal \( \bar{l} \) and \( \bar{h} \), nor an efficient re-keying algorithm that maintains the key tree with minimal \( \bar{l} \) and \( \bar{h} \).

A key tree generation algorithm will be presented in Section 7.2.1 that generates the proposed MRT for key distribution with a given probability of leaving group \( P_i \) for each member \( M_i \). The MRT is optimal in minimizing the re-keying costs in that it keeps the minimal average number of keys \( \bar{l} \) needed to be updated for each
member and the average tree height $\bar{h}$ for each member. Members who leave more frequently (have a higher probability of leaving group $P_i$), should have a smaller number of keys $l_i$ assigned to them than that assigned to members who stay longer in the group. Also, those members who stay longest in the group have the same, or the smallest number of keys assigned. Furthermore, the tree update procedure maintains the optimality of the key tree after re-keying. Section 7.2.4 discussed the computation of the key tree update interval under certain cost constraint.

When the MRT was combined with subgrouping, multiple MRTs is proposed in Section 7.3. The multiple MRTs further minimizes the member storage, GC storage $GCS$ and average update communication overhead $\bar{O}$ compared to that obtained using other key management schemes.
7.2 Minimum Redundancy Tree (MRT)

In the previous section, we showed that the re-keying cost could be analyzed in the same way as the entropy of a symbol in a communication system. This section will discuss the employment of the minimum entropy tree building technique in the generation problem of the proposed MRT. A MRT generation algorithm similar to the prefix code generation method is presented in Section 7.2.1. Such a key tree will achieve the update communication overhead at $O(log_d N)$. The details of the proposed key tree generation scheme is best described by the example in Section 7.2.1.
7.2.1 Key Tree Generation

To simplify the discussion of tree generation, we consider the binary tree case in the following but the same discussion can be applied to the general $d$-ary MRT. Consider a binary key tree with seven members $M_i = (M_1, M_2, M_3, M_4, M_5, M_6, M_7)$ in a multicast group, and with the probability of leaving group $P_i = (0.17, 0.26, 0.2, 0.05, 0.07, 0.23, 0.02)$, respectively. The binary key tree generated by LKH [28, 29] scheme is shown in Figure 7.1. The number of keys assigned to each member and the tree height for each member are listed in Table 7.1. The minimum average number of keys $\bar{l}$ needed to be updated for each member when there is membership change and the average tree height $\bar{h}$ for each member are both equal to 3.

The binary key tree generated by the proposed MRT is shown in Figure 7.2. The key tree is obtained by first combining the two members with the smallest probability of leaving group together into a single node. The newly formed node will be assigned with a probability of leaving group equals to the sum of that of all its child. Therefore, $M_4$ and $M_7$ are combined to form a single node and a probability of leaving group equals to 0.07 is assigned to the new node. Proceeding in this way, the two members or member groups with the least probability of leaving group are combined into one node until all the members have joined the key tree. Then KEKs are assigned to the members in the key tree.

It can be observed that those members who leave more frequently (have a higher probability of leaving group $P_i$) will have a smaller number of keys $l_i$ assigned to them than those members who stay longer in the group. Also, the two members
Table 7.1: The Key Assignment for each Member $M_i$

<table>
<thead>
<tr>
<th>$M_i$</th>
<th>$P_i$</th>
<th>Key Assigned* by LKH</th>
<th>Key Assigned* by MRT</th>
<th>$l_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_1$</td>
<td>0.17</td>
<td>${K_{11}, K_{21}, K_{31}}$</td>
<td>${K_{11}, K_{21}, K_{31}}$</td>
<td>3</td>
</tr>
<tr>
<td>$M_2$</td>
<td>0.26</td>
<td>${K_{11}, K_{21}, K_{32}}$</td>
<td>${K_{11}, K_{22}}$</td>
<td>2</td>
</tr>
<tr>
<td>$M_3$</td>
<td>0.2</td>
<td>${K_{11}, K_{22}, K_{33}}$</td>
<td>${K_{12}, K_{24}}$</td>
<td>2</td>
</tr>
<tr>
<td>$M_4$</td>
<td>0.05</td>
<td>${K_{11}, K_{22}, K_{34}}$</td>
<td>${K_{11}, K_{21}, K_{32}, K_{42}, K_{51}}$</td>
<td>5</td>
</tr>
<tr>
<td>$M_5$</td>
<td>0.07</td>
<td>${K_{12}, K_{23}, K_{35}}$</td>
<td>${K_{11}, K_{21}, K_{32}, K_{41}}$</td>
<td>4</td>
</tr>
<tr>
<td>$M_6$</td>
<td>0.23</td>
<td>${K_{12}, K_{23}, K_{36}}$</td>
<td>${K_{12}, K_{23}}$</td>
<td>2</td>
</tr>
<tr>
<td>$M_7$</td>
<td>0.02</td>
<td>${K_{12}, K_{24}, K_{37}}$</td>
<td>${K_{11}, K_{21}, K_{32}, K_{42}, K_{52}}$</td>
<td>5</td>
</tr>
</tbody>
</table>

* Excluding the Root Key $K_0$ and the SEK

($M_4,M_7$) who stay the longest in the group will have the same number of keys $l_i$ assigned. The average number of keys $\bar{l}$ needed to be updated for each member can be computed by eq.(7.1),

$$\bar{l} = \sum_{i=0}^{N} P_i l_i = 2.52 \text{ keys/member},$$

(7.6)

which is smaller than that in LKH. Similarly, $\bar{h}$ can be computed by eq.(7.5) and is shown to be the same as $\bar{l}$ for the MRT. The information theory told us that the $l$ and $\bar{h}$ obtained by the MRT generation method are the minimal, and are close to the lower bounded given by eq.(7.2) and eq.(7.3), where

$$H_l = - \sum_{i=0}^{N} P_i \log_2 P_i = 2.4896 \text{ keys/member}.$$  (7.7)

### 7.2.2 Membership Changes

The initial MRT generated after the multicast session begun will be corrupted by members joining or leaving the session. In each member join or leave situation, the key management system assigns new keys to all new members and all other affected members. Although it is unlikely that more than one member will join
or leave the multicast session at the same time, it is common to consider a simultaneous joining/leaving due to the nature of packet communication. A similar idea of simultaneous joining/leaving was considered in [33]. Without introducing an additional join latency, the key management scheme processes the member leaving request before processing the member joining request. To simplify the discussion, we discuss the membership changes through examples in the binary tree case. The same discussion can be applied to the general \(d\)-ary MRT. A tree update procedure for member joining or leaving the MRT will be discussed, such that it can minimize the update communication overhead, member joining and leaving latencies and cost (money).

Within an key tree update interval, let \(N_i\) be the number of joining requests, \(N_o\) be the number of leaving requests, and \(N_e\) be the number of empty branches in the tree. The empty branches are the tree branches that have no member inserted. When \(N_i = N_o + N_e\), members that request to join the group are inserted into empty branches and into the branches from where members have left. Furthermore, new members are inserted hierarchically accordingly to the probability of leaving group \(P_i\). As discussed in Section 7.1, the optimal number of keys assigned to each member (excluding the root key and the SEK) should equal \(l_i^* = \lfloor \log_d P_i \rfloor\). Noticed that \(l_i\) is linearly related to the tree height \(h_i\) of the nodes to be inserted. Without loss of generality, assume the probability of leaving group is arranged as

\[
P_N \geq P_{N-1} \geq \cdots \geq P_i \geq \cdots \geq P_2 \geq P_1.
\]

(7.8)

The member \(M_i\) is inserted in the empty branches from the smallest tree heights
Figure 7.3: The Simultaneous Join/Leave Group, Case 1: \( N_i = N_o + N_e \)

to the largest tree height, and from left to right, as shown in Figure 7.3.

When \( N_i < N_o + N_e \), there are empty branches after the member joining and member leaving process. As a result, the key management scheme has to decide which empty branches would remain empty. The tree height of member \( h_i \) is linearly related to the average update communication overhead \( \bar{O} \). In order to minimize \( \bar{O} \) and the join latency, those branches with the largest tree height to be empty are left empty. Figure 7.4 shows one example. The insertion of new members is done from the smallest tree height to the largest tree height, and from left to right, as discussed.

Figure 7.5a shows an example for the case of \( N_i > N_o + N_e \) in which there are not enough empty branches for all the member joining requests. As a result, node splitting is performed to create enough empty nodes to accommodate all the new
Figure 7.4: The Simultaneous Join/Leave Group, Case 2: $N_t < N_o + N_e$

member joining requests. In order to minimize the average update communication overhead $O$, only the empty nodes are split. Furthermore, if there are a lot of requests, after all the empty nodes in the original key tree have been split, the newly generated nodes will then be split. Such a hierarchical splitting will be performed recursively until all the new member join requests are accommodated. In particular, the splitting follows the hierarchy of $P_i$ so that the node with the least probability value, or the largest tree height, are split into two child nodes for $M_j$ to join, as shown in Figure 7.5b. By doing this, the average update communication overhead $O$ and the join latency is minimized as the average tree height $h$ for each member is linearly related to $O$. The above join and leave procedure is logical. The physical key update messages are sent only after the
logical update is finished. Thus, the order of insertion and node splitting does not affect the total number of update communications.

7.2.3 Key Tree Update

The aim of the membership management strategies presented in the previous section is to minimize the average update communication overhead $\bar{O}$ so that the member joining and leaving latencies are the smallest. However, after the joining and leaving process, it is very likely that the $\bar{l}$ and $\bar{h}$ of the updated key tree will not be the same as that in the original MRT. As a result, in the next joining and leaving process, the average update communication overhead $\bar{O}$ is larger than that of the optimal MRT case, and so too are the associated member joining and leaving latencies. As more and more members join and leave the system, the key tree becomes more and more dissimilar to the MRT case and, thus, a larger and
larger average update communication overhead \( \bar{O} \) will be resulted in all future member joining and leaving processes. Such an effect, called the "re-keying cost accumulation effect" can be eliminated by updating the key tree to MRT. As a result, after the key tree update process, a new MRT is formed, and this achieves the optimal \( \bar{l}, \bar{h} \) and the minimal average member joining and leaving latencies.

A similar idea was presented in Kronos [35] in the form of a periodic group re-keying scheme. Instead of performing the re-keying procedure on each membership change, Kronos suggests to perform a batch re-keying process for all the members in a fixed period. The re-keying costs, the average data latency and network overhead are reduced by the batch re-keying, but the joining and leaving latencies are increased. The batch re-keying process is costly because of the large amount of control messages involved. Therefore, the key tree update interval \( \Delta t \) should be chosen appropriately. Also note that not only the re-keying cost accumulation effects depends on \( \Delta t \). The key tree update communication overhead depends on how much in violation the current key tree is compared to that of the associated MRT. This, in turn, depends on \( \Delta t \). Furthermore, the choice of \( \Delta t \) also affects the pricing scheme of the multimedia session. For example, in an Internet group game, if \( \Delta t \) is too large, the current users may lose interests and logout, that results in losing profits of the game providers. On the other hand, if \( \Delta t \) is too small, the key tree update is so frequent, that makes the users to pay more money and it is unfair to them. As a result, \( \Delta t \) must be carefully chosen.

In this thesis, we propose to update the key tree to form a MRT for every key tree update interval \( \Delta t \). It is natural to constrain the amount of key tree update
communication overhead $O$, so as to avoid network congestion. Also, it can limit the member joining and leaving latencies and constrain the cost (money) in each batch re-keying process.

7.2.4 Update Communication Overhead for MRT

Figure 7.6 shows the member leaving events for the member $M_i$ within the key tree update period $[0, \Delta t]$. It is assumed that the new members can only join the group at a particular time instant $\Delta t$. The smaller the $\Delta t$, the smaller the member joining latency. However, the smaller the $\Delta t$, the more frequent the key tree update and the more costly to maintain the MRT. Since the average cost to operate the multicast session for each member per unit time is constrained by the pricing scheme of the multicast session, for example, the multicast group controller might want the average cost to operate the multicast session for each member per unit time, $\bar{C}$, to be equal to a percentage of the subscription fee for each member per unit time, $F_s$ in the multicast session, i.e.

$$\bar{C} = \alpha_c,$$

(7.9)

Figure 7.6: Member Joining and Leaving Events within each Key Tree Update Interval

149
where $\alpha_c$ is the cost constraint and is linearly related to the subscription fee for each member per unit time, $F_r$. Such that the multimedia service providers can make profits from providing the multicast service. As a case in an Internet game, each member needs to pay the subscription fee before joining the game session. And the multicast group controller might need that money to compensate for the cost in maintaining and operating the game session. As a result, the frequency of the MRT update is limited. We can then compute the key tree update interval, $\Delta t$ under such a given pricing scheme in the multicast session.

Before we compute the average cost to operate the multicast session for each member per unit time, $\bar{C}$, we need to compute the total average update communication overhead for each member per unit time, $\bar{O}_{total,t}$, first. Let $T_i$ be the expected session time for the member $M_i$. Assume that the probability for member $M_i$ will leave the group within the key tree update period $[0, \Delta t]$, $Q_i$, is given by

$$Q_i = \frac{\Delta t}{T_i},$$

(7.10)

where the key tree update interval $\Delta t$ should be greatly smaller than the minimum value of $T_i$, i.e. $\Delta t << \min\{T_i\}$ and $\sum_{i=1}^{N} \frac{Q_i}{\sum_{j=1}^{N} Q_j} = 1$. Therefore, the expected total update communication overhead for the member leaving events, $\bar{O}_t$, for the multicast group with $N$ members within the key tree update period $[0, \Delta t]$ is given by

$$\bar{O}_t = \sum_{i=1}^{N} Q_i t_i,$$

(7.11)

$$= \sum_{i=1}^{N} \frac{\Delta t}{T_i} t_i,$$

(7.12)
\[ = \Delta t \sum_{i=1}^{N} \frac{l_i}{T_i}. \quad (7.13) \]

It is assumed that the new members can only join the group at a particular time instant \( \Delta t \). The new members will be linked to form a new MRT and new “Key Encryption Keys” (KEKs) will be assigned and distributed to each member \( M_i \) by the multicast group controller. For simplicity, we use a binary tree case for the MRT reconstruction discussion. As discussed before, the new MRT is obtained by first combining the two members with the smallest probability for the member will leave the group within the key tree update period, together into a single node. The newly formed node will be assigned with a new probability for the member will leave the group within the key tree update period, equals to the sum of that of all its child. Proceeding in this way, the two members or member groups with the least probability for the member will leave the group within the key tree update period are combined into one node until all the members have joined the key tree. Finally, the new KEKs are assigned and distributed to each member \( M_i \) by the multicast group controller.

In the MRT, those members who leave more frequently will have a smaller number of keys \( l_i' \) assigned and distributed to them than those members who stay longer in the group. The average number of keys \( \bar{\nu} \) that needs to be assigned and distributed to each member at particular time instant \( \Delta t \) can be computed by

\[ \bar{\nu} = \frac{1}{N'} \sum_{i=1}^{N'} l_i', \quad (7.14) \]

where \( N' \) is the number of members in the group after all new members joined in
the group and \( l'_i \) is the number of keys that needs to be assigned and distributed to each member \( M_i \) by the group controller. As the computation and processing overhead for the MRT reconstruction is greatly smaller than the key distribution overhead to each member by the multicast group member, therefore, we can assume only the key distribution overhead contributes to the MRT reconstruction overhead. Therefore, the MRT reconstruction overhead for each member equals the number of keys that needs to be assigned and distributed to each member, \( l'_i \) by the multicast group controller. Therefore, The average MRT reconstruction overhead with new members joined in the group, \( \bar{O}_r \), for each member at a particular time instant \( \Delta t \) equals the average number of keys \( \bar{p} \) that needs to be assigned and distributed to each member by the multicast group controller.

\[
\bar{O}_r = \bar{p},
\]

\[
= \frac{1}{N'} \sum_{i=1}^{N'} l'_i.
\]

(7.15)

(7.16)

We can also assume the number of member leaving events within the key tree update period \([0, \Delta t]\) is approximately equal to the number of new member joining events at a particular time instant \( \Delta t \) in order to minimize the member joining latency, therefore, for simplicity, we let \( N' = N \). The average MRT reconstruction overhead with new members joined in the group, \( O_r \), for each member at a particular time instant \( \Delta t \) is given by

\[
O_r = \frac{1}{N} \sum_{i=1}^{N} l'_i.
\]

(7.17)

Therefore, by eq.(7.13) and eq.(7.17), the total expected update communication overhead, \( \bar{O}_{total, \Delta t} \), for each member for each key tree update interval \( \Delta t \) equals
to the sum of the expected total update communication overhead for the member leaving events, $\bar{O}_l$, within the key tree update period $[0, \Delta t]$ and the average MRT reconstruction overhead with new members joined in the group, $\bar{O}_r$, for each member at a particular time instant $\Delta t$. Hence,

$$\bar{O}_{total, \Delta t} = \bar{O}_l + \bar{O}_r,$$

$$\bar{O}_{total, \Delta t} = \sum_{i=1}^{N} Q_i l_i + \frac{1}{N} \sum_{i=1}^{N} l'_i,$$

$$\bar{O}_{total, \Delta t} = \Delta t \sum_{i=1}^{N} \frac{l_i}{T_i} + \frac{1}{N} \sum_{i=1}^{N} l'_i.$$

Therefore, the total expected update communication overhead, $\bar{O}_{total,t}$, for each member per unit time is given by

$$\bar{O}_{total,t} = \frac{\bar{O}_{total, \Delta t}}{\Delta t},$$

$$\bar{O}_{total,t} = \sum_{i=1}^{N} \frac{l_i}{T_i} + \frac{1}{N \Delta t} \sum_{i=1}^{N} l'_i.$$

As discussed before, the total expected update communication overhead for each member per unit time, $\bar{O}_{total,t}$, is linearly related to the average cost to operate the multicast session for each member per unit time, $\bar{C}$, i.e. $\bar{C} = a_o \bar{O}_{total,t}$, where $a_o$ is a constant.

We also want to keep the average cost to operate the multicast session for each member per unit time, $\bar{C}$, to be equal to a cost constraint $\alpha_c$, which is predefined and determined by the multicast group controller. As discussed, $\alpha_c$ is linearly related to the subscription fee $F_s$ for each member per unit time to join the multicast session, i.e. $\alpha_c = a_f F_s$, where $a_f$ is a constant and is predefined by the pricing scheme of the multicast session. Therefore, the key tree update interval,
\( \Delta t \), under a given pricing scheme can be computed by letting

\[
\bar{C} = \alpha_c, \quad (7.23)
\]

\[
a_o \bar{O}_{\text{total,t}} = a_f F_s, \quad (7.24)
\]

\[
a_o (\sum_{i=1}^{N} l_i \! / \! T_i + \frac{1}{N \Delta t} \sum_{i=1}^{N} l_i') = a_f F_s, \quad (7.25)
\]

\[
\Rightarrow \Delta t = \frac{a_o \sum_{i=1}^{N} l_i}{a_f F_s - a_o \sum_{i=1}^{N} \frac{l_i}{T_i}} \quad (7.26)
\]

As a result, the key tree update has to be performed within the time interval \( \Delta t \) as given in eq.(7.26). Otherwise, the average cost to operate the multicast session for each member per unit time and to maintain the MRT will exceed the given cost constraint, and affect the profits of the multimedia service providers.

### 7.3 Multiple Minimum Redundancy Tree (Multiple MRTs)

The MRT minimizes the average number of keys \( \bar{l} \) needed to be updated for each member and the average update communication overhead \( \bar{O} \). Further improvement can be obtained by dividing all the multicast group members into different independent subgroups [30]. By doing so, the amount of the update communication can be reduced. Because the membership changes within each subgroup may not affect other subgroups. In the case, only the SEK of the subgroup of concern needs to be changed.

We further propose to combine the optimal key tree generation scheme by MRT with the subgrouping key management scheme. The multicast session with
Figure 7.7: The proposed Multiple MRTs Approach

$N$ users is therefore divided into $G$ subgroups with subgroup size $M$ according to the probability of leaving group $P_i$ of individual member. The key tree within each subgroup is generated using the same method as that used in MRT, while each subgroup is linked to form a MRT structure. It is shown that $\bar{l}$, $\bar{h}$ and the associated communication overhead, and also the member joining and leaving latencies are reduced under this structure.

The example of a binary key tree using multiple MRTs is shown in Figure 7.7. The group controller GC first sorted the $P_i$ values in the descending order. Without loss of generality, assume

$$P_N \geq P_{N-1} \geq P_{N-2} \geq \cdots \geq P_2 \geq P_1.$$  \hspace{1cm} (7.27)
Suppose the subgroup size is \( M \) for all the subgroups, and that the probabilities of member leaving subgroup \( S_i \) are given by

\[
P_{M}^{S_i} \geq P_{M-1}^{S_i} \geq P_{M-2}^{S_i} \geq \cdots \geq P_2^{S_i} \geq P_1^{S_i}.
\]  

(7.28)

The MRT is generated for each subgroup \( S_i \) according to the probability of leaving hierarchy in eq.(7.14). Furthermore, the MRT subgroup key tree is formed according to the hierarchy of the sum of probability of leaving subgroup for all members in each subgroup, \( P_{\text{sum}}^{S_i} \) as follows:

\[
P_{\text{sum}}^{S_G} \geq P_{\text{sum}}^{S_{G-1}} \geq \cdots \geq P_{\text{sum}}^{S_i} \geq \cdots \geq P_{\text{sum}}^{S_2} \geq P_{\text{sum}}^{S_1}.
\]  

(7.29)

where number of subgroups, \( G = N/M \) and \( P_{\text{sum}}^{S_i} = P_{M}^{S_i} + P_{M-1}^{S_i} + \cdots + P_2^{S_i} + P_1^{S_i} \).

The key tree generated using the multiple MRTs approach has \( \tilde{L} \) equals to \( (2 + \sum_{i=1}^{M} P_i l_i) \), which is \( O(\log_2 M) \). In the multiple MRTs approach, only the subgroup members need to perform re-keying when there is a membership change; members in other subgroups are not affected. The minimum average update communication overhead \( \tilde{O}_{\text{min}} \) is defined as

\[
\tilde{O}_{\text{min}} = \sum_{i=1}^{M} P_i l_i
\]  

(7.30)

\[
< \log_2 M + 1
\]  

\[\Rightarrow O(\log_2 M). \]  

(7.31)

The group controller GC needs to store

\[
GCS = \sum_{i=0}^{\log_2(N/M)} d^i + \frac{N}{M}
\]  

(7.32)

156
\[
\begin{align*}
\frac{d\log_d N + 1}{d - 1} + \frac{N}{M} &= (1 + \frac{d}{d - 1}) \frac{N}{M} - \frac{1}{d - 1}, \\
&\Rightarrow O(N/M),
\end{align*}
\]

keys for the whole multicast session. This minimum update situation holds only if the order of the sum of probability of leaving group \(P_{sum}^{Si}\) for subgroup \(S_i\) does not violate the MRT conditions after the change in membership. However, consideration is needed for the case in which the order of the sum of probability of leaving group \(P_{sum}^{Si}\) for subgroup \(S_i\) is changed and violates the MRT conditions after the membership changes. All the tree branches related to that multicast group will need to be updated. This leads to an average update communication overhead \(\bar{O}\)

\[
\bar{O} = \sum_{i=1}^{N} P_i l_i
\]

\[
< \log_d N + 1
\]

\[
\Rightarrow O(\log_d N).
\]

The update procedure follows that of the MRT generation procedure with respect to the values of the sum of probability of leaving group \(P_{sum}^{Si}\).

Table 7.2 shows the worst case analysis comparison in regard to the member storage, the GC storage \(GCS\), and the update communication overhead \(O\) for different key management schemes. It is shown that under the worst case situation, the member storage, GC storage and update communication overhead of the LKH and the MRT are the same. When multiple MRTs approach is applied, the member storage is reduced to \(O(\log_d M)\) under the minimum update situation. This is better than \(O(\log_d N)\) in the LKH scheme. Moreover, the GC storage
$GCS$ is reduced to $O(N/M)$ on average. This is better than $O(N)$ in the LKH scheme.

### 7.3.1 Membership Changes

The key management scheme for multiple MRTs is a modification of that used in the MRT. When there are members joining the group, each member is classified into an individual subgroup according to the probability of leaving group $P_i$. Within the subgroup, the joining and leaving procedures are the same as those discussed in Section 7.2.2, and the tree update is performed after all the joining and leaving processes have completed.

In Table 7.3 a comparison of the member storage, GC storage and average update communication overhead between the LKH and the multiple MRTs schemes with different parameters $N$ and $M$ are presented. The proposed multiple MRTs can maintain a lower member storage and GC storage than those in the LKH scheme. Although the average update communication overhead $\bar{O}$ for the multiple MRTs is the same as that in the LKH scheme, the minimum amount of average update communication overhead $\bar{O}_{min}$ can be achieved by the multiple MRTs if the minimum update situation is hold. Therefore, the multiple MRTs is better than the LKH scheme in reducing the average update communication overhead $\bar{O}$.
Table 7.2: A WorstCase Analysis of Different Key Management Schemes

<table>
<thead>
<tr>
<th>Key Management Schemes</th>
<th>Member Storage</th>
<th>GC Storage</th>
<th>Update Communication Overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple</td>
<td>$O(1)$</td>
<td>$O(1)$</td>
<td>$O(N)$</td>
</tr>
<tr>
<td>LKH</td>
<td>$O(\log d N)$</td>
<td>$O(N)$</td>
<td>$O(\log d N)$</td>
</tr>
<tr>
<td>Iolus</td>
<td>$O(1)$</td>
<td>$O(N/M)$</td>
<td>$O(M)$</td>
</tr>
<tr>
<td>MRT</td>
<td>$O(\log d N)$</td>
<td>$O(N)$</td>
<td>$O(\log d N)$</td>
</tr>
<tr>
<td>Multiple MRTs</td>
<td>$O(\log d M)$</td>
<td>$O(N/M)$</td>
<td>$O = O(\log d N)$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$O_{\text{min}} = O(\log d M)$</td>
</tr>
</tbody>
</table>

Table 7.3: A Comparison of Different Costs of the LKH and Multiple MRTs Approaches.

<table>
<thead>
<tr>
<th></th>
<th>N</th>
<th>M</th>
<th>d</th>
<th>Member Storage</th>
<th>GC Storage</th>
<th>$\bar{O}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>LKH</td>
<td>1024</td>
<td>-</td>
<td>2</td>
<td>12</td>
<td>2047</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>1024</td>
<td>-</td>
<td>3</td>
<td>8</td>
<td>1536</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>1024</td>
<td>-</td>
<td>15</td>
<td>1572864</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1024</td>
<td>-</td>
<td>4</td>
<td>7</td>
<td>1365</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>1024</td>
<td>-</td>
<td>12</td>
<td>1398101</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Multiple MRTs</td>
<td>1024</td>
<td>10</td>
<td>2</td>
<td>6</td>
<td>306</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>1024</td>
<td>10</td>
<td>3</td>
<td>5</td>
<td>256</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>1024</td>
<td>20</td>
<td>3</td>
<td>6</td>
<td>131072</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>1024</td>
<td>10</td>
<td>5</td>
<td>239</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>1024</td>
<td>20</td>
<td>4</td>
<td>5</td>
<td>122334</td>
<td>10</td>
</tr>
</tbody>
</table>
Chapter 8

Conclusions and Future Work

8.1 Conclusions

The design of the end-to-end architecture of multimedia streaming over the Internet that provides real-time playback has been discussed. The multimedia bitrate adaptation flow control system which maintains the client’s buffer at, or near, a predefined capacity even during bursty loss periods, was proposed. The multimedia bitrate adaptation mechanism provides an overall playback quality improvement by adjusting the multimedia bitrate of the streaming encoder. It can maintain a high buffer fill-up rate and prevent multimedia dropout problems. The loss packet recovery mechanism provides error recovery capability to the system. The simulation results showed that a high successful packet transmission rate is achieved when the proposed mechanism is applied. Moreover, a nonuniform packet arrival mechanism capable of dealing with the out-of-sequence packet arrival problems was proposed. Such a protocol can also be used to initiate the multimedia bitrate adaptation, as described earlier, by monitoring the
multimedia dropout rate of the streaming system.

The client-based congestion control mechanism was also proposed which works with the proposed multimedia bitrate adaptation flow control mechanism for resolving network congestion problems. In case of network congestion, the client requests the server to adjust the sending rate $R$ while maintaining a certain degree of TCP-friendliness compared to that of the TFRC scheme used in TCP-friendly congestion control for multimedia traffic. The proposed client-based congestion control mechanism also integrates the use of a simple weight factor scheme to better allocate resources among multiple streaming traffic.

The “Jitter Detection” (JD) for gateway-assisted congestion control was also proposed. The JD scheme improves the QoS of multimedia network by detecting and discarding multimedia packets that have violated the delay jitter tolerance measure. We proposed the classification of the TCP and UDP traffic by looking into the packet header of the received packet, so that the RED and JD buffer management schemes are applied to individual traffic respectively, before sending those traffic to the output FIFO queue. The JD scheme was investigated by incorporating knowledge of residual distance into the jitter detection, and packet discarding algorithm. Simulation results showed that the proposed JD scheme can maintain the same TCP-friendliness as that of RED and DropTail. At the same time, the JD scheme reduces the average delay jitter of the multimedia packets and maintains a high useful throughput for the multimedia traffic compared to those using other traditional gateway-assisted congestion control schemes like RED and DropTail.
The modified JD to stream layered multimedia multicast traffic over the Internet, in order to preserve the base layer of layered video traffic with the best-effort when passing through the gateway, was also presented. This is achieved by modifying the original JD scheme to identify the packets for each level in layered multicast traffic. The dropping priority is given to multicast traffic which correspond to the higher level layers (enhancement layers). The performance of the modified JD scheme had been shown through the use of a queuing analysis. The proposed modified JD scheme can lower both the average number of packets and the average waiting time in the queue compared to that in which no gateway-assisted congestion control is employed. Simulation results showed that the modified JD scheme can effectively provide better quality improvement of the received layered multimedia multicast traffic and achieve similar TCP-friendliness compared to the use of RED for gateway-assisted congestion control.

Finally, the "Minimum Redundancy Tree" (MRT), used for key distribution with the given probability of leaving group for each member, was presented in this dissertation. The MRT is optimal in terms of minimum re-keying costs as it keeps the minimum average number of keys assigned to each member and maintains the minimum average tree height for each member. The tree update procedure maintains the optimality of the key tree after re-keying. An analytical analysis of the proposed algorithms was presented. The key tree update interval under constrained network resources was also computed. By combining MRT with subgrouping, the multiple MRTs was proposed, such that member storage, "Group Controller" (GC) storage and update communication overhead can be
further minimized compared to other key management schemes.

8.2 Future Work

We proposed a robust streaming protocol with multimedia bitrate adaptation flow control, congestion control and loss recovery mechanisms which provide multimedia streaming over the Internet. However, the proposed streaming protocol is only suitable for a wired network such as the Internet, and not for wireless networks. The multimedia streaming over the wireless networks faces several challenges including error resilience, and the need to deal with bandwidth variations [37]. Wireless channel bandwidth can vary significantly depending on the signal strength and interference level that a user receives. As a user travels through different parts of the cell, different bandwidth may be dynamically assigned to that user. In addition, depending on the QoS capability of the wireless network, multi-user sharing of the wireless network with heterogeneous data types also leads to significant user channel bandwidth variation. This bandwidth variation further leads to buffer overflow and packet loss. This affects the quality of multimedia streams. Finally, data transmission will be interrupted completely due to cell selection and handoff process, resulting in transmission gaps ranging from a fraction of a second to several seconds. This unpredictability of available wireless channel bandwidth introduces high delay jitter for the multimedia streaming packets.

To alleviate these problems, we will develop a scalable, efficient and robust streaming protocol with provisions for multimedia streaming applications over
a heterogeneous network composed of both wired and wireless connections with different types of terminated devices. It will provide error resilience, robustness, and scalability. We will develop several robust transport mechanisms and system formats for the delivery of multimedia contents over wireless networks. We will focus on multimedia information packetization, network characteristics estimation and monitoring, adaptive error recovery, adaptive bandwidth allocation, adaptive power control, and multimedia synchronization.

We will also explore into new areas such as multimedia server design and multimedia communications in the DiffServ network. A robust and efficient admission control algorithm will be developed, together with the multimedia server that supports the QoS. The admission control algorithm will determine whether or not a new request can be admitted to the server without compromising the required performance of the in-service users. Network resources will be assigned to each request and will be dynamically allocated and released in response to users changing demands throughout the multimedia session. Admission control procedures for prioritizing users' requests for stream delivery and deciding which resources to allocate will be considered. Since existing admission control schemes only base their decisions on the available network resources estimated by the server [38, 39, 40]. They do not provide any support for pricing policy. This leads to unfair resource allocation. In order to solve this problem, we will augment the above admission control schemes so that the decisions are based on available network resources, the priority of users, and pricing policy. These features are in great demand from service providers looking to offer customized streaming with
impeccable quality to their customers.

The Diffserv network is a new prototype network for multimedia communications via which network service providers can offer different levels of network QoS to different customers and their traffic streams. In the Diffserv network, packets of the same QoS specification are grouped together and forwarded in the same manner, e.g. to a given subnet, or set of subnets, with some level of service provisioning. However, the Diffserv network does not provide a full QoS guarantee to every application flow, especially for multimedia applications. We will investigate an adaptive packet forwarding mechanism which uses priority dropping for the Diffserv network so as to provide service differentiation. Multimedia packets will be mapped onto different Diffserv levels based on their priority levels. Another avenue for research is provided by having service differentiation between different traffic in different Diffserv levels. In addition, end-to-end multimedia quality with the same pricing constraint could also be enhanced.

Moreover, the application of the JD scheme to Diffserv will be investigated in the future. The proposed JD scheme cannot work with the Diffserv protocol due to the fact that Diffserv utilizes all of the eight bits in the TOS field in the IPv4 header, which makes it impossible to implement the delay jitter counter $v$. To remedy this problem, we propose to overlay the delay jitter counter function on top of the Diffserv levels such that amongst all the QoS levels, four of them will be used as the delay jitter counter as well. This can be done because the edge routers in the Diffserv domain have the ability to change the Diffserv QoS level for each flow, which is similar to changing $v$ in the JD algorithm. However,
further modifications should be considered in the original JD scheme to further enhance its performance and to apply the JD scheme to the DiffServ network.
Bibliography


http://www.isi.edu/nsnam/ns.

[22] S. McCanne, V. Jacobson, and M. Vetterli, “Receiver-driven Layered Multi-


Layered Multicast Data Transfer,” Proc. IEEE INFOCOM, pp.996-1003,
April 1998.

[26] T. Turletti, S. Parisis, and J. Bolot, “Experiments with a Layered Transmis-


[28] D.M. Wallner, E.J. Harder and R.C. Agee, “Key Management for Multicast:

2000.


[57] US Patent No. 20030046032, “Method to measure the perceived quality of streaming media.”

