Traffic Management for TCP/IP over Asynchronous Transfer Mode (ATM) Networks

DISSERTATION

Presented in Partial Fulfillment of the Requirements for

the Degree Doctor of Philosophy in the

Graduate School of The Ohio State University

By

Rohit Goyal, B.S., M.S.

* * * * *

The Ohio State University

1999

Dissertation Committee:

Professor Raj Jain, Adviser

Professor Wu-chi Feng

Professor Ten-Hwang Lai

Approved by

Adviser

Department of Computer and Information Science

ABSTRACT

The problem of traffic management has been widely recognized as critical to the development of an operational Internet. The goal of traffic management is to efficiently allocate network resources including buffers and bandwidth, and provide the negotiated QoS guarantees to users. This thesis studies the problem of traffic management for the Transmission Control Protocol/Internet Protocol (TCP/IP) suite over Asynchronous Transfer Mode (ATM) networks. The ATM Unspecified Bit Rate (UBR), Guaranteed Frame Rate (GFR) and Available Bit Rate (ABR) service categories are intended for data applications. The main focus of this research is to analyze and improve the performance of TCP over UBR, GFR and ABR.

This thesis proposes buffer management algorithms for the UBR and GFR services, and a feedback control algorithm for the ABR service. A buffer management scheme called Selective Drop is presented for the UBR service. An analysis of the relative performance of three TCP flavors – slow start and congestion avoidance (Vanilla), fast retransmit and recovery (Reno), and selective acknowledgments (SACK), with three buffer management policies – Tail Drop, Early Packet Discard (EPD), and Selective Drop, is then presented for LAN, WAN and satellite networks.

The results show that for LANs, TCP performance over UBR can be improved by using intelligent buffer management policies. As the propagation delay increases, the TCP policies become more significant than buffer management – SACK performs the

best while Reno performs the worst. In the presence of high priority variable bit rate traffic, guaranteed rate can be provided to the UBR service to prevent bandwidth starvation.

This research proposes the Differential Fair Buffer Allocation (DFBA) scheme for the GFR service. DFBA provides minimum rate guarantees to VCs carrying TCP traffic by allocating buffers in proportion to the guaranteed rates and probabilistically dropping packets.

Finally, the thesis proposes a virtual source virtual destination scheme for ABR that can be used to limit the buffer requirements of terrestrial networks connected to long delay satellite networks.

The experiments and analysis in this research provide valuable insight into designing traffic management solutions for broadband networks.

Traffic Management for TCP/IP over Asynchronous Transfer Mode (ATM) Networks

By

Rohit Goyal, Ph.D.

The Ohio State University, 1999

Professor Raj Jain, Adviser

The problem of traffic management has been widely recognized as critical to the development of an operational Internet. The goal of traffic management is to efficiently allocate network resources including buffers and bandwidth, and provide the negotiated QoS guarantees to users. This thesis studies the problem of traffic management for the Transmission Control Protocol/Internet Protocol (TCP/IP) suite over Asynchronous Transfer Mode (ATM) networks. The ATM Unspecified Bit Rate (UBR), Guaranteed Frame Rate (GFR) and Available Bit Rate (ABR) service categories are intended for data applications. The main focus of this research is to analyze and improve the performance of TCP over UBR, GFR and ABR.

This thesis proposes buffer management algorithms for the UBR and GFR services, and a feedback control algorithm for the ABR service. A buffer management scheme called Selective Drop is presented for the UBR service. An analysis of the relative performance of three TCP flavors – slow start and congestion avoidance (Vanilla),

fast retransmit and recovery (Reno), and selective acknowledgments (SACK), with three buffer management policies – Tail Drop, Early Packet Discard (EPD), and Selective Drop, is then presented for LAN, WAN and satellite networks.

The results show that for LANs, TCP performance over UBR can be improved by using intelligent buffer management policies. As the propagation delay increases, the TCP policies become more significant than buffer management – SACK performs the best while Reno performs the worst. In the presence of high priority variable bit rate traffic, guaranteed rate can be provided to the UBR service to prevent bandwidth starvation.

This research proposes the Differential Fair Buffer Allocation (DFBA) scheme for the GFR service. DFBA provides minimum rate guarantees to VCs carrying TCP traffic by allocating buffers in proportion to the guaranteed rates and probabilistically dropping packets.

Finally, the thesis proposes a virtual source virtual destination scheme for ABR that can be used to limit the buffer requirements of terrestrial networks connected to long delay satellite networks.

The experiments and analysis in this research provide valuable insight into designing traffic management solutions for broadband networks.

ACKNOWLEDGMENTS

As I reflect on my journey through graduate school, I realize that it has been a very special time that will have a long lasting effect on my professional and personal life. At this time, I am left with an immense feeling of accomplishment, of excitement about the future and of overwhelming gratitude towards everyone who's been with me to make this possible.

My parents have been unbelievably patient and supportive throughout what might have seemed to them, a long time in graduate school. I owe it to my dad and the old IBM 1401 at his workplace, for my interest in computer science. Both mom and dad have unselfishly supported and facilitated my decisions. I feel lucky to have them by my side.

My advisor, Professor Raj Jain has been a phenomenal source of inspiration and motivation throughout my stay at Ohio State. I have learned a lot from his insight into problem solving and his work ethic. It is through his support, guidance and constant energy that I have been able to bring this work to a completion. I thank him profoundly and I hope that some day I will make him proud.

The members of the academic community at Ohio State have been very influential in laying a strong foundation in Computer Science. I am very grateful to my committee members, Dr. Steve Lai and Dr. Wu-chi Feng who have very patiently read through my dissertation proposal and candidacy exam, and to my minor advisors,

Dr. Feng Zhao for giving me AI lessons, and Dr. Anish Arora for giving me a sense of distributed computing.

I will always cherish my stay with my colleagues and friends in the 'cong' group at *Netlab* – Shivku, Sonia, Bobby and Mukul. They have helped me in every step of my research – generating ideas, running simulations, writing utilities, writing and proofreading papers. Thanks to Sohail for ensuring that the systems were up and running when we needed them.

My graduate career and my work have been generously sponsored by the Ohio State University Presidential and University fellowships, the CIS Teaching Assistantship, the National Science Foundation, Lockheed Martin, NASA Lewis Research Center and Rome Airforce Labs. I am thankful to them for making the work possible.

I have greatly benefited from members of the networking community. Sastri Kota has been a colleague, a mentor and a friend. I am very grateful to him for his support. I am also thankful to the members of the ATM Forum for their comments and suggestions on our contributions.

Everyone in the CIS office has had an important role to play in my stay. I would like to particularly thank Elizabeth, Tom, Marty, Elley, Sandy and James for helping me work with the administrative hurdles at Ohio State.

Scott, Jill, Bryan, Kristin and their families have given me a home in Columbus. They have always been there for me as family and friends. Their friendship and support is invaluable to me.

I have met my closest friends over the last few years and I am greatly indebted to them for the fun times we have had. I thank Gautam, Rob, Dave, Sonal, Mitul, Susheela and my roommate Steve for supporting me through good and bad times.

VITA

1994	B.S. Computer Science, Denison University, Granville, Ohio
1995	M.S. Computer and Information Science, The Ohio State University, Columbus, Ohio

PUBLICATIONS

Research Publications

Rohit Goyal, Raj Jain, Shivkumar Kalyanaraman, Sonia Fahmy and Bobby Vandalore. Improving the Performance of TCP over the ATM-UBR Service. *Computer Communications*, Vol 21, No. 10, July 1998

Rohit Goyal, Raj Jain, Sastri Kota, Mukul Goyal, Sonia Fahmy and Bobby Vandalore. Traffic Management for TCP over Satellite-ATM Networks. *IEEE Communications Magazine,* March 1999

Rohit Goyal, Raj Jain, Thomas VonDeak, Mukul Goyal, Sonia Fahmy and Bobby Vandalore. Traffic Management in ATM Networks over Satellite Links. TIA/EIA Standard TR34.1

Raj Jain, Shivkumar Kalyanaraman, Rohit Goyal, Ram Vishwanathan and Sonia Fahmy. ERICA: Explicit Rate Indication for Congestion Avoidance in ATM Networks. US Patent 5805577

FIELDS OF STUDY

Major Field: Computer and Information Science

TABLE OF CONTENTS

Page

Abstract ii

Acknowledgments iv

Vita vi

List of Tables xi

List of Figures xiv

List of Results xviii

Chapters:

1. Introduction 1

    1.1 Components of Traffic Management 4
    1.2 Key Contributions of this Work 9
    1.3 Outline of this Dissertation 10

2. Background and State of the Art 13

    2.1 Transporting TCP/IP over ATM 13
    2.2 Traffic Management for TCP/IP over ATM 17
    2.3 An Overview of ATM Service Categories 19
    2.4 A Survey of TCP Congestion Control 21
        2.4.1 TCP Congestion Control Principles 22
        2.4.2 Slow Start and Congestion Avoidance: Vanilla TCP 27
        2.4.3 Fast Retransmit and Recovery: TCP Reno 32
        2.4.4 The Fast Retransmit Phase: TCP New Reno 34

2.4.5 Selective Acknowledgments: TCP SACK 37
2.4.6 Other TCP Implementations 40
2.4.7 Miscellaneous TCP Features 44
2.5 A Survey of Buffer Management 52
2.5.1 A Framework for Buffer Management 52
2.5.2 SA-ST schemes 56
2.5.3 MA-ST Schemes 58
2.5.4 MA-MT Schemes 60
2.5.5 SA-MT Schemes 62
2.5.6 Schemes that require per-VC Queuing 63
2.6 An Overview of ABR Feedback Control 64
2.7 An Overview of Satellite-ATM Networks 66
2.8 A Queuing Architecture 70

3. Problem Statement 76

3.1 TCP/IP over ATM: Problem Specification 77
3.2 Performance Metrics 80
3.3 Approach 82

4. UBR+: Improving the Performance of TCP over UBR 85

4.1 Chapter Goals 86
4.2 TCP over UBR 87
4.2.1 Simulation Model 88
4.2.2 Simulation Results 90
4.2.3 Buffer Requirements For Zero Loss 92
4.3 Early Packet Discard 95
4.4 Per-VC Accounting: Selective Drop and Fair Buffer Allocation . . . 96
4.4.1 Effect of Parameters 100
4.4.2 Simulation Model 101
4.4.3 Simulation Results 102
4.5 TCP Enhancements 104
4.5.1 TCP Reno: Simulation Results 104
4.5.2 SACK TCP: Simulation Results 106
4.5.3 SACK TCP: Analysis of Recovery Behavior 110
4.6 Effect of a Large Number of Sources 113
4.7 Effect of Long Latency: Satellite Networks 114
4.7.1 Simulation Model 115
4.7.2 Simulation Results 118
4.8 Buffer Requirements for TCP over UBR 121
4.8.1 Simulation Model 121

4.8.2 Simulation Results 124
4.9 Chapter Summary 125

5. Guaranteed Rate: Effect of Higher priority Traffic 129

5.1 Chapter Goals 130
5.2 The UBR+ Guaranteed Rate Model 131
5.3 TCP over UBR+ with VBR background 134
5.3.1 Simulation Model 134
5.3.2 Simulation Results 135
5.4 Guaranteed Rate 138
5.4.1 Simulation Results 138
5.5 Chapter Summary 146

6. Guaranteed Frame Rate: Providing per-VC Minimum Rate Guarantees . 147

6.1 Chapter Goals 148
6.2 The Guaranteed Frame Rate Service 148
6.3 TCP Behavior with Controlled Windows 153
6.4 TCP Rate Control using Buffer Management 156
6.5 The Differential Fair Buffer Allocation Scheme 163
6.5.1 DFBA Description 164
6.5.2 DFBA Drop Probability 167
6.5.3 DFBA Thresholds 169
6.6 Simulation Model 171
6.7 Simulation Results 173
6.8 TCP Friendly Buffer Management 179
6.9 Chapter Summary 180

7. ABR Virtual Source / Virtual Destination: Buffer Allocation in Long Delay Networks 182

7.1 Chapter Goals 182
7.2 The VS/VD Option in ABR 183
7.3 The ERICA Switch Scheme 187
7.4 A VS/VD Switch Architecture 190
7.4.1 A Non-VS/VD Switch Model 191
7.4.2 A VS/VD Switch Model 194
7.5 A Per-VC Rate Allocation Algorithm for VS/VD 197
7.6 Simulation Model 202
7.7 Simulation Results 203
7.8 Chapter Summary 205

8. Summary and Future Work 210

8.1 Comparison of ATM Service Categories 211
8.2 Summary of Results 214
8.3 Limitations and Future Work 217

Appendices:

A. ATM QoS: A Description of Current Standards 221

A.1 The QoS Problem 222
A.2 The ATM Forum Quality of Service Model 226
A.2.1 Traffic Parameters 227
A.2.2 QoS Parameters 228
A.2.3 ATM Service Categories 229
A.3 The ITU-T Quality of Service Model 230
A.3.1 I.362 Service Classes 230
A.3.2 ATM Forum support for I.362 233
A.3.3 I.356 QoS Classes 234
A.3.4 I.371 ATM Transfer Capabilities 237

B. Pseudocode 243

B.1 Early Packet Discard 243
B.2 Selective Drop and Fair Buffer Allocation 245
B.3 Differential Fair Buffer Allocation 248
B.4 Virtual Source / Virtual Destination 251

C. Miscellaneous Tables and Results 254

Bibliography 277

Acronyms 284

LIST OF TABLES

Table		Page
2.1	Summary of TCP flavors	45
2.2	Classification of buffer management	55
2.3	Properties of the four categories of buffer management schemes.	63
2.4	Priorities for ATM service categories	70
4.1	Vanilla TCP over UBR : UBR enhancements (Efficiency)	93
4.2	Vanilla TCP over UBR : UBR enhancements (Fairness)	93
4.3	TCP over UBR: Buffer requirements for zero loss	94
4.4	Reno TCP over UBR (Efficiency)	106
4.5	Reno TCP over UBR (Fairness)	107
4.6	SACK TCP over UBR+: (Efficiency)	108
4.7	SACK TCP over UBR+: (Fairness)	108
4.8	TCP over UBR: Comparative Efficiencies	112
4.9	TCP over UBR: Comparative Fairness	112
4.10	TCP over Satellite (UBR): Efficiency	119
5.1	SACK TCP with VBR (strict priority) : Efficiency	136

5.2 Allocation of Variation:Efficiency 142

5.3 Allocation of Variation:Fairness 143

6.1 Fifteen TCP buffer thresholds 158

6.2 Fifteen TCP throughputs 159

6.3 Fifteen TCP buffer:throughput ratio 160

6.4 DFBA: 50 TCPs 5 VCs, 50% MCR Allocation 173

6.5 DFBA: 50 TCPs 5 VCs, 85% MCR Allocation 174

6.6 DFBA: 100 TCPs 5 VCs, 85% MCR Allocation 175

6.7 DFBA: Effect of Buffer Size (6 kcells) 176

6.8 DFBA: Effect of Buffer Size (3 kcells) 176

6.9 Heterogeneous RTT. VC3 = 60ms RTT 177

6.10 Minimum rate guarantees with DFBA. GEO backbone 178

6.11 DFBA: Effect of Z_i 179

A.1 The ATM Forum QoS Model 231

A.2 I.362 Service Classes 232

A.3 A proposed mapping for ATM Forum and I.362 234

A.4 I.356 cell transfer performance objectives – default values 235

A.5 I.356 QoS Classes 236

A.6 I.371 ATCs 241

A.7 QoS classes for carrying ATCs 241

A.8 QoS classes for service categories 241

C.1 TCP over UBR+: Parameter analysis of Fair Buffer Allocation: 5 sources, LAN 255

C.2 TCP over UBR+: Parameter analysis of Fair Buffer Allocation: 15 sources, LAN 256

C.3 TCP over UBR+: Parameter analysis of Fair Buffer Allocation: 5 sources, WAN 257

C.4 TCP over UBR+: Parameter analysis of Fair Buffer Allocation: 15 sources, WAN 258

C.5 TCP over UBR+: Parameter analysis of Selective Drop: LAN 259

C.6 TCP over UBR+: Parameter analysis of Selective Drop: WAN 260

C.7 Guaranteed Rate: TCP with VBR (300ms on/off): Efficiency for LAN 261

C.8 Guaranteed Rate: TCP with VBR (300ms on/off): Efficiency for WAN 262

C.9 Guaranteed Rate: TCP with VBR (300ms on/off): Fairness for LAN 263

C.10 Guaranteed Rate: TCP with VBR (300ms on/off): Fairness for WAN 264

C.11 Guaranteed Rate: TCP with VBR (300ms on/off): Satellite 265

C.12 TCP with VBR (300ms on/off) over UBR+ with GR : Satellite . . . 266

C.13 Guaranteed Rate: TCP with VBR (300ms on/off): Satellite 267

LIST OF FIGURES

Figure **Page**

1.1 Components of traffic management 7

2.1 TCP/IP over ATM protocol layers 15

2.2 TCP/IP over ATM: Protocol data units 16

2.3 Traffic management for TCP/IP over ATM 18

2.4 Load versus delay and throughput 23

2.5 Vanilla TCP: Slow start and congestion avoidance 30

2.6 TCP Reno: Fast retransmit and recovery 34

2.7 TCP New Reno: The fast retransmit phase 36

2.8 TCP SACK: Selective acknowledgments 38

2.9 ABR traffic management 65

2.10 Explicit rate feedback 66

2.11 LEO satellite in access networks 67

2.12 GEO satellite in backbone networks 68

2.13 Satellite-ATM network architecture 74

2.14 The TCP over satellite-ATM protocol stack 75

2.15 A queuing architecture for ATM 75

3.1 Research Methodology 83

4.1 Design issues for TCP over UBR 86

4.2 The N-source TCP configuration 88

4.3 Selective Drop and FBA: Buffer occupancies for drop 99

4.4 FBA: Effect of the minimum drop threshold 101

4.5 FBA: Effect of the linear scale factor 102

4.6 N-Source GEO Configuration 115

4.7 N-Source LEO Configuration 117

4.8 Buffer requirements for 30 ms RTT 121

4.9 Buffer requirements for 120 ms RTT 122

4.10 Buffer requirements for 550 ms RTT 123

5.1 Switch model for UBR+ with GR 131

5.2 Link Capacity allocations for VBR and UBR with GR 133

5.3 The N source TCP configuration with VBR 134

6.1 Use of GFR in ATM connected LANs 149

6.2 Network Architecture with tagging, buffer management and scheduling 151

6.3 Single TCP Congestion Window Control. Drop thresholds (bytes of window size) = 125000, 250000, 500000, None 155

6.4 N source configuration 157

6.5 15 TCP rate control by packet drop 161

6.6 DFBA Target Operating Region 164

6.7 DFBA Drop Regions 166

6.8 DFBA Buffer Occupancies for Drop 167

6.9 DFBA Simulation Configuration 172

6.10 Per-VC Buffer Occupancy Levels 175

7.1 End-to-End Control vs VS/VD Control 185

7.2 A satellite-ATM network 189

7.3 VS/VD switch architecture 191

7.4 Queuing model for non-VS/VD switch 192

7.5 Queuing model for per-VC VS/VD switch 197

7.6 Data Cell Received 201

7.7 FRM cell received 201

7.8 BRM cell received 202

7.9 End of averaging interval 203

7.10 Time to send expires (now() $\geq$ time_to_send$_{ij}$) 207

7.11 Five sources satellite configuration 208

7.12 Switch Queue Length for VS/VD and non-VS/VD:GEO 208

7.13 Switch Queue Length for VS/VD and non-VS/VD Case: LEO 209

A.1 QoS Triangle 223

B.1 EPD: Initialization 243

B.2 EPD: Cell dequeued 244

B.3 EPD: Cell Received 244

B.4 SD and FBA: Initialization 245

B.5 SD: Cell dequeued 246

B.6 SD and FBA: Cell Received 247

B.7 DFBA: Initialization 249

B.8 DFBA: Cell dequeued 249

B.9 DFBA: Cell Received 250

B.10 VS/VD: Data Cell Received 251

B.11 VS/VD: FRM cell received 251

B.12 VS/VD: BRM cell received 252

B.13 VS/VD: End of averaging interval 252

B.14 VS/VD: Time to send expires (now() $\geq$ time_to_send$_{ij}$) 253

C.1 Link Utilizations for VS/VD and non-VS/VD:GEO 268

C.2 Link Utilizations for VS/VD and non-VS/VD: LEO 269

C.3 ACRs for VS/VD and non-VS/VD:GEO 270

C.4 ACRs for VS/VD and non-VS/VD Case:LEO 271

C.5 Switch Queue Comparsion for Different t0v (config55) 272

C.6 Switch Queue Comparsion for Different t0v (config10) 273

C.7 Link Utilization Comparsion for Different t0v (config10) 274

C.8 TCP Configuration with VS/VD 275

C.9 TCP Configuration for t0v = 5000 276

LIST OF RESULTS

Result **Page**

4.1 TCP over vanilla UBR results in low fairness in both LAN and WAN configurations. 91

4.2 The default TCP maximum window size leads to low efficiency in LANs. 92

4.3 For a switch to guarantee zero loss for TCP over UBR, the amount of buffering required is equal to the sum of the TCP maximum window sizes of all the TCP connections. 95

4.4 EPD improves the efficiency of TCP over UBR, but it does not significantly improve fairness in LANs. 96

4.5 There is a tradeoff between efficiency and fairness. 102

4.6 The fairness of FBA is sensitive to parameters. 103

4.7 Both Selective Drop and FBA improve both fairness and efficiency of TCP over UBR. 103

4.8 Fairness and efficiency increase with increase in buffer size. 104

4.9 For long latency connections (WAN), fast retransmit and recovery hurts the efficiency. 104

4.10 Fast retransmit and recovery improves the efficiency of TCP over UBR for the LAN configuration. 105

4.11 The addition of EPD with fast retransmit and recovery results in a large improvement in both fairness for LANs. 105

4.12 For most cases, for a given drop policy, SACK TCP provides higher efficiency than the corresponding drop policy in vanilla TCP. 106

4.13 For LANs, the effect of drop policies is very important and can dominate the effect of SACK. 107

4.14 The throughput improvement provided by SACK is significant for wide area networks. 109

4.15 The performance of SACK TCP can be improved by intelligent drop policies like EPD and Selective drop. 109

4.16 The fairness values for selective drop are comparable to the values with the other TCP versions. 109

4.17 For long delays, selective acknowledgments significantly improve the performance of TCP over UBR. 118

4.18 As delay increases, fast retransmit and recovery is detrimental to the performance of TCP. 119

4.19 The effect of intelligent buffer management policies studied above is not significant in satellite networks. 120

4.20 A buffer size of 0.5RTT at the bottleneck provides high efficiency and fairness to TCPs over UBR+ for satellite networks. 125

5.1 When strict priority traffic starves TCP traffic, throughput may be degraded. 137

5.2 For LANs, the dominating factors that effect the performance are the switch drop policy and the buffer size. 144

5.3 For WANs, the dominating factor is the GR. A GR of 0 hurts the TCP performance. 144

5.4 For satellite networks, the TCP congestion control mechanism makes the most difference; SACK TCP produces the best results and Reno TCP results in the worst performance. 145

6.1 TCP throughput can be controlled by controlling its congestion window, which in turn, can be controlled by setting buffer thresholds to drop packets. . . . 163

6.2 With a FIFO buffer, the average throughput achieved by a connection is proportional to the fraction of the buffer occupancy of the connection's cells. . . 163

6.3 DFBA satisfied the MCR guarantees for 50 TCPs with heterogeneous MCRs and 50% MCR allocation. 173

6.4 The unallocated link capacity is utilized efficiently. The overall system efficiency with DFBA is high, even with low MCR allocation. 174

6.5 DFBA provides MCR guarantees with high MCR allocation. 174

6.6 MCR guarantees provided by DFBA are not dependent on the number of TCP connections in the network. 175

6.7 A buffer size of half round trip delay-bandwidth product is sufficient for DFBA to provide MCR guarantees. 176

6.8 DFBA does not have a bias against VCs with large RTTs. 177

6.9 DFBA can be used to provide MCR guarantees over long delay networks such as satellite-ATM networks based on GEO systems. 178

6.10 The drop probability function controls the excess capacity sharing policy in DFBA. 178

7.1 ABR switches must have buffers proportional to the round trip delay-bandwidth

product of the ABR feedback control loop to which they belong. 205

7.2 ABR switches at the edges of two VS/VD control segments must have buffers proportional to the round trip delay-bandwidth product of the upstream VS/VD segment to which they are connected. 205

CHAPTER 1

Introduction

Current and future networking technologies are expected to fulfill the goal of delivering Quality of Service (QoS) guarantees across integrated services networks. These networks carry voice, video and data traffic on the same physical link. Each kind of traffic (also called traffic class) has specific QoS requirements. QoS requirements are typically specified in terms of timely delivery of packets at a negotiated rate with minimal packet loss across the network. Networks must achieve a high multiplexing gain for bursty traffic as well as maintain QoS guarantees.

The current Internet uses the Transmission Control Protocol (TCP) with the Internet Protocol (IP) [82, 18] to reliably transport user data. TCP/IP is the most widely used set of protocols in the existing Internet, and most applications such as World Wide Web (WWW), File Transfer Protocol (FTP) and Telnet use TCP/IP for data transfer. Extensive research efforts to improve TCP also suggest that TCP is expected to be the transport protocol of choice for the future Internet. The TCP/IP protocol suite is designed to be transparent to the underlying network technology. Asynchronous Transfer Mode (ATM) [97, 49, 10] is one such network technology that can support TCP/IP.

ATM is the new generation of computer and communication network technology. ATM uses a point-to-point network architecture and transports data over Virtual Channels (VCs) using fixed size 53 bytes long packets, called cells. ATM distinguishes itself from legacy networking technologies by providing a framework for various types of services, and by specifying end-to-end Quality of Service (QoS) guarantees for these services. In this regard, ATM is a multiservice network technology that lays the foundation for Broadband Integrated Services Digital Networks (B-ISDN). The ATM service categories include real time services – Constant Bit Rate (CBR) and real-time Variable Bit Rate (rt-VBR), and non real-time services – non real-time Variable Bit Rate (nrt-VBR), Available Bit Rate (ABR), Guaranteed Frame Rate (GFR) and Unspecified Bit Rate (UBR) [34]. Of these, ABR, GFR and UBR are intended to be primarily used by data applications. These data services are expected to transport TCP/IP data and provide best effort services as well as basic fairness and rate guarantees to their users.

ATM networks are being deployed throughout the information infrastructure – in campus backbones, wide area networks as well as in satellite networks. ATM networks provide a framework for service guarantees to applications that may need them. The TCP/IP protocol suite is the most widely used protocol suite used in the Internet, and a majority of applications use TCP as their transport layer protocol. As a result, it is very important that ATM networks support the seamless and efficient transport of TCP/IP data for a variety of network topologies including Local Area Networks (LANs), Wide Area Networks (WANs) and Satellite networks.

Traffic management is a key problem in designing high speed networking architectures. The goal of traffic management is to control network congestion, efficiently

utilize network resources and deliver the negotiated quality of service to users [55]. ATM supports a comprehensive QoS framework for the transport of higher layer data, and provides several real-time as well as non real-time services. TCP implements a variety of flow and congestion control mechanisms such as slow start/congestion avoidance, fast retransmit/fast recovery and selective acknowledgments. In addition, the network has several functional tools available for traffic management. These include connection admission control (CAC), policing, shaping, scheduling, buffer management and feedback control. These functions can be used to optimize network and end-user resources while providing QoS guarantees to the user.

In this research, we identify one distinct problem in traffic management – *transporting Internet protocols over ATM.* In particular, we address the problem of traffic management for TCP/IP over ATM networks. The goal is to design a network architecture and its mechanisms to efficiently transport TCP data over the various ATM service categories (and thus meeting the requirements specified by those service categories), using the above functions available to the network. We focus on network based solutions to manage queues and deliver high end-to-end performance for the TCP protocol. Although the focus of this research is on TCP over ATM networks, the techniques presented here are applicable to any other networking technology that supports a similar QoS framework.

In the next section, we provide an overview of the various functions and components of traffic management listed above. We highlight the components for which we will develop new implementations, namely *buffer management and feedback control.* A detailed background and survey of existing algorithms for these components is given in chapter 2.

1.1 Components of Traffic Management

The goal of traffic management is to fairly allocate resources to network users, to efficiently utilize network resources such as buffer space and link capacity, and to isolate a traffic class from the effects of other traffic that may be violating its contract. In addition, a traffic management system must provide negotiated QoS guarantees to its constituent users. The system should work for a wide variety of realistic traffic workloads, provide seamless interoperability of current Internet protocols like the Transmission Control Protocol / Internet Protocol suite (TCP/IP) with link layer protocols like ATM, and be a single solution for all network topologies including local area networks, wide area networks as well as satellite networks. Finally, the system should be implementable in a cost-effective manner.

Note that the congestion control problem [58] is a subset of the traffic management problem. Congestion in a network or in a network node occurs when the demand for resources exceeds the supply. Network resources include buffers, link capacity and processor time. Congestion typically results in buffer overflows and subsequent packet loss. Lost packets may be retransmitted by higher layers, leading to further congestion. Congestion is a dynamic problem and static solutions like increasing memory size, increasing link capacity and increasing processing power do not eliminate congestion. Jain [58] shows that partial deployment of static solutions increases heterogeneity in the network and leads to even more congestion. As a result, it is very important to design mechanisms to efficiently manage buffers and link bandwidth so as to provide dynamic solutions for congestion avoidance and control. Traffic management attempts to solve the congestion problem in a multiservice network. Traffic management must not only control congestion, it must also provide QoS guarantees

to users of a network that supports a variety of real-time and non real-time services. Figure 1.1 illustrates the components for traffic management in a network. The figure illustrates two end systems communicating over two network clouds. Shaping, policing, scheduling, buffer management and feedback control are highlighted in figure 1.1. We briefly discuss these below.

Shaping and Policing

The end systems are responsible for sending data that conforms to a negotiated traffic contract. Informally, a traffic contract typically specifies the average rate, peak rate and maximum burst size of a traffic flow. The end system may use a *traffic shaping* function to ensure conformance to the contract. The network may *police* the incoming traffic to test its conformance to the contract. Policing is typically performed at the entrance to the network. If an incoming packet is non-conforming (out of profile), the policing function (also called Usage Parameter Control (UPC)) has four possible choices.

- **Dropping:** It may drop the packet, thus making sure that the traffic entering the network conforms to the contract (is in profile).
- **Tagging:** It may mark the packet as a low priority packet by setting one or more bits in the packet header. In ATM, this is done by setting the value of the Cell Loss Priority (CLP) bit in the ATM header to 1. During congestion, the network may choose to discard low priority packets in preference to high priority packets.

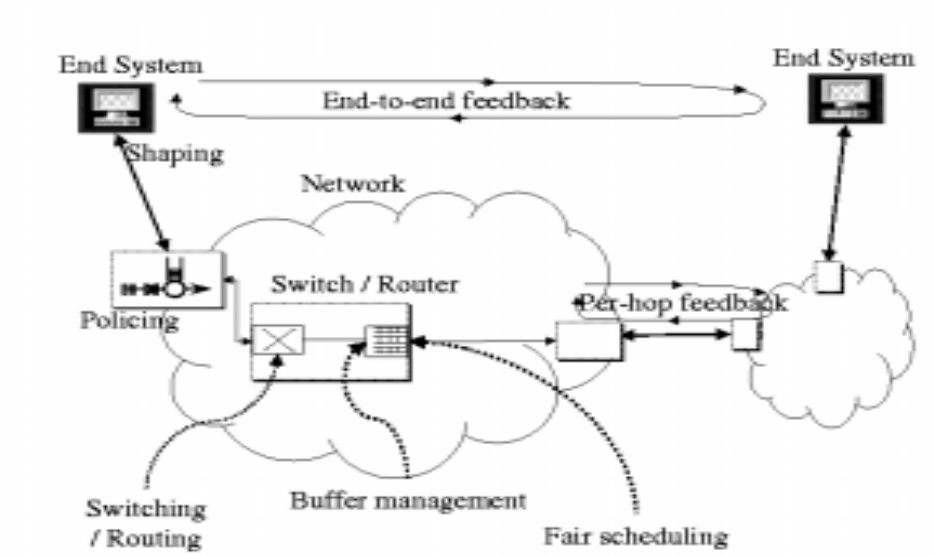
- **Buffering:** The policer may also buffer the packet at the entrance to the network and send the packet at a later time when the packet becomes conforming to the contract. This might be a desired solution if the edge device performing the policing has information about the interior state of the network and temporarily holds the packet until the network is uncongested.

- **No action:** The non-conforming packet may be allowed into the network without any changes. This is typically an undesirable solution because it can cause congestion in the interior of the network.

Extensive research has been conducted in shaping and policing. Partridge [80] describes the *leaky bucket algorithm* which is the most widely used algorithm for shaping and policing. Several variations of leaky bucket are also common. These include the token bucket algorithm and the Generic Cell Rate Algorithm (GCRA) [35] that is used in ATM. In this research we do not design a new policing/shaping component. Where a shaping or policing function is needed, we assume that the GCRA algorithm or its variant is used.

Queuing and Scheduling

At each hop within the network, packets are switched (or routed) to the output port of the next hop and may be queued in the port buffer before being sent out on the link. In our work, we assume that the switch is an *output queued* switch. In an input queued switch, the queuing is performed before the switching. Input queuing raises several performance and design problems such as head of line blocking that can restrict the throughput achievable by the switching matrix. A comparison of input versus output queued switches is beyond the scope of this work. Although in this



Traffic management consists of several components including policing and shaping, buffer management, queuing and scheduling, and feedback control.

Figure 1.1: Components of traffic management

work, we assume that switches are output queued, the techniques presented here are equally applicable to input queued switches.

The packets of different VCs may be queued separately (per-VC queuing), or may be queued in a single queue with packets of other VCs (FIFO queuing). In general, a hybrid queuing architecture is designed where both per-VC and FIFO queuing are used depending on the traffic class. Higher priority classes may allocate per-VC queues, while best effort classes may use per-class queues. We describe the queuing architecture used in this research in chapter 2.

A scheduling function controls the exit of packets from the queues of a link. The scheduling component services each queue according to a scheduling policy (or service

discipline). Popular service disciplines include Weighted Fair Queuing [23], Weighted Round Robin [102] and Class Based Queuing [29].

Buffer Management

A buffer management component controls the entry of the packet into the queue. Depending on the buffer and connection states, the function may discard the packet, or may admit the packet into an appropriate queue. Once admitted, the packet waits for its turn behind other packets in the queue to be scheduled on to the link. Buffer management is an important component of this research. While some work on buffer management has been conducted, the performance and service guarantees provided by existing buffer management schemes are not adequate to meet the requirements of an ATM network that supports TCP/IP traffic. The goal of buffer management is to fairly and efficiently allocate network buffers to user data. The buffer management function must also be sensitive to the behavior of higher layer protocols and their response to packet loss. In particular, the buffer management mechanism must be *friendly* to the TCP protocol [73]. In this research, we perform experiments to illustrate how the existing schemes, Early Packet Discard and Fair Buffer Allocation perform, and develop two new buffer management algorithms, Selective Drop and Differential Fair Buffer Allocation.

Feedback Control

Nodes within a network may provide feedback to one another about their congestion state. Feedback in ATM is enabled by the ABR service category using Resource Management (RM) cells. At the IP layer, feedback may be provided in the form

of ICMP messages. Feedback can be exchanged by any nodes in the network – it can be end-to-end, or hop-by-hop. It can be performed at any protocol layer. For example, the ATM layer implements feedback in the form of ABR RM cells, while the TCP layer also implements feedback in the form of window advertisements and acknowledgments. Feedback is useful because it provides the ability to control the rate of packets entering the network using the knowledge of the congestion state of the interior. The ABR service category provides rate based closed loop feedback control that can be either end-to-end or segment-by-segment. Kalyanaraman [61], addresses the problem of end-to-end feedback control in ATM. Segment-by-segment feedback control in ABR, called the virtual Source / Virtual Destination (VS/VD) option, has not been studied extensively. We argue that VS/VD is useful in buffer sizing in networks with heterogeneous latencies connected to each other. We design a VS/VD scheme that substantiates our claim.

1.2 Key Contributions of this Work

The primary goal of this research is to design components of traffic management to optimize the performance of TCP/IP over ATM. We expect that our research will provide insight into network architecture. In particular, we will provide guidelines about what level of quality of service can be achieved by different queuing, buffering, feedback and scheduling mechanisms. The mechanisms proposed here can be used with other traffic management components to deliver quality of service across a network. The system proposed here is based on an ATM network architecture. However, the techniques proposed are applicable to a more general class of quality of service networks. The following is a list of the key tasks and deliverables for this research:

1. Performance evaluation of TCP/IP transport over various ATM service categories. We are particularly interested in the UBR, GFR and the ABR service categories.

2. The logical architecture of the network queues in a multiservice ATM network, and the assignment of VCs of ATM service categories to these queues.

3. The techniques for managing network queues, including buffer management, feedback control, and rate guarantees.

4. The techniques for providing efficient and fair network operation for best effort TCP/IP transport.

5. The techniques for providing simple rate guarantees to ATM VCs carrying TCP/IP traffic.

6. A comparison of ATM service categories for TCP/IP transport.

7. A study of the above traffic management issues for various network types, including campus networks, wide area networks and satellite networks.

1.3 Outline of this Dissertation

The rest of this dissertation is organized as follows.

Chapter 2 discusses the background material necessary for understanding the problem of traffic management for TCP/IP over ATM. The chapter provides a brief overview of the interoperability standards and design issues for TCP/IP over ATM. A discussion of ATM standards is followed by a description of traditional TCP/IP congestion control mechanisms. The chapter contains a comprehensive survey of the

state of the art in TCP/IP and buffer management in ATM networks. The chapter also presents an overview of the ABR service, and of how ATM can be deployed over satellite networks for TCP/IP data transport.

Chapter 3 clearly defines the problem we are trying to solve, and discusses our approach and methodology for solving the problem. The chapter also presents the definitions of the performance metrics used in the analysis. Our approach breaks the problem into four parts discussed in the following four chapters.

Chapter 4 presents performance results for TCP over the UBR service category. The chapter analyses two buffer management schemes, Early Packet Discard and Fair Buffer Allocation, and proposes the Selective Drop scheme for UBR VCs. UBR with intelligent buffer management is termed UBR+. Performance results of TCP over UBR+ for LAN, WAN and satellite latencies are presented. The key result of the chapter is that both buffer management schemes and TCP enhancements can help improve the performance of TCP over UBR.

Chapter 5 analyzes the effect of higher priority variable bit rate background traffic on TCP over UBR+. The simulations show that high priority traffic can degrade TCP performance in some cases. The chapter then studies the effect of providing a minimum rate guarantee to UBR traffic to prevent starvation from higher priority traffic. An extensive set of simulations and performance analysis leads to a comparison of the relative effects of buffer management, TCP enhancements, buffer sizes and guaranteed rates for LAN, WAN and satellite networks.

Chapter 6 extends the notion of rate guarantees to per-VC rate guarantees. The design of the ATM Guaranteed Frame Rate service category is discussed. The chapter

presents mechanisms for controlling TCP rates using buffer management. The Differential Fair Buffer Allocation (DFBA) technique is proposed for providing per-VC minimum rate guarantees to GFR VCs carrying TCP/IP traffic. The chapter presents simulation results to show how DFBA performs in a variety of network scenarios.

Chapter 7 studies the use of feedback control to supplement buffer management and provide rate guarantees. The chapter presents a feedback control algorithm that uses the virtual source / virtual destination option in the ABR service category. Simulation results illustrate that this algorithm is useful in buffer allocation in large bandwidth-delay networks such as satellite networks.

Chapter 8 concludes this dissertation with a summary of the results and limitations of this work. The chapter also presents future directions for research in traffic management for broadband networks.

CHAPTER 2

Background and State of the Art

In this chapter, we provide a description of the background material for this dissertation and a survey of the state of the art in TCP/IP over ATM networks. This chapter is organized as follows. We first describe the interoperability standards for the transport of TCP/IP over ATM. We then provide an overview of the design options for optimizing the performance of TCP/IP over ATM. We describe each design option and provide a survey of the existing state of the art in that field. Our discussion includes a description of TCP congestion control, ATM service categories, buffer management and feedback control. Finally, we present an overview of satellite-ATM networks. This discussion highlights the peculiarities of this emerging class of ATM networks that can be used as an alternative to terrestrial ATM networks.

2.1 Transporting TCP/IP over ATM

The Internet Request For Comments (RFC) number 2225 [68] provides the interoperability specification for the transport of TCP/IP over ATM. This project uses the baseline operational framework for IP and Address Resolution Protocol (ARP) over ATM described in Request For Comments (RFC) 2225. In this section, we briefly present this protocol architecture for IP over ATM.

Figure 2.1 illustrates the protocol layers for TCP/IP over ATM. Figure 2.2 illustrates the structures of Protocol Data Units (PDUs) at each layer. The application layer packets get broken into TCP segments. The TCP Maximum Segment Size (MSS) is based on the Maximum Transmission Unit (MTU) of the link layer. A TCP segment is less than MSS bytes long. The TCP layer attaches a 20 byte header to the segment and passes it to the IP layer which attaches another 20 byte header. The IP packet is then processed by the IP-ATM interworking layer specified by RFC 2225. The layer attaches an 8 byte Logical Link Layer (LLC) header that contains the Protocol ID for the higher layer. It is in this header that IP packets are distinguished from packets of other network layer protocols. The packet is then passed on to the ATM Adaptation Layer 5 (AAL 5), that attaches an 8 byte trailer to form an AAL 5 frame. The AAL 5 layer then breaks the frame into ATM cell payloads of size 48 bytes each and attaches 5 byte ATM cell headers to each payload. Each cell header includes a bit called the End of Message (EOM) bit. All cells in the frame except the last cell have this bit cleared (set to 0). This bit is used by ATM layer devices to recognize frame boundaries.

The IP-ATM interworking unit can reside within a host system, or can be an edge device at the boundary of an IP network connecting to the ATM network. In either case, the device implements IP over ATM as specified by RFC 2225. Also note that the IP data can be transported over any ATM service category. The mapping of ATM VCs to service categories is handled by the ATM layer. In this research, we focus on the UBR, GFR and ABR service categories.

The RFC also describes the use of an ATM Address Resolution Protocol (ATM ARP) for translating IP addresses into ATM addresses. This is used to setup switched

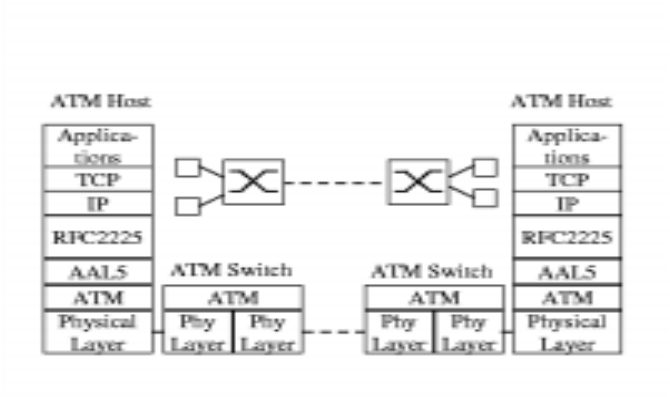


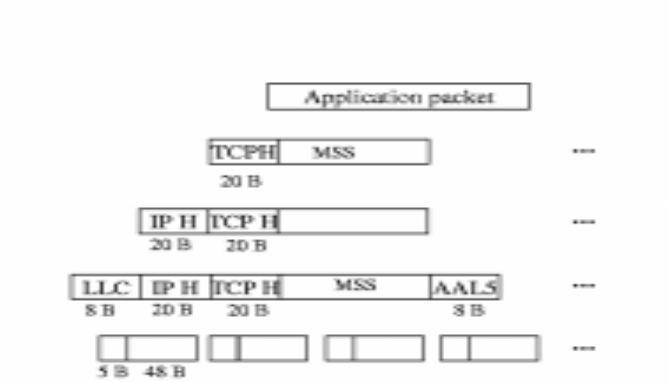
Figure 2.1: TCP/IP over ATM protocol layers

VCs between ATM nodes. In this research, we assume that VCs have been setup using either ARP or have been provisioned using Permanent Virtual Circuits (PVCs). For a detailed specification of ARP, the reader is referred to [68].

The IETF specification does not define any mapping from a TCP connection to a VC. As a result, one or more TCP connections may be transported over a single VC. For example, each host may establish a single UBR VC to the switch and transport all traffic over that VC. It is conceivable that a single TCP connection may be transported over multiple VCs. However, in this situation, care must be taken to preserve ordering of packets of a single TCP connection. Reordering packets of a TCP connections is undesirable as it leads to the triggering of TCP fast retransmits and can lead to excessive congestion.

In this work, we use two kinds of TCP–VC mappings

- **One-to-one**: In this case, each TCP connection is mapped on to a separate VC. In real deployments, this is not a very scalable solution. However, a one-to-one mapping makes it easier to study per-TCP behavior using per-VC behavior.



An application packet is processed by each layer in the protocol stack and finally broken into ATM cells

Figure 2.2: TCP/IP over ATM: Protocol data units

Almost all other research in this area has used a one-to-one mapping between TCPs and VCs [70, 69, 89].

Many-to-one: In this case, several TCP connections are transported over a single VC. This is a more realistic scenario, especially for bt effort services lik e UBR, GFR and ABR. An IP-ATM edge device (either a host or a dge router) performs the aggregation of TCPs over a single VC. The aggregation process must ensure that cells from different IP packets are not interleaved. This is because, at the ATM layer, interframe boundaries are determined only by the EOM bit. If cells from different frames are interleaved, then the destination end system cannot distinguish between interleaved cells from different frames. Although a many-to-one mapping is more realistic, it has not been studied in past research on this area.

2.2 Traffic Management for TCP/IP over ATM

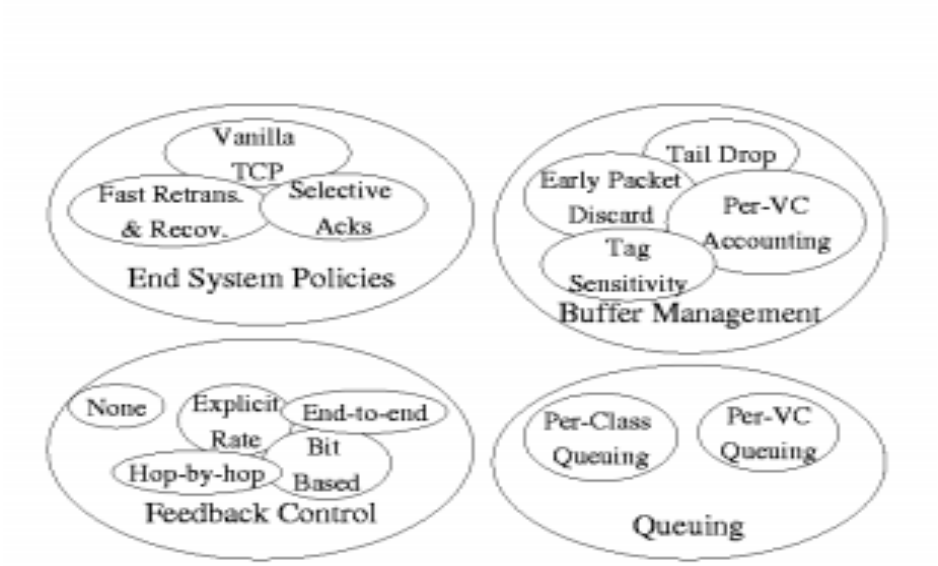
Besides the interoperability issues presented in the previous section, several traffic management issues have largely been unsolved. The focus of this work is to study the traffic management issues briefly described below.

Figure 2.3 illustrates a framework for the various design options available to networks and TCP hosts for traffic Management. The figure is divided into four groups

- Buffer management
- Queuing
- TCP end-system policies
- Feedback control

The mechanisms outlined in the figure can be used to implement various ATM services in the network. Best effort ATM service can be provided by FIFO queuing and tail drop. This could be a baseline implementation of the UBR service category. Enhancements that perform intelligent buffer management policies at the switches can be developed for UBR to improve transport layer throughput and fairness.

Providing a minimum Guaranteed Rate (GR) to the UBR traffic can also improve TCP performance over UBR. The goal of providing guaranteed rate is to protect the UBR service category from total bandwidth starvation by providing a continuous minimum bandwidth guarantee. Guaranteed Frame Rate (GFR) has been recently proposed in the ATM Forum as an enhancement to the UBR service category. Guaranteed Frame Rate provides a minimum rate guarantee to VCs at the frame level. The GFR service also allows for the fair usage of any extra network bandwidth. GFR



End system policies, buffer management policies, feedback control policies and queuing policies can be used to optimize the performance of TCP/IP over ATM.

Figure 2.3: Traffic management for TCP/IP over ATM

and GR can be implemented using per-VC queuing or per-class queuing and buffer management.

The Available Bit Rate (ABR) service category is another option to implement TCP/IP over ATM. ABR connections use a rate-based closed-loop feedback-control mechanism for congestion control. The network tries to maintain a low Cell Loss Ratio (CLR) by changing the allowed cell rates (ACR) at which a source can send. Switches can also use the virtual source/virtual destination (VS/VD) feature to segment the ABR control loop into smaller loops. ABR can be implemented using the feedback control mechanisms in figure 2.3.

In addition to network based drop policies, end-to-end flow control and congestion control policies can be effective in improving TCP performance over UBR. The fast retransmit and recovery mechanism can be used in addition to slow start and congestion avoidance to quickly recover from isolated segment losses. The selective acknowledgments (SACK) option has been proposed to recover quickly from multiple segment losses. A change to TCP's fast retransmit and recovery has been suggested in [48] to improve the performance of fast retransmit and recovery.

2.3 An Overview of ATM Service Categories

We now briefly describe the current ATM standards for Quality of Service. In this discussion we focus on the best effort services that are studied in this research. A complete description of the state of the art in ATM QoS standards is given in appendix A These standards are primarily provided by the ATM Forum and the International Telecommunications Union (ITU).

ATM networks carry traffic from multiple service categories and support Quality of Service (QoS) requirements for each service category. Each service category is defined using a traffic contract and a set of QoS parameters [34].

The *traffic contract* is a set of parameters that specify the characteristics of the source traffic. This defines the requirements for compliant cells of the connection. The traffic contract consists of:

- *The source traffic descriptors.* These are used to specify the characteristics of the traffic from a source end system (SES) and consist of the Peak Cell Rate (PCR), the Sustained Cell Rate (SCR) and the Maximum Burst Size (MBS).

- *The Cell Delay Variation Tolerance (CDVT) and Burst Tolerance (BT)* parameters are used to specify a tolerance for PCR and SCR respectively. The Generic Cell Rate Algorithm (GCRA) specified in [34] (a version of the leaky bucket algorithm) is used to enforce the PCR/CDVT and SCR/BT parameters.

The *QoS parameters* are negotiated by the source with the network and used to define the expected quality of service provided by the network. The parameters are – *Maximum Cell Transfer Delay (Max CTD)*, *Peak to Peak Cell Delay Variation (peak-to-peak CDV)*, and *Cell Loss Ratio (CLR)*. For each service category, the network guarantees the negotiated QoS parameters if the end system complies with the negotiated traffic contract. For non-compliant traffic, the network need not maintain the QoS objective.

The *Constant Bit Rate (CBR)* service category guarantees a constant rate specified by PCR. The network guarantees that all cells emitted by the source that conform to this PCR are transferred by the network at PCR. The *real time Variable Bit Rate (VBR-rt)* class is characterized by PCR, SCR and MBS that controls the bursty nature of traffic. The network attempts to deliver cells of these classes within fixed bounds of cell delay (max-CTD) and delay variation (peak-to-peak CDV). *Non-real-time VBR* sources are also specified by PCR, SCR and MBS, but the network does not specify the CTD and CDV parameters for VBR-nrt.

The *Available Bit Rate (ABR)* service category is specified by a PCR as well as an Minimum Cell Rate (MCR) which is guaranteed by the network. Excess bandwidth is shared in some fair manner by the network. ABR connections use a rate-based closed-loop feedback-control mechanism for congestion control. The network tries to maintain a low CLR by changing the allowed cell rates (ACR) at which a source

can send. The *Virtual source/Virtual destination (VS/VD)* option in ABR allows the switches to break the ABR control loop into smaller segments so as to isolate networks or subnetworks from one another. In addition, a new service category called *real-time ABR* is currently being proposed in the ATM Forum. This class is expected to carry low quality video traffic that can tolerate some loss. The network in this case would provide loose delay guarantees to the traffic flows.

The *Unspecified Bit Rate (UBR)* does not support any service guarantees. UBR VCs are not required to conform to any traffic contract. PCR however, may be enforced by the network. Switches are not required to perform any congestion control for UBR VCs. When queues become full, switches simply drop cells from UBR connections. Buffer management techniques have been proposed to enhance the UBR service. The enhanced version of UBR has been termed UBR+. The work undertaken in this research has been involved in the recommendations for UBR+. We describe the enhancements to UBR in detail in chapter 4.

The *Guaranteed Frame Rate (GFR)* service category is a proposed enhancement of UBR that guarantees a minimum rate at the frame level. GFR is different from ABR because it does not use feedback control. The GFR class is intended to be a simple enhancement of UBR that guarantees some minimum rate to application frames. We describe GFR in detail in chapter 6.

2.4 A Survey of TCP Congestion Control

The Transmission Control Protocol (TCP) [18] is the most widely used transport protocol in the Internet. TCP is a reliable protocol that uses acknowledgments and a window based flow control protocol for packet transmission and error recovery.

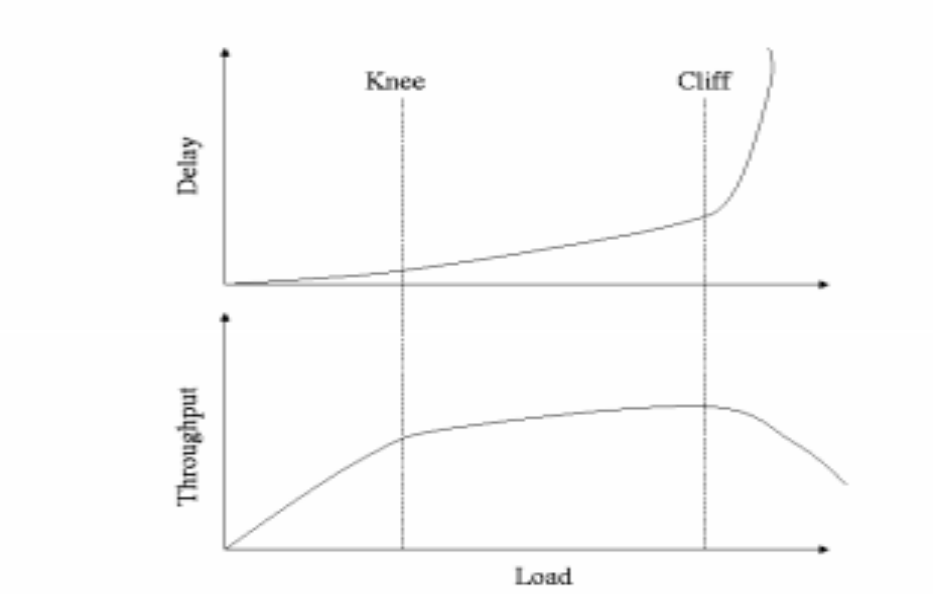
The flow control protocol is based on a selective repeat ARQ mechanism [11, 91]. In addition, TCP also uses the window protocol to implement congestion control mechanisms.

2.4.1 TCP Congestion Control Principles

Congestion occurs when the total demand for network resources exceeds their supply. When users increase their load beyond the capacity of the network, the network becomes congested. Figure 2.4 [21] illustrates the performance of a network in response to increasing load. The figure plots the throughput and delays against load. When the load is low, increase in load increases throughput, and does not significantly increase the response time (delay). When the load reaches the capacity of the network, further increases in load increase delay but not throughput. The increase in delay is due to queuing in the network. An excessive increase in load results in a decrease in effective throughput and a drastic increase in delay. This is because of queue overflow leading to packet retransmissions and a decrease in the useful work done by the network. The point in the graph when the load equals the capacity is called the *knee*. The point when the throughput falls is called the *cliff*. The goal of *congestion avoidance* is to maintain operation at the knee, while the goal of *congestion control* is to maintain operation to the left of the cliff [21].

Jain et. al. [56] describe two components of congestion avoidance, and a set of possible choices for each component:

1. **Network policies:** The network elements measure the load in the system and periodically send congestion signals to end systems.



Below the knee, as load increases, throughput increases and delay remains almost constant. After the knee, delay increases with the load without substantial increase in throughput. When the load exceeds the cliff, delay increases drastically while throughput decreases.

Figure 2.4: Load versus delay and throughput

2. **End system policies:** End systems control their sending rate based on the congestion signals received from the network.

Both network elements and end systems have several options for congestion avoidance and control. In this subsection, we briefly examine these options. This discussion lays the foundation for the mechanisms used by TCP congestion control and the proposed mechanisms used by the buffer management schemes in this research.

Network policies

Network elements have several choices for measuring load and giving feedback. Jain et. al. [56] show that the average queue length is a good indicator of network load.

The averaging is done over a long enough time to include two *regenerative cycles* of the queue. Once the load is determined, the router or switch can provide feedback in several ways, including:

- Send choke packets to the source.
- Set a bit in packet headers indicating congestion.
- Drop packets.

Jain et. al. [56] recommend the use of a bit in the packet header that can be set by routers to indicate congestion. Sources use the feedback bits to increases or decrease their sending rate (or window size)[1]. The set of users to which such feedback is given is determined by fairness criteria for sharing the bottleneck resource. In addition, the end-systems may use implicit signals to detect congestion. Jain describes ways to use increased delay [54] and timeout [53] as indications of congestion. In fact, TCP uses principles very similar to those prescribed in [53]. Once the users (sources/end-systems) get the network feedback (either implicit or explicit) they must react appropriately to the signal.

End system policies

The goal of the end system is to reach steady state operation. A network is in steady state if the number of packets in the network is equal to the network capacity. The network capacity is defined using the bandwidth-delay product of the network

[1] As we will see, TCP does not recognize bit based congestion notification and uses timeout due to packet loss as an indication of congestion. As a result, in this research, we focus on packet dropping schemes for TCP

expressed in packets. For congestion avoidance, the steady state is the knee of the graph in figure 2.4. The end system has two important goals for congestion avoidance:

- How to reach steady state quickly?
- How to maintain steady state in the presence of changing network conditions?

The end system is notified of changing network conditions by the feedback mechanisms above. On receiving feedback from the network, the end system has to make three basic decisions:

1. How to use the feedback signals from the network?
2. How often to change sending rate?
3. How much to change (increase or decrease) the sending rate?

Jain et. al. [21] analyze various increase-decrease policies for window based congestion avoidance schemes. The paper recommends a linear increase and a multiplicative decrease of the window. The optimal linear increase factor is 1 and the optimal multiplicative decrease factor is 0.875. To maintain low oscillations, the frequency of change should be once every two round trip times.

In addition to trying to reach steady state quickly, when congestion results in lost packets, the end system must retransmit the lost packets as efficiently as possible.

TCP uses acknowledgment based window flow and congestion control mechanisms. Flow control ensures that the sender does not overflow the receiver. Congestion control ensures that the sender does not overflow the network. Flow control is enforced by the receiver window (RCVWND) which reflects the buffering available in the

receiver TCP. Congestion control is enforced by the congestion window (CWND) which is as estimate of the network's capacity.

At any given time, TCP can have a minimum of the RCVWND and CWND packets outstanding (unacknowledged). In this research, we focus on network bottlenecks, i.e., we assume that RCVWND > CWND always.

For TCP, steady state operation is reached when CWND equals the network capacity. The basis for TCP congestion control is to obey the *packet conservation principle* during steady state operation [50]. The packet conservation principle states that during steady state, a packet should not enter the network until a packet leaves the network. Jacobson and Karels [50] argue that a system with this property is robust under congestion.

The functions of the basic TCP congestion control mechanisms are outlined below. Various components of TCP congestion control try to address one or more of the following issues:

1. **How to initially estimate steady state?**
2. **How to quickly reach the initial steady state?**
3. **What to do during steady state?**
4. **How to sense congestion?**
5. **What to do during a congestion episode?**
6. **How to recover from packet loss?**
7. **How to sense the end of the congestion episode?**

8. **How to estimate the new steady state?**

9. **At the end of the congestion episode, how to quickly reach the new steady state?**

In the following subsections, we describe the above design choices in the TCP congestion control algorithms and their recently proposed enhancements.

2.4.2 Slow Start and Congestion Avoidance: Vanilla TCP

How to initially estimate steady state?

In the basic TCP congestion control scheme [50] (we refer to this as *vanilla TCP*), the "Slow Start" and "Congestion Avoidance" phases correspond to the non-steady state and steady state phases respectively. The goal of the slow start phase is to reach the steady state as quickly as possible. The congestion avoidance phase is the steady state phase. The variable SSTHRESH is maintained at the source to distinguish between the two phases. SSTHRESH is thus the estimate of the steady state network capacity. Initially, SSTHRESH is assigned a constant value typically equal to 65535 Bytes.

How to quickly reach steady state?

The source starts transmission in the slow start phase by sending one segment (typically 512 Bytes) of data, i.e., CWND = 1 TCP segment. When the source receives an acknowledgment (ACK) for a new segment, the source increments CWND by 1. The source can now send 2 segments (CWND = 2) into the network. In this way, the source increases CWND by 1 segment for every new ACK it receives during the congestion avoidance phase. Since the time between the sending of a segment and

the receipt of its ACK is an indication of the Round Trip Time (RTT) of the connection, CWND is doubled every round trip time during the slow start phase. The slow start phase continues until CWND reaches SSTHRESH and then the congestion avoidance phase begins. The slow start phase takes the CWND from 1 to SSTHRESH in $log_2(SSTHRESH/MSS)$ round trip times, where MSS is the maximum segment size and both MSS and SSTHRESH are expressed in bytes.

What to do during steady state?

During the congestion avoidance phase, the source increases its CWND by 1/CWND every time a segment is acknowledged. Since in each round trip time, TCP sends a single window of data, during the congestion avoidance phase, the congestion window increases by one segment size. During steady state, the TCP probes the network for small increases in capacity by conservatively increasing CWND every RTT. The slow start and the congestion avoidance phases correspond to an exponential increase and a linear increase of the congestion window every round trip time respectively.

How to sense congestion?

If a TCP connection loses a packet, the destination responds by sending duplicate ACKs for each out-of-order packet received. The source maintains a retransmission timeout for the last unacknowledged packet. The timeout value is reset each time a new segment is acknowledged. The source detects congestion by the triggering of the retransmission timeout.

What to do during a congestion episode?

When duplicate ACKs are received, the source does not send any new packets. It simply waits for either a new ACK to arrive or a timeout to occur.

How to sense the end of the congestion episode?

When the timeout occurs, the source has been idle for 1 round trip time. The packet that triggered the timeout was lost 1 retransmission timeout value before the timeout. Assuming the packet was lost due to congestion, the idle period until the timeout is enough time for the congestion to clear. The timeout not only indicates congestion, it also triggers the recovery behavior in the source.

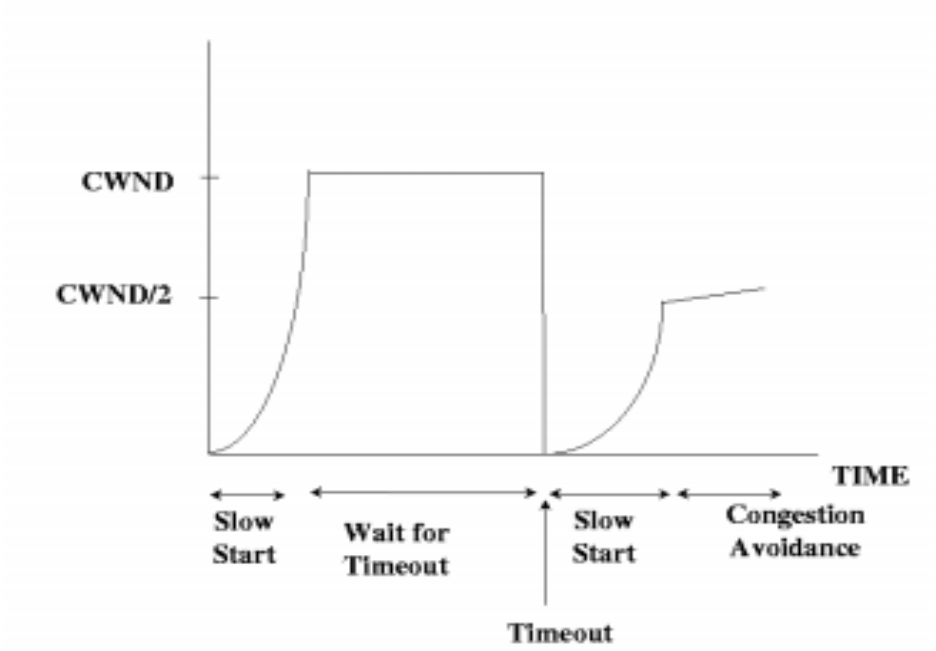
How to estimate the new steady state?

At this point, the source sets SSTHRESH to half of CWND. More precisely, SSTHRESH is set to max{2, min{CWND/2, RCVWND}}. CWND is set to one segment size. SSTHRESH is thus the new estimate of the network capacity (steady state)[2].

At the end of the congestion episode, how to quickly reach the new steady state?

As a result, CWND < SSTHRESH and the source enters the slow start phase. The source then retransmits the lost segment and increases its CWND by one every time a new segment is acknowledged. It takes $log_2(\text{CWND}_{\text{orig}}/(2 \times \text{MSS}))$ RTTs from the point when the congestion was detected, for CWND to reach the target value of half

[2]In effect, the window increase-decrease policy for steady state estimation in TCP is a linear increase and multiplicative decrease as recommended in [56]. The linear increase factor is 1, but the multiplicative decrease factor is 0.5.



TCP congestion control consists of the slow start and congestion avoidance phases. Packet loss triggers timeout and resetting of the congestion window to 1 segment. Slow start is performed until half the original CWND, after which congestion avoidance is performed.

Figure 2.5: Vanilla TCP: Slow start and congestion avoidance

its original size ($\text{CWND}_{\text{orig}}$). Here MSS is the TCP maximum segment size value in bytes. This behavior is unaffected by the number of segments lost from a particular window.

How to recover from packet loss?

If a single segment is lost, and if the receiver buffers out of order segments, then the sender receives a cumulative acknowledgment and recovers from the congestion. Otherwise, the sender attempts to retransmit all the segments starting from the lost segment. In either case, the sender congestion window increases by one segment for each acknowledgment received and not for the number of segments acknowledged.

Note that although the congestion window may increase beyond the advertised receiver window (RCVWND), the source window is limited by the minimum of the two. The typical changes in the source window plotted against time are shown in Figure 2.5.

Most TCP implementations use a 500 ms timer granularity for the retransmission timeout. The TCP source estimates the Round Trip Time (RTT) of the connection by measuring the time (number of ticks of the timer) between the sending of a segment and the receipt of the ACK for the segment. The retransmission timeout is calculated as a function of the estimates of the average and mean-deviation of the RTT [50]. Because of coarse grained TCP timers, when there is loss due to congestion, significant time may be lost waiting for the retransmission timeout to trigger. Once the source has sent out all the segments allowed by its window, it does not send any new segments when duplicate ACKs are being received. When the retransmission timeout triggers, the connection enters the slow start phase. As a result, the link may remain idle for a long time and experience low utilization.

We call the above behavior of TCP congestion control, *Vanilla TCP*. Several modifications to Vanilla TCP have been proposed in recent literature. Some of these have been standardized or are in the process of being standardized by the IETF. In this work, we focus on the modifications to TCP that are considered as standards or potential standards. These are TCP Reno, TCP New Reno and TCP SACK. TCP Tahoe was developed before TCP Reno, and is essentially a subset of TCP Reno. However, TCP Reno is much more widely deployed than Tahoe. As a result, we do not consider the behavior of TCP Tahoe. TCP New Reno is an enhancement to TCP Reno. We describe the protocol in section 2.4.4. In our simulation we use Vanilla,

Reno and SACK because of their common use or projected use in real deployments. We briefly describe the above enhancements in the following subsections.

2.4.3 Fast Retransmit and Recovery: TCP Reno

TCP Reno introduces an additional congestion control mechanism called Fast Retransmit and Recovery (FRR). Fast Retransmit and Fast Recovery is designed to improve TCP performance when a single segment is lost. Current TCP implementations use a coarse granularity (typically 500 ms) timer for the retransmission timeout. As a result, during congestion, the TCP connection can lose much time waiting for the timeout (detecting congestion). In Figure 2.5, the horizontal CWND line shows the time lost in waiting for a timeout to occur. During this time, the TCP neither sends new packets nor retransmits lost packets. Moreover, once the timeout occurs, the CWND is set to 1 segment and then the connection takes several round trips to efficiently utilize the network. TCP Reno implements the fast retransmit and recovery algorithms that enable the connection to quickly recover from isolated segment losses [92].

Fast retransmit and recovery changes the following mechanisms in TCP:

- How to detect congestion?
- What to do during the congestion episode?
- How to sense the end of the congestion episode?
- At the end of the congestion episode, how to quickly reach the new steady state?
- How to recover from packet loss?

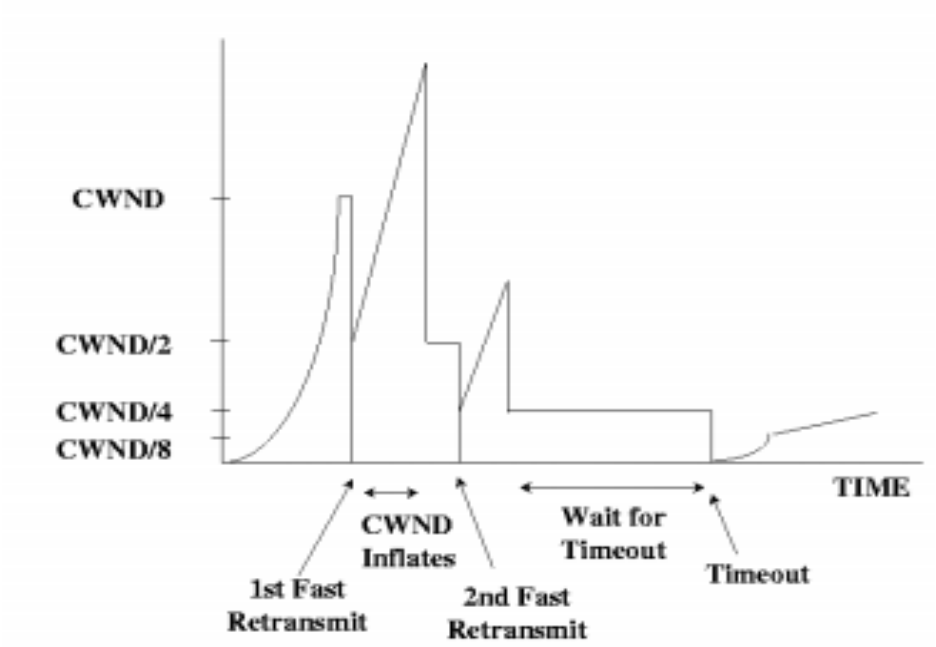
When a TCP destination receives an out-of-order segment, it immediately sends a duplicate acknowledgment to the sender. When the sender receives three duplicate ACKs, it concludes that the segment indicated by the ACKs has been lost, and immediately retransmits the lost segment. This is called "Fast Retransmit." Thus, congestion is sensed by the receipt of three duplicate ACKs.

The sender then reduces its CWND by half (plus 3 segments) and also saves half the original CWND value in SSTHRESH (the estimate of the new steady state). Now for each subsequent duplicate ACK, the sender inflates CWND by one and tries to send a new segment. Effectively, the sender waits for half a round trip before sending one segment for each subsequent duplicate ACK it receives. As a result, during congestion, the sender maintains the network pipe at half of its capacity at the time of fast retransmit.

Approximately one round trip after the missing segment is retransmitted, its ACK is received (assuming the retransmitted segment was not lost). Reno uses this ACK as an indication of the end of the congestion episode.

At this time, instead of setting CWND to one segment and proceeding to do slow start, the TCP sets CWND to SSTHRESH and then does congestion avoidance. This is called "Fast Recovery." The estimate for the new steady steady state is the same as before, but by doing fast recovery, slow start to get to the steady state is avoided.

When a single segment is lost from a window, Reno TCP recovers within approximately one RTT of knowing about the loss or two RTTs after the lost packet was first sent. The sender receives three duplicate ACKS one RTT after the dropped packet was sent. It then retransmits the lost packet. For the next round trip, the sender receives duplicate ACKs for the whole window of packets sent after the lost



TCP fast retransmit and recovery cannot recover efficiently from multiple packet losses. In this example, three back to back packets are lost. After the second fast retransmit, there are no outstanding packets to trigger duplicate ACKs. This results in timeout at a very low window, and the ensuing congestion avoidance phase is very slow especially in long delay networks.

Figure 2.6: TCP Reno: Fast retransmit and recovery

packet. The sender waits for half the window and then transmits a half window worth of new packets. All of this takes one RTT after which the sender receives a new ACK acknowledging the retransmitted packet and the entire window sent before the retransmission. CWND is set to half its original value and congestion avoidance is performed.

2.4.4 The Fast Retransmit Phase: TCP New Reno

In high bandwidth links, network congestion typically results in several dropped segments during a single congestion episode. In this case, fast retransmit and recovery

are not able to recover from the loss and slow start is triggered. Figure 2.6 shows a case when three consecutive packets are lost from a window and the sender TCP incurs fast retransmit twice and then times out. At that time, SSTHRESH is set to one-eighth of the original congestion window value (CWND in the figure) As a result, the exponential phase lasts a very short time, and the linear increase begins at a very small window. Thus, the TCP sends at a very low rate and loses much throughput.

The figure shows a case when fast retransmit and recovery loses its self clocking property and results in the wrong SSTHRESH estimate at the end of the congestion episode. This phenomenon has also been observed in [48].

Moreover, Hoe [48] points out that in some cases, retransmission of packets cached in the receiver's reassembly queue result in false retransmits. In this case, the sender goes into congestion avoidance mode when there is no congestion in the network. As a result, fast retransmit and recovery are effective only in isolated packet losses.

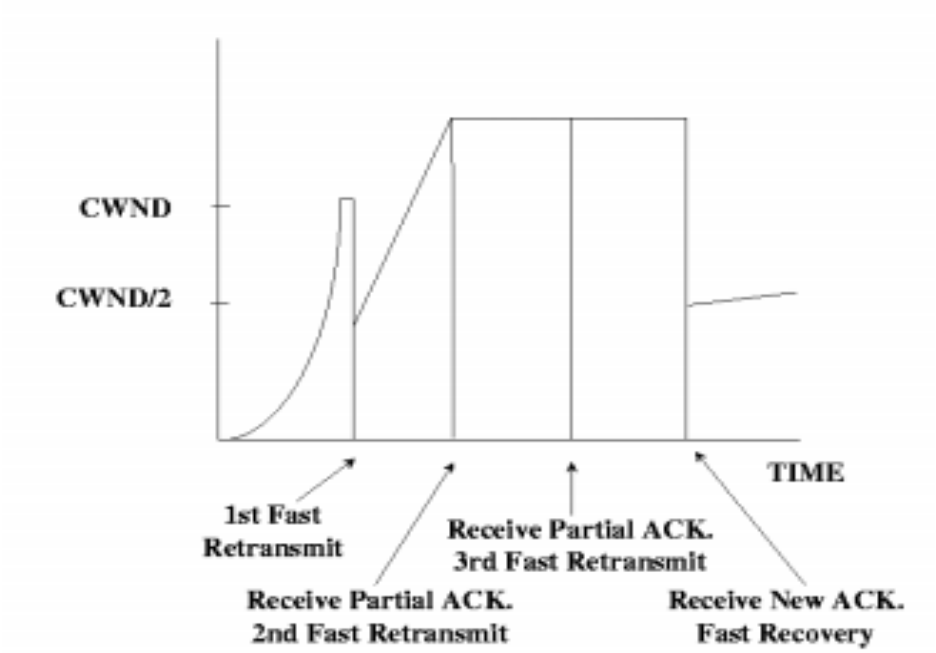
New Reno changes the following mechanisms in Reno these problems:

1. What to do during a congestion episode?

2. How to sense the end of a congestion episode?

3. How to recover from packet loss?

In New Reno, the *fast-retransmit phase* is introduced, in which the sender remembers the highest sequence number sent (RECOVER) when the fast retransmit is first triggered. The fast retransmit phase corresponds to the sender's estimate of the congestion episode. After the first unacknowledged packet is retransmitted (when three duplicate ACKs are received), the sender follows the usual fast recovery algorithm and inflates the CWND by one for each duplicate ACK it receives. When the sender



TCP Reno with the fast retransmit phase can retransmit one lost packet every round trip time. Timeout does not occur, and the congestion avoidance phase starts when all lost packets are retransmitted and ACKed.

Figure 2.7: TCP New Reno: The fast retransmit phase

receives an acknowledgment for the retransmitted packet, it checks if the ACK acknowledges all segments including RECOVER. If so, the ACK is a *new* ACK, and the sender exits the fast retransmit-recovery phase (the congestion episode has ended), sets its CWND to SSTHRESH and starts a linear increase (congestion avoidance).

If on the other hand, the ACK is a *partial* ACK, i.e., it acknowledges the retransmitted segment and only a part of the segments before RECOVER, then the sender immediately retransmits the next expected segment as indicated by the ACK. This continues until all segments including RECOVER are acknowledged. **This mechanism ensures that the sender recovers from N segment losses in N round trips.** This is called "New Reno."

As a result, the sender can recover from multiple packet losses without having to time out. In case of small propagation delays and coarse timer granularities, this mechanism can effectively improve TCP throughput over vanilla TCP. Figure 2.7 shows the congestion window graph of a TCP connection for three contiguous segment losses. The TCP retransmits one segment every round trip time (shown by the CWND going down to 1 segment) until a new ACK is received.

In our implementation, we have combined "New Reno" and SACK TCP as described in the following subsection.

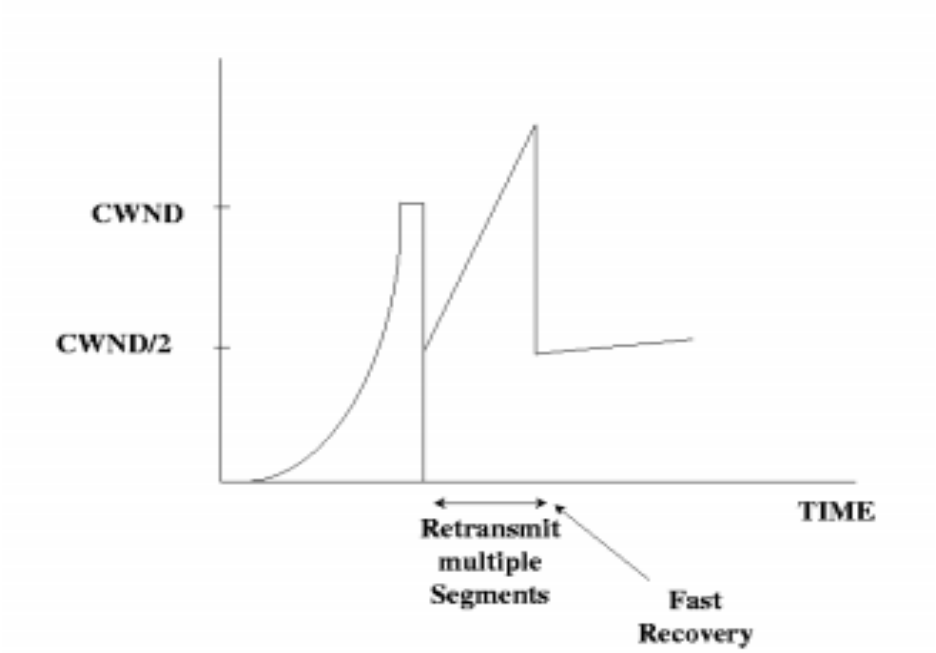
2.4.5 Selective Acknowledgments: TCP SACK

TCP with Selective Acknowledgments (SACK TCP) has been proposed to efficiently recover from multiple segment losses [74].

The SACK specification only changes the following behavior in TCP:

- How to recover from packet loss?

In SACK TCP, acknowledgments contain additional information about the segments that have been received by the destination. When the destination receives out-of-order segments, it sends duplicate ACKs (SACKs) acknowledging the out-of-order segments it has received. From these SACKs, the sending TCP can reconstruct information about the segments not received at the destination. When the sender receives three duplicate ACKs, it retransmits the first lost segment and inflates its CWND by one for each duplicate ACK it receives. This behavior is the same as Reno TCP and New Reno TCP. However, when the sender's CWND allows the TCP to send a segment in response to duplicate ACKs, the TCP uses the SACK information to retransmit lost segments before sending new segments. As a result, the sender can



TCP SACK uses selective acknowledgments to inform the sender of the packets received. During the fast retransmit phase, the sender retransmits missing packets before sending any new packets. TCP SACK can recover from multiple packet losses within a single round trip time.

Figure 2.8: TCP SACK: Selective acknowledgments

recover from multiple dropped segments in about one round trip. Figure 2.8 shows the congestion window graph of a SACK TCP recovering from segment losses. During the time when the congestion window is inflating (after fast retransmit has incurred), the TCP is sending missing packets before any new packets.

The SACK option is negotiated in the SYN segments during TCP connection establishment. The SACK information is sent with an ACK by the data receiver to the data sender to inform the sender of the out-of-sequence segments received. The format of the SACK packet has been proposed in [74]. The SACK option is sent whenever out of sequence data is received. All duplicate ACK's contain the SACK

option. The option contains a list of some of the contiguous blocks of data already received by the receiver. Each data block is identified by the sequence number of the first byte in the block (the left edge of the block) and the sequence number of the byte immediately after the last byte of the block. Because of the limit on the maximum TCP header size, at most three SACK blocks can be specified in one SACK packet.

The receiver keeps track of all the out-of-sequence data blocks received. When the receiver generates a SACK, the first SACK block specifies the block of data formed by the most recently received data segment. This ensures that the receiver provides the most up to date information to the sender. After the first SACK block, the remaining blocks can be filled in any order, but the receiver should try to include as many distinct blocks as possible.

The sender keeps a table of all the segments sent but not ACKed. When a segment is sent, it is entered into the table. When the sender receives an ACK with the SACK option, it marks all the segments specified in the SACK option blocks as SACKed. The entries for each segment remain in the table until the segment is ACKed. The remaining behavior of the sender is very similar to Reno implementations with the New Reno modification described in section 2.4.4. When the sender receives three duplicate ACKs, it retransmits the first unacknowledged packet. During the fast retransmit phase, when the sender is sending one segment for each duplicate ACK received, it first tries to retransmit the holes in the SACK blocks before sending any new segments. When the sender retransmits a segment, it marks the segment as retransmitted in the table. If a retransmitted segment is lost, the sender times out and performs slow start. When a timeout occurs, the sender resets the SACK table.

During the fast retransmit phase, the sender maintains a variable PIPE that indicates how many bytes are currently in the network pipe. When the third duplicate ACK is received, PIPE is set to the value of CWND and CWND is reduced by half. For every subsequent duplicate ACK received, PIPE is decremented by one segment because the ACK denotes a packet leaving the pipe. The sender sends data (new or retransmitted) only when PIPE is less than CWND. This implementation is equivalent to inflating the CWND by one segment for every duplicate ACK and sending segments if the number of unacknowledged bytes is less than the congestion window value.

When a segment is sent, PIPE is incremented by one. When a partial ACK is received, PIPE is decremented by two. The first decrement is performed because the partial ACK represents a retransmitted segment leaving the pipe. The second decrement is done because the original segment that was lost and had not been accounted for, is now actually considered to be lost.

2.4.6 Other TCP Implementations

In this section, we briefly describe other proposed TCP versions proposed in recent literature. These are non-standard implementations and are being studied by various research groups. The purpose of listing them here is to provide the reader with an overview of the possible changes to TCP in the future.

TCP Tahoe

TCP Tahoe is a precursor to TCP Reno. The only difference between TCP Tahoe and Reno is how they reach the new steady state at the end of a congestion episode. In both Reno and Tahoe, fast retransmit is triggered on the receipt of three duplicate

ACKS. However, when a partial or new ACK is received, TCP Tahoe sets its CWND to 1 segment. As a result, Tahoe enters the slow start phase until CWND reaches SSTHRESH. Recall that in TCP Reno, when a partial or new ACK was received, the CWND was set to SSTHRESH causing the system to immediately enter the congestion avoidance phase.

TCP Vegas

TCP Vegas [19] uses three techniques to improve TCP throughput and reduce packet loss. These three changes correspond to the following three mechanisms:

- How to sense congestion?
- How to estimate the new steady state?
- How to quickly reach the new steady state?

First, Vegas enhances the TCP RTT estimation algorithm by using the system clock instead of the coarse granularity TCP timer. Vegas uses the more accurate RTT estimate to sense congestion and can retransmit a packet even before the receipt of three duplicate ACKs. Specifically, certain ACKs are used to check for timeout and retransmission using the enhanced RTT estimate.

Second, Vegas detects changes in throughput by comparing measured throughput to expected throughput calculated using the window size. The congestion avoidance algorithm uses this information to maintain the optimal amount of data in the network. Thus, the steady state estimation is continually performed using throughput as an indicator.

Finally, Vegas modifies the slow start algorithm by increasing the window exponentially every other round trip and keeping it constant during the other roundtrip. During the constant phase, the algorithm compares the estimated and achieved throughputs. Based on the difference, it makes a decision to enter the increase or decrease mode. Again, the steady state estimation is continually performed. It takes longer to get to steady state, but the steady state estimation is more accurate than blindly setting SSTHRESH.

TCP Vegas-AFR: Vegas with Adaptive Fast Retransmit

Aron et. al. [6] present an enhancement to Vegas that changes the packet recovery behavior during a congestion episode. Vegas-AFR observes the frequency of consecutive packet losses. When consecutive losses occur, the TCP enters a state in which for every packet loss detected, it retransmits the lost packet and its successor. When an isolated packet loss is detected in this state, the TCP reverts back to its original state where it retransmits one segment per loss detected. Other features of this TCP are the same as TCP Vegas.

TCP Boston

TCP Boston [12] uses an efficient encoding technique to make TCP more resilient to fragmentation. The encoding technique adds redundancy to the fragments so that only m fragments are needed to construct a packet of N ($m < N$) fragments. Boston makes no changes to the congestion control mechanisms. This feature of TCP can be used with any congestion control scheme such as Reno, Tahoe, SACK or Vegas.

Forward Acknowledgment: FACK

The FACK algorithm for congestion control [76] uses the TCP SACK option for recovery from packet loss. The SACK algorithm described in section 2.4.5 is based on the fast retransmit and recovery algorithm. This algorithm uses duplicate ACKs to estimate the amount of data in the network pipe. FACK defines a new TCP state variable at the sender called *snd.fack* that reflects the forward-most data byte that is held by the receiver. When TCP is in its normal sending state, *snd.fack* is the same as the last unacknowledged packet in the sender's state. When the sender receives duplicate ACKs (or partial ACKs) with SACK information, *snd.fack* is updated to reflect the highest sequence number received by the destination.

Retransmission is triggered not only on the receipt of three duplicate ACKs, but also when the sender has more than three out of orders segments in its reassembly queue. Recovery ends in similar fashion as the SACK algorithm described before. The FACK proposal attempts to change the mechanisms for sensing the onset of congestion, as well as the end of congestion. Preliminary simulation results in [76] have shown that FACK can further improve TCP performance over the SACK algorithm described in the previous section.

TCP Net Reno

Lin and Kung [71] propose a network-sensitive version of Reno called Net Reno that improves performance by reducing retransmission timeouts. The paper observes that 85% of the timeouts in current TCP implementations on the Internet are because windows are too small for fast retransmission to trigger. In Net Reno, a new packet is sent for each duplicate ACK received before fast retransmission is triggered. In case

of small windows, these new packets trigger more duplicate ACKs that trigger fast retransmission.

TCP for Transactions (T/TCP)

T/TCP proposes TCP extensions for transactions [17], which allows TCP to bypass the three-way handshake during connection establishment. T/TCP allows the sender to send data with the connection open message (SYN). T/TCP improves TCP response time for short lived connections especially over long delay paths, because time is not wasted in the connection establishment phase. This extension does not effect any features of the TCP congestion control mechanisms.

Table 2.1 lists the enhancements made by each of the above TCP flavors to the nine goals given above.

2.4.7 Miscellaneous TCP Features

In this section, we describe other features and enhancements to various parts of the TCP protocol. Some of these are standard features and we use them in our simulations. Others are proposed enhancements that are being studied by the research community.

Delayed Acknowledgments

In most implementations, when a TCP receives a packet, its default behavior is to wait for a DELAY_ACK timer to expire before sending an acknowledgment [93]. This timer is typically 200 ms long. As a result, the sending of an ACK can be delayed by

TCP Type	Modification to Goal								
	1	2	3	4	5	6	7	8	9
Reno				X	X	X	X		X
New Reno				X	X	X	X		X
SACK						X			
Tahoe				X	X	X	X		
Vegas				X			X		X
Vegas-AFR				X		X	X		X
FACK				X	X	X	X		X
Net Reno				X	X	X	X		X
Startup dynamics	X								
Boston	No change to congestion control								
T/TCP	No change to congestion control								

Goals: 1 = initial steady state estimation, 2 = reaching initial steady state quickly, 3 = what to do during steady state, 4 = sensing congestion, 5 = reacting to congestion, 6 = recovering from packet loss, 7 = sensing the end of congestion, 8 = estimating the new steady state, and 9 = quickly reaching the new steady state after congestion. A check mark implies that the TCP flavor in the row modifies the corresponding technique used by TCP vanilla.

Table 2.1: Summary of TCP flavors

as much as 200 ms after receiving a packet. In case of two way traffic, this feature gives the receiver some time to wait for data from its application so that the ACK information can be piggybacked onto the data packet. However, the receiving TCP cannot have more than one outstanding ACK while waiting for the DELAY_ACK timer to expire. If another packet is received while waiting for the timer, an ACK acknowledging both packets is immediately sent out. Also, when an out of sequence packet is received, the receiver must send a duplicate ACK immediately.

Note, that the sender TCP changes its CWND based on the number of ACKs received, and not on the amount of data acknowledged. Due to the DELAY_ACK timer, the CWND window doubles in two round trips during the slow start phase and increase by one in two round trips during the congestion avoidance phase. Delayed

acknowledgments can be turned off at the receiver. Another proposal suggests that delayed ACKs not be used during the slow start phase [3] so that the window can increase aggressively during this time.

Path MTU Discovery

The TCP Maximum Segment Size (MSS) determines the maximum size of TCP packets sent on the network. If the MSS is greater than the MTU of the underlying link layer protocol anywhere in the path of the packet, the TCP segment is fragmented to fit the MTU. Fragmentation is undesirable in TCP because the loss of a single fragment results in the retransmission of the entire TCP packet. The path MTU discovery protocol [93] in TCP probes the network for the minimum MTU size in the TCP connection's path, and sets the MSS to be smaller than the MTU.

Silly Window Avoidance

When small amounts of data are exchanged on a TCP connection, the receiver may advertise small increases in its window size. As a result, the sender is forced to send data in small segments (less than MSS), leading to increased overhead due to headers. Silly window avoidance [93] can be implemented in both sender and receiver to prevent this phenomenon. The sender tries to send full segments whenever possible and the receiver does not advertise increases in its window that are smaller than MSS. Silly window avoidance, also known as Nagle's algorithm, is an option that can be set in typical TCP implementations.

TCP Window Scaling

The standard TCP maximum window size is limited by the size of the TCP packet header field (32 bits) to 65,535 bytes. In long delay-bandwidth environments, the default maximum TCP window size is not sufficient to fill the network capacity. The window scaling option [51] can be used to increase the maximum default window size. The window scaling option is a 16 bit field called SCALE that is used to increase the window as follows:

$$\text{SCALED\_WND} = \text{WND} \times 2^{\text{SCALE}}$$

The scaling option is typically used in long delay-bandwidth environments such as satellite networks.

Round Trip Time Measurement (RTTM)

The RTTM option [51] allows the TCP sender to obtain more accurate estimates of the round trip time (RTT) of the connection. The option allows the sender to place a timestamp in every segment. The receiver copies this value in the ACK, so that the sender can calculate an accurate RTT for each ACK. Details on the RTT calculations can be found in [51].

Protection Against Wrapped Sequence Numbers (PAWS)

In large delay-bandwidth networks, the traditional 32-bit TCP sequence number space may not be sufficient to count the number of outstanding segments in the network. Wrapping of sequence numbers can cause different segments with the same sequence number to be present in the network. The PAWS option [51] uses TCP timestamps

from the RTTM mechanism and assumes that the timestamps in packets are monotonically non-decreasing in time. A segment is considered a duplicate if its timestamp is less than that some timestamp "recently" received on the connection. Typically, this "recent" timestamp is the timestamp of the last insequence packet received on the connection.

Improving Startup Dynamics

The initial value of SSTHRESH is set to the receiver's advertised window. As a result, the slow start phase continues until CWND reaches the receiver's window size. Hoe [48] recommends the use of bandwidth estimation techniques to determine a value for SSTHRESH that reflects the current network capacity (steady state). In this way, slow start can be terminated when the window size approximately equals the bandwidth-delay product of the network. This avoids bursty losses during the slow start phase because of overshooting the network capacity.

Larger Initial Window

A TCP connection starts with an initial window size of 1 segment. A larger initial window of up to four segments (or 4 k bytes, whichever is smaller) has been proposed in [5]. As a result, more segments can be sent during the first round trip time, triggering more ACKs and allowing the window to increase faster. This is especially useful in long delay environments.

Byte Counting

This feature tries to overcome the lost throughput due to the delayed acknowledgment option. In byte counting, on receiving an ACK, the sending TCP increments its congestion window by the amount of data ACKed [3]. However byte counting may increase the burstiness of TCP traffic and the loss rate on some networks.

Explicit Congestion Notification (ECN)

TCP uses packet loss as a signal to detect congestion. Network elements that do not implement congestion avoidance techniques such as Random Early Detection (RED) drop packets when their buffers become full. In many cases, it is desirable to detect congestion before its onset. ECN allows network elements to notify the TCP of mild or impending congestion in the network, so that TCP can reduce its window. Ramakrishnan and Floyd, [83] propose the use of two bits in the IP header for ECN purposes. During mild congestion, network elements can set these bits in TCP packets. The receiver TCP accumulates congestion information received in these bits and sends an ACK with an ECN signal to the sender. The sender then reduces its window appropriately.

In another ECN option called Backward Explicit Congestion Notification (BECN), a router can send ICMP source quench messages to the source. On receiving this message, the source reduces its congestion window.

Constant Rate TCP

Floyd [31], illustrates the performance degradation of TCP connections that traverse multiple congested gateways. This leads to higher throughput in TCPs traversing a

fewer number of hops. The paper suggests the use of a more aggressive congestion avoidance algorithm based on equalizing the rate at which all TCPs send data. Simulations are used to show that constant rate TCP does not have a bias against multiple congested gateways.

Increase by K during Congestion Avoidance

During the congestion avoidance phase, CWND increases by one segment every round trip time. This increase is very inefficient for long delay networks. Henderson [46] suggests an increase-by-k congestion avoidance algorithm for long delay connections, so that the window increases by k segments every RTT. Experimental values for k are presented in [46].

TCP Performance Enhancing Proxies (PEPs)

These are better known as TCP spoofing gateways (or proxies), because they break the end-to-end TCP connection into parts. PEPs are typically deployed over specific link layers to improve TCP performance over these layers. For example, a pair of TCP PEPs across a satellite or wireless link can split an end-to-end TCP connection with the satellite link in its path, into three connections – two connections, between the source TCP and the proxy, and the destination TCP with the other proxy, and one connection between the proxies over the satellite link. The protocol running over the satellite link may be an enhanced version of TCP or even a different transport protocol. The end-system TCP is typically unaware of the existence of the proxy.

ACK Spacing

Bursty TCP traffic patterns are caused by sudden increases in the sender's window when a burst of ACKs is received. Bursty traffic can cause buffers in the network to overflow, causing retransmission and congestion. ACK Spacing is used to reduce burstiness in the sender TCP by sending ACKs at regular intervals instead of bursts. Partridge [81] recommends using a distance of at least two segments between ACKs to avoid burstiness in sender traffic.

TCP ACK Control

This enhancement proposes to reduce the rate of ACKs when they may cause reverse channel congestion on asymmetric links. Balakrishnan et. al. [8] proposes that routers should mark reverse direction congestion in the ECN fields of the ACK packets. The sender echos ECN information to the receiver. On receiving congestion notification, the receiver reduces its ACK rate.

ACK Filtering

ACK filtering controls the ACK rate without requiring changes in TCP. TCP ACKs carry cumulative acknowledgment information, as a result of which some ACKs may be redundant. In this scheme, routers drop redundant ACKs from their queues during congestion [8].

ACK Regulation

Ack regulation techniques require routers to control the flow of TCP acknowledgments based on network feedback [88]. These techniques can be applied to ATM edge devices

or to Internet routers. ACK regulation techniques can considerably improve fairness, throughput and end-to-end delay properties of TCP applications.

2.5 A Survey of Buffer Management

Buffer management schemes control the number of packets from each flow in the output buffers. In the case of ATM, buffer management schemes decide whether an arriving cell for a VC should be enqueued in the buffer, or discarded. In other words, buffer management determines the policy for buffer sharing among the data flows. When a packet is received on an output port, a queuing algorithm selects the appropriate queue for the packet, and then the buffer management algorithm for that queue either enqueues the packet or discards it. Each queue may contain packets from one or more VCs.

In this section, we present a framework for buffer management schemes and describe existing buffer management schemes in the context of this framework.

2.5.1 A Framework for Buffer Management

Recent research has focussed on fair buffer management for network traffic. In these proposals, packets are dropped when the buffer occupancy exceeds a certain threshold. The proposals for buffer management can be classified based on several factors. A buffer management scheme consists of two main components:

1. The buffer partitioning policy

2. The packet discard function

The buffer partitioning policy

The buffer partitioning policy is a specification of a set of rules or heuristics that

determine buffer utilization and partitioning behavior. This outlines how the buffers are shared among the packets of different VCs or groups of VCs. A Complete Sharing policy [63] does not distinguish between packets of different VCs, while a Complete Partitioning policy [63] statically allocates buffer space for the packets of different VCs. When a packet arrives, the buffer partitioning policy is used to determine if the packet meets the partitioning criteria for the buffer. Buffer partitioning policies can be classified based on the two factors listed below:

- **Accounting (Single versus Multiple)**: Schemes using single accounting (SA) maintain a single count of the number of cells currently in the buffer. The multiple accounting (MA) schemes classify the traffic into several flows and maintain a separate count for the number of cells in the buffer for each flow. Typically, each flow corresponds to a single connection (or VC) and these schemes maintain per-connection (or per-VC) occupancies. In cases where the number of connections far exceeds the buffer size, the added overhead of per-connection accounting may be very expensive. In this case, a set of active connections is defined as those connections with at least one packet in the buffer and only the buffer occupancies of active connections are maintained.

- **Number of Thresholds (Single versus Multiple)**: Schemes with a global threshold (ST) compare the buffer occupancy(s) with a single threshold and drop packets when the buffer occupancy exceeds the threshold. Multiple thresholds (MT) can be maintained corresponding to classes, connections or to provide differentiated services. For example, some schemes may differentiate packets based on packet tags. Examples of packet tags are the Cell Loss Priority (CLP)

bit in ATM cells or the Differentiated Services Code Point (DSCP) field in the IP header of the IETF's differentiated services architecture.

The packet discard function

This function determines if a packet is accepted or discarded depending on the outcome of the buffer partitioning policy and other factors such as the nature of the drop and congestion in the queue. Packet discard functions depend on the following factors:

- **Threshold Calculation (Static versus dynamic)**: Static threshold schemes compare the buffer occupancy with a fixed threshold determined by the buffer partitioning policy. In dynamic threshold schemes, thresholds may vary based on the number of active connections, total queue size and per-connection queue size.

- **Buffer occupancy calculation (Instantaneous versus average queues)**: Some schemes use weighted exponential averages to calculate buffer occupancy levels, whereas others simply rely on instantaneous queue lengths.

- **Drop behavior (Deterministic versus probabilistic)**: Deterministic drop schemes drop packets every time the discard conditions are met, whereas probabilistic drop schemes use a random variable to determine the drop decision.

- **Target packets (Pushout versus incoming)**: When a decision is made to drop a certain category of packets, the scheme can drop incoming packets belonging to that category, or may drop packets of that category already in the queue. The latter schemes are called *pushout* schemes. Pushout schemes

typically require separate queues for each flow and are thus harder to implement at high speeds.

Based on the partitioning policy, buffer management schemes are divided into the following four groups:

- Single Accounting - Single Threshold (SA-ST)
- Single Accounting - Multiple Threshold (SA-MT)
- Multiple Accounting - Single Threshold (MA-ST)
- Multiple Accounting - Multiple Threshold (MA-MT)

Table 2.2 lists the four classes of buffer management schemes and examples of schemes for these classes. The example schemes are discussed below.

Group	Examples	Threshold (Static / Dynamic)	Drop Type (Deterministic / Probabilistic)	Tag/TOS Sensitive (Yes/No)
SA-ST	EPD, PPD	Static	Deterministic	No
	RED	Static	Probabilistic	No
MA-ST	FRED	Dynamic	Probabilistic	No
	SD, FBA	Dynamic	Deterministic	No
	VQ+Dynamic EPD	Dynamic	Deterministic	No
MA-MT	PME+ERED	Static	Probabilistic	Yes
	DFBA	Dynamic	Probabilistic	Yes
	VQ+MCR scheduling	Dynamic	Deterministic	No
SA-MT	Priority Drop	Static	Deterministic	Yes

Buffer management schemes can be classified into four categories depending on the number of thresholds and the state information used by the schemes.

Table 2.2: Classification of buffer management

2.5.2 SA-ST schemes

Tail Drop

Tail drop switches allow complete sharing of buffer space among the connections. When buffers become full, incoming packets are dropped. Tail drop ATM switches drop *cells* when the buffers become full. Tail drop in ATM may result in some cells of a packet being dropped while other cells may be enqueued and forwarded. Floyd [33] shows that the tail drop scheme causes poor TCP performance because of phase effects in bursty TCP traffic.

Random Drop

The Random Drop scheme is similar to the tail drop scheme in that it drops a packet when the buffer becomes full. However, unlike tail drop, this scheme randomly selects a packet in the queue and drops the packet. For a buffer size of P packets, Random Drop picks a random number r between 1 and P and drops the rth packet from the queue. The incoming packet that arrived on the full queue can now be accepted. The goal of random drop is to drop packets from a connection in proportion to the connection's share of the total throughput. However [33, 44] illustrate that random drop does not achieve this goal and results in low throughput and fairness among competing connections. In these papers, the Early Random Drop scheme has been suggested that randomly drops a packet when the total queue length exceeds a certain threshold. Early Random Drop has since evolved into Random Early Detection (RED) discussed below.

Early Packet Discard (EPD) and Partial Packet Discard (PPD)

EPD and PPD [87] were designed to avoid the transmission of partial packets in an ATM network. Both EPD and PPD use the EOM cell information to distinguish end of frame boundaries. In the PPD scheme, when a cell is dropped due to buffer overflow, the remaining cells from the frame are also dropped. EPD drops complete packets instead of partial packets. As a result, the link does not carry incomplete packets which would have been discarded during reassembly. In EPD, a threshold R less than the buffer size is set in the buffer. When the switch queue length exceeds this threshold, all cells from any new packets are dropped. Packets which had been partly received before exceeding the threshold are still accepted if there is buffer space.

In chapter 4, we show that EPD improves performance because it minimizes the transmission of partial packets by the network. Since EPD does not discriminate between connections in dropping packets, the scheme is unfair in allocating bandwidth to competing connections. For example, when the buffer occupancy reaches the EPD threshold, the next incoming packet is dropped even if the packet belongs to a connection that is using less than its fair share of the buffer.

Random Early Detection (RED)

Random Early Detection (RED) [28] maintains two global thresholds min_{th} and max_{th} and calculates buffer occupancy using an exponentially averaged queue length q_{ave}. When q_{ave} exceeds min_{th}, RED drops incoming packets probabilistically using a uniform random variable as the drop probability. The basis for this is that uniform dropping drops packets in proportion to the input rates of the connections. Connections with higher input rates lose proportionally more packets than connections with

lower input rates. In this way, RED tries to maintain equal rate allocation. When q_{ave} exceeds max_{th}, RED deterministically drops all arriving packets until q_{ave} reduces to below max_{th}.

Simulation results have shown that RED eliminates biases against bursty traffic shown by tail drop and random drop. However, TCP connections with a longer round trip time are not treated fairly in RED. The designers of RED attribute that to the TCP congestion control algorithms that are inherently dependent on the round trip time of the connection.

Weighted RED

W-RED is a commercial implementation of RED [94] that distinguishes packets with different priorities. In this scheme, hosts or edge routers may add precedence values to packets as they enter the network. In the interior of the network, when the RED threshold is reached, higher priority packets are dropped with a lower probability than lower priority packets. In this way, W-RED can be used to provide priority based quality of service to data flows.

2.5.3 MA-ST Schemes

Flow Random Early Detection (FRED)

Lin and Morris [72] show that RED like proportional dropping cannot guarantee equal bandwidth sharing. The paper also contains a proposal for Flow Random Early Drop (FRED). FRED maintains per-connection buffer occupancies and drops packets probabilistically if the per-TCP connection occupancy exceeds the average queue length.

In addition, FRED ensures that each connection has at least a minimum number of packets in the queue. FRED ensures that each flow has roughly the same number of packets *in the buffer* and First Come First Serve (FCFS) scheduling guarantees equal sharing of bandwidth. FRED can be classified as one that maintains per-connection queue lengths, but has a global threshold (MA-ST).

Fair Buffer Allocation (FBA)

The Fair Buffer Allocation (FBA) [45] scheme is an MA-ST scheme proposed for the ATM UBR service category. This scheme uses per-VC accounting to maintain the current buffer utilization of each UBR VC. A fair allocation is calculated for each VC and if the VC's buffer occupancy exceeds its fair allocation, its subsequent incoming packet is dropped. A threshold, R, is maintained as a fraction of the buffer capacity K. When the total buffer occupancy exceeds R×K, new packets are dropped depending on VC_i's buffer occupancy (Y_i). A VC's *entire packet* is dropped if

$$(X > R) \text{ AND } (Y_i \times N_a/X > Z \times ((K-R)/(X-R)))$$

where N_a is the number of active VCs (VCs with at least one cell the buffer), and Z is another threshold parameter ($0 < Z \leq 1$) used to scale the effective drop threshold. Performance studies of FBA have not been reported literature. We present a performance analysis of FBA in chapter 4.

Virtual Queuing (VQ)

The Virtual Queuing (VQ) [101] scheme is unique because it achieves fair buffer allocation by emulating on a single FIFO queue, a per-VC queued round-robin server. At each cell transmit time, a per-VC accounting variable ($\hat{Y}_i$) is decremented in a

round-robin manner and is incremented whenever a cell of that VC is admitted in the buffer. When $\hat{Y_i}$ exceeds a fixed threshold, incoming packets of the ith VC are dropped. An enhancement called Dynamic EPD changes the above drop threshold to include only those sessions that are sending less than their fair shares.

Since the above MA-ST schemes compare the per-connection queue lengths (or virtual variables with equal weights) with a global threshold, they can only guarantee equal buffer occupancy (and thus throughput) to the competing connections. These schemes do not allow for specifying a guaranteed rate for connections or groups of connections. Moreover, in their present forms, they cannot support packet priority based on tagging.

2.5.4 MA-MT Schemes

Weighted FBA

The Weighted Fair Buffer Allocation [15] is an extension of the FBA scheme to support unequal buffer sharing by the packets. The paper also describes how W-FBA scheme can be applied to the GFR service category to provide minimum rate guarantees to VCs carrying TCP traffic. Each VC is assigned a weight based on its minimum cell rate (MCR_i). The scheme is the same as FBA, except that the drop condition is as follows:

$$(X > R) \text{ AND } (Y_i <> Z \times (\frac{MCR_i}{\sum_{j=i}^{j=N_a} MCR_j}) \times ((K-R)/(X-R)))$$

The paper presents results to show that W-FBA can be used for the Guaranteed Frame Rate service category. Chapter 6 presents a scheme which we developed independently in this research that achieves the same goals.

Virtual Queuing with MCR Scheduling

Another enhancement to VQ, called MCR scheduling [90], proposes the emulation of a weighted scheduler to provide Minimum Cell Rate (MCR) guarantees to ATM connections. In this scheme, a per-VC weighted variable (W_i) is maintained, and compared with a global threshold. A time interval T is selected, at the end of which, W_i is incremented by $\text{MCR}_i \times T$ for each VC i. The remaining algorithm is similar to VQ. As a result of this weighted update, MCRs can be guaranteed. However, the implementation of this scheme involves the update of W_i for each VC after every time T. To provide tight MCR bounds, a smaller value of T must be chosen, and this increases the complexity of the scheme. For best effort traffic (like UBR), thousands of VC could be sharing the buffer, and this dependence on the number of VCs is not an efficient solution to the buffer management problem. Since the variable W_i is updated differently for each VC i, this is equivalent to having different thresholds for each VC at the start of the interval. These thresholds are then updated in the opposite direction of W_i. As a result, VQ+MCR scheduling can is a MA-MT scheme.

Packet Marking Engine + Enhanced RED

Feng et. al. [20] propose a combination of a Packet Marking Engine (PME) and an Enhanced RED scheme based on per-connection accounting and multiple thresholds (MA-MT). PME+ERED is designed for the IETF's differentiated services architecture and provides loose rate guarantees to connections. The PME measures per-connection bandwidths and probabilistically marks packets if the measured bandwidths are lower than the target bandwidths (multiple thresholds). High priority packets are marked, and low priority packets are unmarked. The ERED mechanism

is similar to RED except that the probability of discarding marked packets is lower that that of discarding unmarked packets. The PME in a node calculates the observed bandwidth over an update interval, by counting the number of accepted packets of each connection by the node. Calculating bandwidth can be complex and may require averaging over several time intervals.

2.5.5 SA-MT Schemes

A simple SA-MT scheme can be designed that implements multiple thresholds based on the packet priorities. When the global queue length (single accounting) exceeds the first threshold, packets tagged as lowest priority are dropped. When the queue length exceeds the next threshold, packets from the lowest and the next priority are dropped. This process continues until EPD/PPD is performed on all packets. The performance of such schemes needs to be analyzed. However, these schemes cannot provide per-connection throughput guarantees and suffer from the same problem as EPD, because they do not differentiate between overloading and underloading connections. The double EPD scheme described next is an example of a simple SA-MT scheme.

Double EPD

The Double EPD scheme [15] proposed in [43] uses two drop thresholds (Low Buffer Occupancy (LBO) and High Buffer Occupancy (HBO)) to distinguish between high priority and low priority cells. HBO is basically an EPD threshold for the queue. When the queue length exceeds LBO, the incoming low priority frames (frames with

CLP=1 cells) are dropped. CLP=0 frames are accepted until the buffer occupancy reaches HBO.

Table 2.3 illustrates the fairness properties of the four buffer management groups presented above.

Group	Equal bandwidth allocation	Weighted bandwidth allocation
SA-ST	No	No
MA-ST	Yes	No
MA-MT	Yes	Yes
SA-MT	-	-

Table 2.3: Properties of the four categories of buffer management schemes.

2.5.6 Schemes that require per-VC Queuing

Pushout with Longest Queue Drop

Longest queue drop (LQD) is a pushout based scheme presented in [66]. The paper points out that pushout based schemes are hard to implement in ATM switches, but can be implemented in IP routers. IP routers generally queue relatively small packet descriptors and the additional memory bandwidth overhead is not significant. Among the pushout schemes, Drop From Front schemes that drop packets from the front of the queue enhance TCP performance because congestion information reaches sources quicker than drop from back schemes. Longest queue drop relies on per-flow queuing and drops the queue whose excess occupancy (total occupancy minus nominal occupancy) is the largest. Lakshman et. al. [66] show that LQD with drop from front

provides high TCP throughput when per-VC queuing is available.

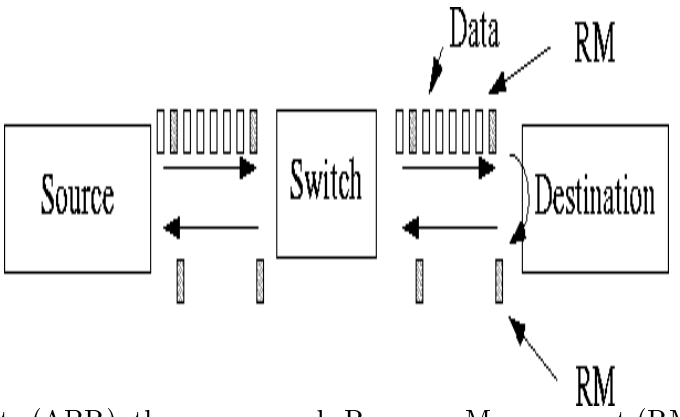
Quasi Pushout

The Quasi Pushout [25] scheme is a variation of Longest Queue Drop that maintains a variable for the Quasi-longest queue. This variable is updated during cell arrival and departure events. When a decision to discard is made, then the quasi-longest queue is dropped.

2.6 An Overview of ABR Feedback Control

ABR mechanisms allow the network to divide the available bandwidth fairly and efficiently among the active traffic sources. In the ABR traffic management framework, the source end systems limit their data transmission to rates allowed by the network. The network elements use their current load information to calculate the allowable rates for the sources. These rates are sent to the sources as feedback via resource management (RM) cells. RM cells are generated by the sources and travel along the data path to the destination end systems. The destinations simply return the RM cells to the sources. The components of the ABR traffic management framework are shown in Figure 2.9.

The ABR traffic management model is a *rate-based closed-loop* model. The model is rate-based because the sources send data at a specified rate. This is different from TCP where the control is window based and the sources limit their transmission to a particular number of packets. The ABR model is called closed-loop because there is a continuous flow of control cells between the network and the source. The model used



In Available Bit Rate (ABR), the source sends Resource Management (RM) cells that travel to the destination and back to the source. RM cells carry feedback from the network.

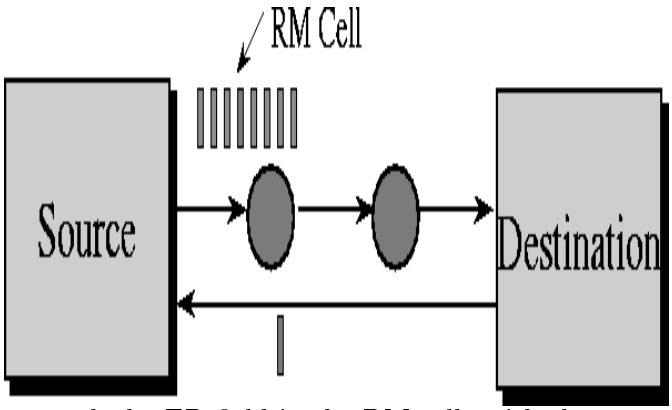
Figure 2.9: ABR traffic management

for UBR, on the other hand, is open-loop in the sense that no explicit feedback is provided to the hosts. The control loop may be end-to-end where the RM cells travel from the source to the destination and back to the source, or segment-by-segment using the virtual source/virtual destination (VS/VD) feature discussed in chapter 7.

There are three ways for switches to give feedback to the sources:

1. *Explicit Forward Congestion Indication.* Each cell header contains a bit called Explicit Forward Congestion Indication (EFCI), which can be set by a congested switch. Such switches are called *binary* or *EFCI* switches. The destination then aggregates these EFCI bits and returns feedback to the source in an RM cell. In the current specification, the RM cell is sent by the source periodically and is turned around by the destination with the bit-feedback.

2. *Relative rate marking.* RM cells have two bits in their payload, called the Congestion Indication (CI) bit and the No Increase (NI) bit, that can be set by



Explicit rate switches mark the ER field in the RM cells with the rate at which the source is allowed to send. On receiving RM cells, the source adjusts its rate accordingly.

Figure 2.10: Explicit rate feedback

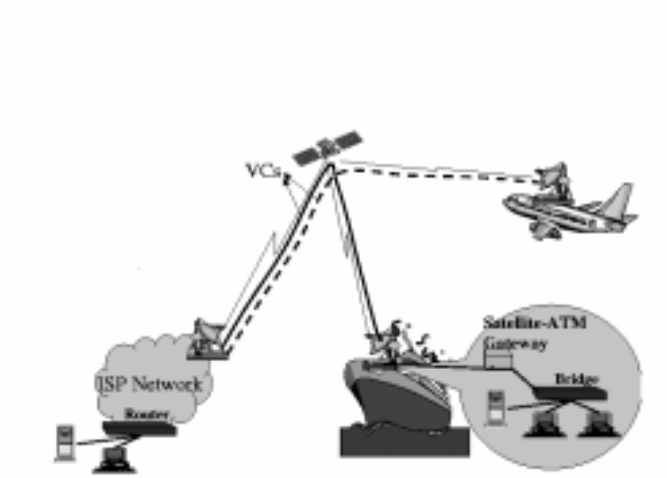
congested switches. Switches that use only this mechanism are called relative rate marking switches.

3. *Explicit Rate.* RM cells also have another field in their payload called explicit rate (ER) that can be set by congested switches to any desired value. Such switches are called explicit rate switches. The explicit rate mechanism is shown in figure 2.10.

Explicit rate switches normally wait for the arrival of an RM cell to give feedback to a source. However, under extreme congestion, they are allowed to generate an RM cell and send it immediately to the source. This optional mechanism is called backward explicit congestion notification (BECN).

2.7 An Overview of Satellite-ATM Networks

The rapid advances in ATM technology and Ka-Band satellite communications systems are leading to a vast array of opportunities for new value added services.

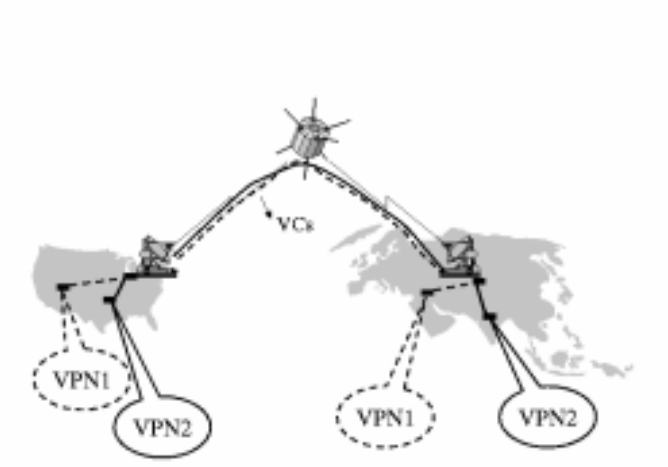


Low Earth Orbit (LEO) satellites can be used in access networks for remote locations not accessible by the fiber infrastructure.

Figure 2.11: LEO satellite in access networks

Examples of such services include interactive as well as distribution services such as video conferencing, transmission of audio/video and high resolution image documents. Current trends in satellite communications exhibit an increased emphasis on new services as opposed to point-to-point data communications. The new services gaining momentum include mobile services, direct broadcast, private networks and high-speed hybrid networks in which services would be carried via integrated satellite-fiber networks. To fully realize these integrated systems, it is essential that advanced network architectures be developed that seamlessly interoperate with existing standards, interfaces and higher layer protocols.

With the deployment of ATM technology, there is a need to provide interconnection of geographically dispersed ATM networks. Although ATM technology has



Geosynchronous (GEO) satellites can be used in backbone networks connecting two or more geographically dispersed parts of a network.

Figure 2.12: GEO satellite in backbone networks

been developed to provide an end-to-end transparent service over terrestrial networks, satellite-ATM systems will play a significant role in achieving global connectivity.

The growing interest in Satellite ATM networking is due to the several advantages offered by satellite communications technology [2, 36]. These include,

- Wide geographic coverage including interconnection of ATM islands
- Multipoint to multipoint communications facilitated by the inherent broadcasting ability of satellites
- Bandwidth on demand or Demand Assignment Multiple Access (DAMA) capabilities
- An alternative to fiber optic networks for disaster recovery options.

Figures 2.11 and 2.12 illustrate two typical deployments of satellites in access and backbone networks respectively.

However, satellite systems have several inherent constraints. The resources of the satellite communication network, especially the satellite and the earth station have a high cost and must be used efficiently. *A crucial issue is that of the high end-to-end propagation delay of satellite connections.* Apart from interoperability issues, several performance issues need to be addressed before a transport layer protocol like TCP can satisfactorily work over large delay-bandwidth networks. With an acknowledgment and timeout based congestion control mechanism (like TCP's), performance is inherently related to the delay-bandwidth product of the connection. As a result, the congestion control issues for broadband satellite networks are somewhat different from those of low latency terrestrial networks.

Figure 2.13 illustrates a satellite-ATM network model represented by a ground segment, a space segment and a network control segment. The ground segment consists of ATM networks which may be further connected to other legacy networks. The network control center (NCC) performs various management and resource allocation functions for the satellite media. Inter-satellite crosslinks in the space segment provide seamless global connectivity via the satellite constellation. The network allows the transmission of ATM cells over satellite, multiplexes and demultiplexes ATM cell streams for uplinks, downlinks and interfaces to interconnect ATM networks as well as legacy LANs.

Broadband switches should be able to multiplex thousands of transport connections that use ATM virtual circuits (VCs) for non-real time applications. On-board

satellite switches and switches at the earth stations fall into this category and are expected to multiplex a large number of non-real time transport connections over ATM virtual circuits. Figure 2.14 illustrates the protocol stack for Internet protocols over satellite-ATM. The satellite-ATM interface device separates the existing SONET and Physical Layer Convergence Protocol (PLCP) [2, 64].

2.8 A Queuing Architecture

In ATM networks, the service categories can be organized in the four priority setup shown in table 2.4. The table also informally lists the QoS requirements for each class. The CBR and the VBR-rt classes form the highest priority class with strict throughput, delay and jitter requirements. The VBR-nrt VCs only require throughput guarantees. ABR VCs are guaranteed an MCR. In addition, the network must allocate a fair share of unused bandwidth to ABR VCs. The definition of fairness is implementation dependent. GFR VCs are also guaranteed a minimum cell rate for conforming cells, while UBR VCs receive no guarantees.

Priority Level	Service Category	QoS Requirements
0	CBR	Throughput, delay, delay-variation
0	VBR-rt	Throughput, delay, delay-variation
1	VBR-nrt	Throughput
2	ABR	Throughput, fair share of excess
3	GFR	Throughput, fair share of excess
3	UBR	None

ATM service categories can be prioritized into four levels based on the requirements of each category.

Table 2.4: Priorities for ATM service categories

In our model, the network may allocate a minimum fraction of the link capacity collectively to each service category. For example, all ABR traffic could be guaranteed an aggregate long term throughput of 10 Mbps. The priority setup is enforced by a scheduling mechanism, where the connections within each priority level have equal scheduling priority as long as their rate allocations allow them to be scheduled. When connections from different priority levels compete to be scheduled, then the connection with the highest priority (lowest priority level) is scheduled first. When multiple connections from the same priority level are eligible to be scheduled, they have equal scheduling priority.

Figure 2.15 illustrates the queuing model presented here. We define a flow as a cell or packet stream that can be managed separately from other streams. For example, an ATM VC is a flow. In this model, each flow belongs to a class of service. Service classes are given a priority value. Multiple flows may belong a a single class of service (or service category) and multiple service categories may share the same priority. Each flow may have its own queue (per-VC queue) or may share a queue (class queue) with other flows from the same service class.

Per-flow queues drain into the class queue for their service class. The class queues drain into the link queue that drains at the link rate. Each queue has an allocated output rate that might change during the lifetime of the flow. A server at each queuing point monitors the flow of packets into the queue and the current number of packets in the queue. The server also controls the output rate of the flow. Some queues may have additional capabilities like providing feedback to other queues in the network. A buffer management function at the servers controls the occupancy of each queue. This function may discard packets in the queue based on congestion conditions and

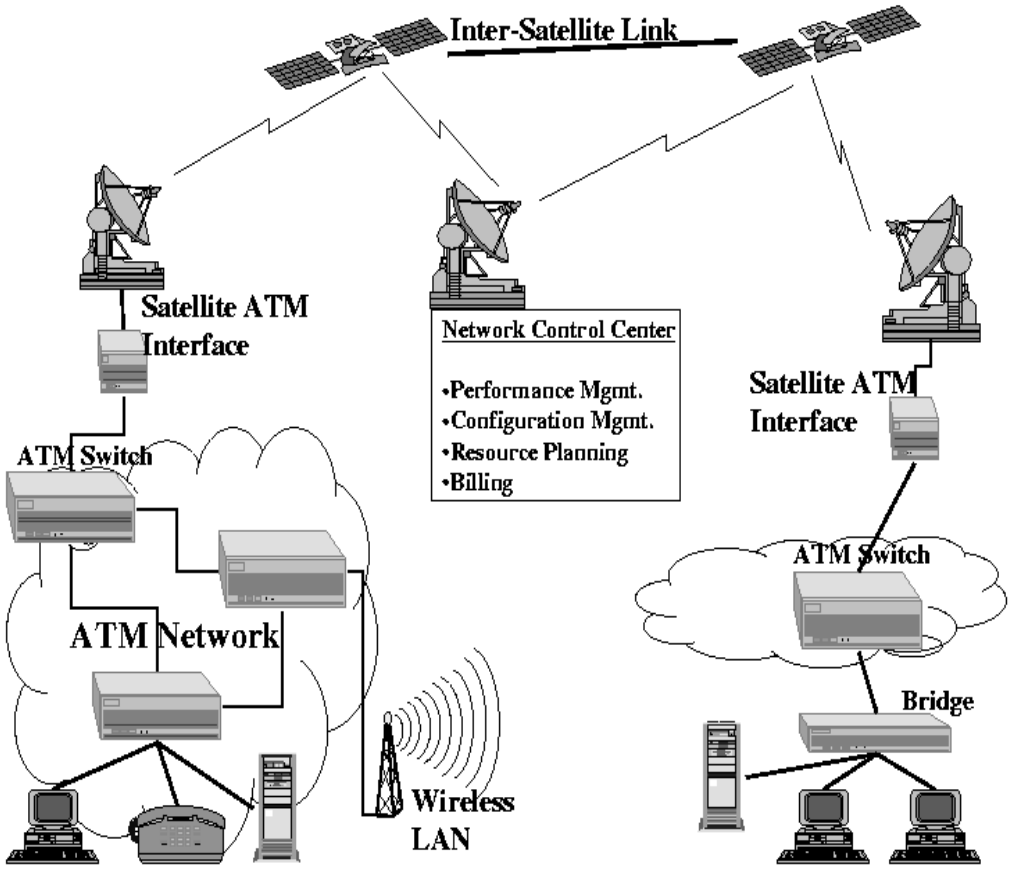
quality of service requirements. A scheduling mechanism at each multiplexing point serves the queues based on similar criteria.

Note that the above description illustrates a general model, and particular networking protocols or switch implementations may change the queuing structure at various queuing points. For example, low priority flows may be multiplexed directly into the low priority class queue without any per-flow queues. Also, the physical structure of the queues may vary depending on the switch architecture. For example, the queues may all share a common buffer pool allowing a dynamic allocation of resources to each queue, or may be physically separate buffers with static sizes. In most cases, a hybrid architecture will be implemented.

The design of the model is dictated by the QoS requirements of the various classes and the various capabilities and limitations of existing per-VC queuing and buffer management techniques. The CBR and VBR-rt classes require per-VC scheduling to provide throughput, delay and jitter guarantees. The VCs from this class form the highest priority class in the scheduler. The scheduler parameters must be set to reflect the guarantees and the traffic characteristics of each connection. The nature of the physical layer must also be taken into account in determining delay and jitter guarantees. The VCs from the VBR-nrt class form the next priority level in the scheduler. Delay guarantees are not provided for this class or any other class with lower priority. As a result, the scheduler uses strict priority when scheduling between the schedulable VCs from the various classes.

The ABR, GFR and UBR classes do not require per-VC queuing to meet their QoS guarantees, with one exception. The exception is made for connections that use the virtual source/virtual destination (VS/VD) option for the ABR class. VS/VD

requires the switches to act as a virtual end-system for the VS/VD sources. As a result, switches must control the output rates of the VS/VD VCs and enforce the ATM ABR source rules for those VCs. Per-VC queuing is thus used to enforce per-VC rates for VS/VD. The VS/VD queues drain into a common ABR class queue that is shared by the other non-VS/VD ABR VCs. A fair buffer sharing policy must be used in the ABR class queue to ensure that each ABR VC gets a fair share of its allocated throughput. The GFR and the UBR classes have a single FIFO buffer each. The GFR class can use DFBA (see chapter 6) for minimum throughput guarantees, while the UBR class can use selective drop or fair buffer allocation (see chapter 4) for equal division of UBR throughput.



A satellite-ATM network consists of a space segment, a ground segment and a network control segment. The space segment is the satellites and their crosslinks. The ground segments are the terrestrial ATM networks connected by the space segment. The network control segment consists of the NCC and performs the management, resource allocation and billing functions.

Figure 2.13: Satellite-ATM network architecture

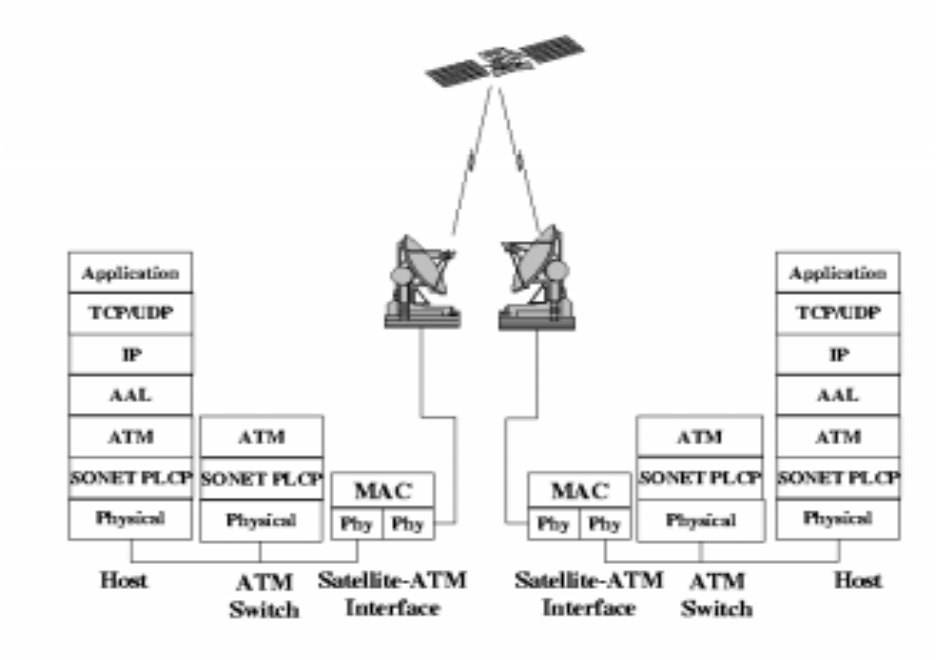
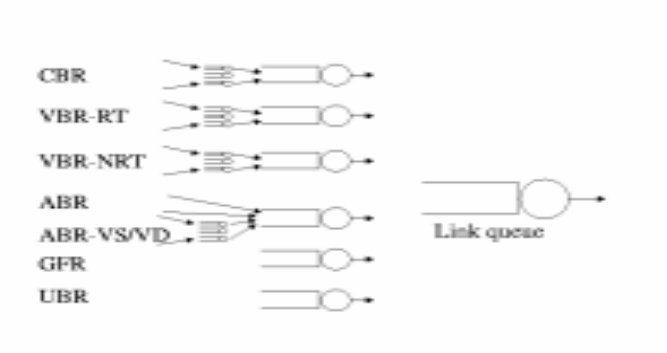


Figure 2.14: The TCP over satellite-ATM protocol stack



An ATM queuing model consists of per-VC queues, per-class queues and a link queue. VCs of priority classes use per-VC queues while best effort VCs are multiplexed into a per-class queue. A scheduling algorithm services each queue according to a service discipline.

Figure 2.15: A queuing architecture for ATM

CHAPTER 3

Problem Statement

The traffic management problem can be analyzed from two perspectives:

1. Network policies

2. End system policies.

The network can implement a variety of mechanisms to optimize resource utilization, fairness and higher layer throughput. For ATM, these include enhancements like intelligent drop policies to improve utilization, per-VC accounting to improve fairness and minimum throughput guarantees to the higher layers.

At the end system, the transport layer can implement various congestion avoidance and control policies to improve its performance and to protect against congestion collapse. Several transport layer congestion control mechanisms have been proposed and implemented. The mechanisms implemented in TCP are slow start and congestion avoidance, fast retransmit and recovery, and selective acknowledgments. Several others like forward acknowledgments and negative acknowledgments have been proposed as enhancements to the timeout based schemes.

The switch based policies interact with the end-system policies, and both of these are affected by the delay-bandwidth product of the network. As a result, it is important that the correct policies are designed for switches located in large delay-bandwidth networks, so that they work well with the end system policies. **In this project, we design switch policies and analyze end-system policies for ATM networks**. It should be noted that the end system policies are not controllable by the network. The network must optimize its own drop policies so that performance is optimized. The end system must choose a policy that works well with the switches. We make recommendations for the ideal switch and end system policies that maximize performance.

3.1 TCP/IP over ATM: Problem Specification

In this research, we design mechanisms to optimize the performance of TCP/IP over the following three ATM Service categories:

- Unspecified Bit Rate (UBR)
- Guaranteed Frame Rate (GFR)
- Available Bit Rate (ABR)

In particular, we study the following design options for an ATM network supporting efficient services to transport TCP data:

- **UBR with tail drop:** UBR is a best effort service category that provides no guarantees to the user. The baseline implementation of UBR uses a FIFO buffer with tail drop to discard cells when the buffer becomes full. It is interesting to study the performance of different TCP flavors over UBR with tail drop. We

show that TCP performs poorly over UBR for several scenarios. Two main reasons for the poor performance are the coarse grained TCP transmission timeout and TCP synchronization.

- **UBR with frame based discard:** Among frame based discard policies, the Early Packet Discard [87] policy is widely used [30]. The effect of EPD on TCP over ATM must be examined. We show that EPD increases the overall throughput of TCP over ATM. However, EPD cannot provide fairness among TCP connections.

- **UBR with intelligent buffer management:** To overcome the limitations of EPD, we present the design of the Selective Drop (SD) scheme to improve the performance for TCP over UBR. SD is based on FBA, but is simpler to implement and less sensitive to parameters. We show that SD and FBA improve the fairness of TCP over UBR in many cases.

- **Buffer Requirements for TCP/IP over UBR:** The size of buffers in the switch is a critical design parameter that effects TCP performance. A quantitative study is required for estimating the buffer size necessary to provide high TCP throughput over UBR. We perform simulations to show that a buffer size of one-half round trip delay-bandwidth product is necessary to provide good performance.

- **Effect of higher priority background traffic on TCP over UBR:** A multiservice network transports higher priority variable bit rate traffic along with UBR traffic. The effect of higher priority traffic on TCP over UBR has not been studied before. We study the performance of higher priority VBR traffic.

We propose the use of guaranteed rates for UBR to improve TCP performance in the presence of higher priority variable bit rate traffic.

- **GFR implementation options for TCP/IP transport:** The GFR service is intended to support per-VC minimum rate guarantees. Currently very few suggested implementations of the GFR service exist and these suffer from several limitations. Sample implementations can use a combination of policing, buffer management and scheduling in the network. We describe a buffer management scheme called Differential Fair Buffer Management (DFBA) scheme that can be used to implement the GFR service using a FIFO buffer.

- **ABR with Virtual Source / Virtual Destination:** The ABR service provides an MCR guarantee to the VCs and a fair share of any unused capacity. ABR is different from GFR in several ways, but the most important is that ABR uses a rate-based closed-loop feedback control mechanism for congestion control. ABR allows the feedback control to be end-to-end, or broken into several hops using the virtual source/virtual destination option (VS/VD). Extensive research on ABR [61] has shown that optimal buffer requirements at the bottleneck are proportional to the round trip delay-bandwidth product of the control loop. We argue that this requirement is unrealistic for terrestrial networks connected to satellite links. We design a VS/VD scheme that can help in buffer sizing in the different segments of a network based on the round trip feedback delays of the segments.

In addition to the network based options, we study three TCP congestion control techniques that are standardized or in the process of being standardized:

- Slow start and congestion avoidance (TCP Vanilla)
- Fast retransmit and recovery (TCP Reno)
- Selective acknowledgments (TCP SACK)

Vanilla and Reno TCP are standard mechanisms that are widely deployed in TCP stacks. TCP New Reno and SACK have recently been proposed as performance enhancements to TCP congestion control and are being incorporated in TCP implementations. Studies have reported performance results of the above TCP options over ATM [4]. However, these studies have focused only on TCP mechanisms and have not considered intelligent network based traffic management and guaranteed rate policies. Also, the studies are all performed using a best effort service framework without any provision for rate guarantees.

3.2 Performance Metrics

When ATM networks carry TCP/IP data, the end-to-end performance is measured at the TCP layer in the form of TCP throughput. To measure network performance, the throughputs of all TCPs passing through the bottleneck link are added and expressed as a fraction of the total capacity of the bottleneck link. This is called the *efficiency* of the network. We now define this formally.

Let N TCP source-destination pairs send data over a network with bottleneck link capacity R bits/sec. Let x_i be the observed throughput of the ith TCP source ($0 < i \leq N$). Let C be the maximum TCP throughput achievable on the link.

Definition 1 (Efficiency, E) *The Efficiency of the network is the ratio of the sum of the actual TCP throughputs to the maximum possible throughput achievable at the TCP layer.*

$$E(x_1, \ldots, x_N, C) = \frac{\sum_{i=1}^{i=N} x_i}{C}$$

The TCP throughputs x_i's are measured at the destination TCP layers. Throughput is defined as the total number of bytes delivered to the destination application (excluding retransmission and losses) divided by the total connection time. This definition is consistent with the definition of *goodput* in [31].

The maximum possible TCP throughput C is the throughput attainable by the TCP layer running over an ATM network with link capacity R. For example consider TCP over UBR on a 155.52 Mbps link (149.7 Mbps after SONET overhead) with a 9180 byte byte TCP MSS. For 9180 bytes of data, the ATM layer receives 9180 bytes of data, 20 bytes of TCP header, 20 bytes of IP header, 8 bytes of LLC header and 8 bytes of AAL5 trailer. These are padded to produce 193 ATM cells. Thus, each TCP segment results in 10229 bytes at the ATM Layer. From this, the maximum possible throughput = 9180/10229 = 89.7% = 135 Mbps approximately. It should be noted that ATM layer throughput does not necessarily correspond to TCP level throughput because some bandwidth may be wasted during TCP retransmissions.

In addition to providing high overall throughput, the network must also allocate throughput fairly among competing connections. The definition of fairness is determined by the particular service guarantees. For example, although UBR makes not service guarantees, fairness for TCP over UBR can be defined as the ability for UBR to provide equal throughput to all greedy TCP connections. In ABR and GFR, fairness is determined ability to meet the MCR guarantee and to share the excess capacity in some reasonable fashion. We measure fairness using the Fairness Index F described in [57].

Definition 2 (Fairness Index, F) *The Fairness Index is a function of the variability of the throughput across the TCP connections defined as*

$$F((x_1, e_1), \ldots, (x_n, e_N)) = \frac{(\sum_{i=1}^{1=N} x_i/e_i)^2}{N \times \sum_{i=1}^{i=N} (x_i/e_i)^2}$$

where x_i = observed throughput of the ith TCP connection $(0 < i \leq N)$
and e_i = expected throughput or fair share of the ith TCP connection.

For a symmetrical configuration using TCP over UBR, e_i can be defined as an equal share of the bottleneck link capacity ($e_i = C/N$). Thus, the fairness index metric applies well to N-source symmetrical configurations. In this case, note that when $x_1 = x_2 = \ldots = x_n$ then fairness index = 1. Also, low values of the fairness index represent poor fairness among the connections. The desired values of the fairness index must be close to 1. We consider a fairness index of 0.99 to be near perfect. A fairness index of 0.9 may or may not be acceptable depending on the application and the number of sources involved. Details on the fairness metric can be found in [57]. This fairness index has been used in several studies including [31] and [7]. In general, for a more complex configuration, the value of e_i can be derived from a a fair share definition that provides max-min fairness to the connections [52].

3.3 Approach

The primary means of performance analysis for this work is through experimental design, discrete event simulation and analysis. Simulation is a useful technique for the design and analysis of computer and communications systems. Simulations provide a controlled environment in which to test various hypotheses. Proper experimental design ensures that the set of simulations completely tests the effects of the different

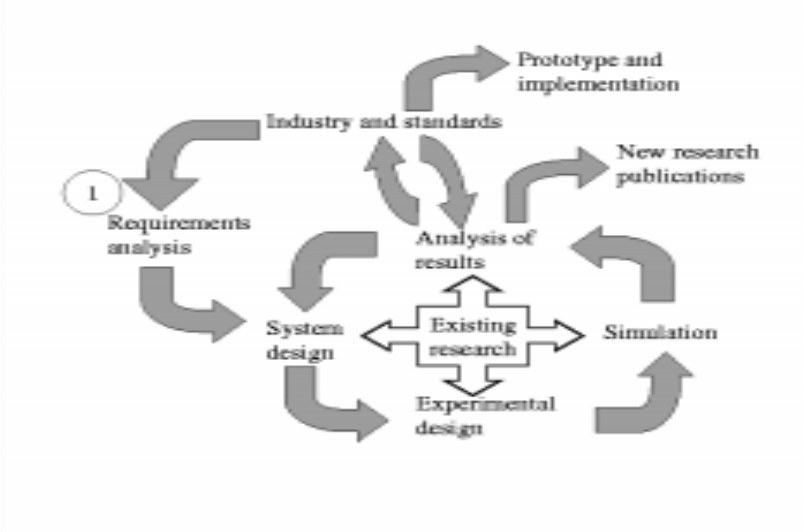


Figure 3.1: Research Methodology

factors in the system. We evaluate the issues for a wide range of networks, including low delay campus networks, wide area networks and satellite networks. Figure 3.1 illustrates the individual steps in the research process. The techniques used in this research are outlined in [57, 16]. In order to ensure that the research results are applicable to real networks, we have worked closely with ATM forum Traffic Management group during the definition phase of the problem and its solution.

We use the *netsim* [47] simulation tool for our research experiments. Netsim is an event driven simulator which consists of two parts: a core simulator that has support for event handling, and a set of components that pass events among one another based on an a standard interface with the core simulator. The simulator simulates a set of objects, called components, that send messages to one another. Each component models a corresponding component in the network. For example, in this research we develop components for switches, links, TCP end systems, ATM network interfaces

and traffic sources. *Netsim* was originally developed at MIT and is widely used by other research groups doing ATM Network research. Our version of the *netsim* source is from NIST [77]. For a detailed description of *netsim* the reader is referred to [47].

The above set of problems are categorized into the following four groups, corresponding to the components for optimizing TCP performance over ATM networks:

Part 1 (UBR+: Optimizing the performance of TCP over UBR.) We study the performance of TCP vanilla, TCP Reno and TCP SACK, with frame based and intelligent buffer management policies within a best effort framework. We present simulation results to calculate the optimal buffer sizes for a large number of TCP sources over satellites.

Part 2 (Guaranteed Rate: Effect of higher priority traffic on TCP.) We show how the performance of TCP degrades in the presence of higher priority traffic sharing the link. We describe the use of guaranteed rate to improve TCP/UBR performance in the presence of higher priority traffic.

Part 3 (GFR: Performance of TCP over GFR.) We describe the GFR service category and its implementation options. We propose the DFBA scheme that uses a FIFO buffer and provides per-VC minimum rate guarantees to TCP traffic.

Part 4 (ABR: Use of VS/VD in buffer allocation.) We design a VS/VD algorithm and show how it can be used for buffer sizing in satellite networks.

CHAPTER 4

UBR+: Improving the Performance of TCP over UBR

The Unspecified Bit Rate (UBR) service provided by ATM networks has no explicit congestion control mechanisms [34]. However, it is expected that many TCP implementations will use the UBR service category. TCP employs a window-based end-to-end congestion control mechanism to recover from segment loss and avoids congestion collapse. Several studies have analyzed the performance of TCP over the UBR service. TCP sources running over ATM switches with limited buffers experience low efficiency and low fairness [27, 40, 70, 69].

Figure 4.1 illustrates a framework for the various design options available to networks and end-systems for congestion control. Intelligent drop policies at switches can be used to improve throughput of transport connections. We show that Early Packet Discard (EPD) [87] improves TCP efficiency but not fairness [40]. Enhancements that perform intelligent cell drop policies at the switches need to be developed for UBR to improve transport layer performance. A policy for selective cell drop based on per-VC buffer management can be used to improve fairness.

In addition to network based drop policies, end-to-end flow control and congestion control policies can be effective in improving TCP performance over UBR. The fast retransmit and recovery mechanism [39] can be used in addition to slow start and

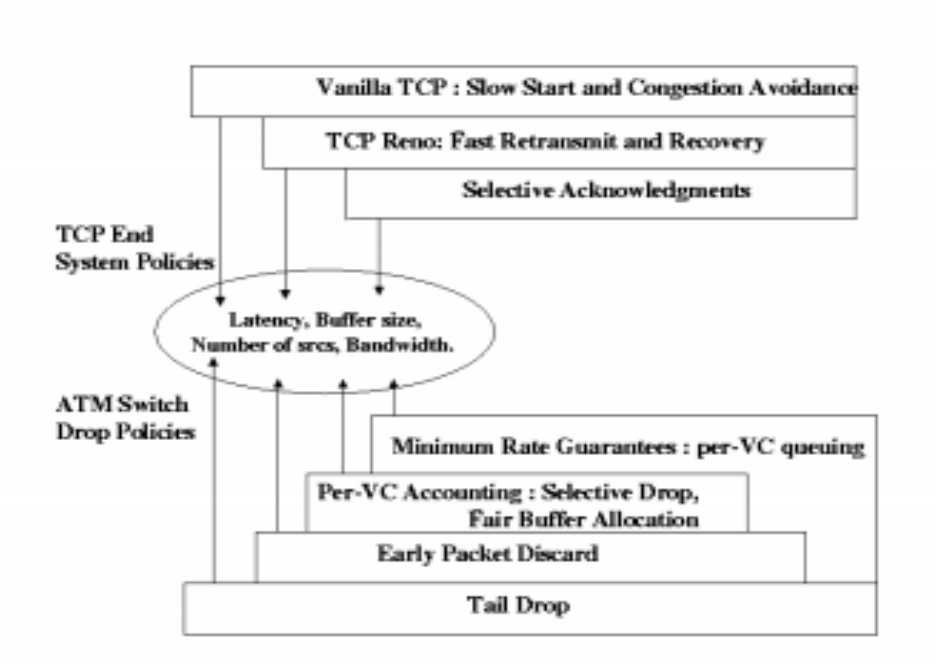


Figure 4.1: Design issues for TCP over UBR

congestion avoidance to quickly recover from isolated segment losses. The selective acknowledgments (SACK) option has been proposed to recover quickly from multiple segment losses. A change to TCP's fast retransmit and recovery has also been suggested in [26, 48].

4.1 Chapter Goals

In this chapter, we propose a per-VC buffer management scheme called Selective Drop to improve TCP performance over UBR. This scheme is similar to the Fair Buffer Allocation (FBA) scheme proposed in [45]. FBA has been proposed for the UBR service, but no performance analysis has been performed on FBA. We present an analysis of the operation of these schemes and the effects of their parameters. We also provide guidelines for choosing the best parameters for FBA and Selective Drop.

We then present simulation results for TCP over the various UBR enhancements. Although, other buffer management schemes have been presented in recent literature, their performance has not been evaluated with the various TCP congestion control options and for different latencies. We evaluate the performance of the enhancements to UBR, as well as to TCP congestion control mechanisms. We study the performance and interoperability of the network and the end-system enhancements for low latency and large latency configurations.

We first explain why TCP congestion control mechanisms can result in low throughput during congestion. We then describe our simulation setup used for all our experiments. We present the performance of TCP over vanilla UBR and explain why TCP over vanilla UBR results in poor performance. Sections 4.3 and 4.4 describe in detail, the enhancements to the UBR service category and our implementations. These enhancements include EPD, Selective Drop and Fair Buffer Allocation. Section 4.5 presents simulation results for the TCP modifications with each of the UBR changes. We describe the relative impact of buffer management and discard policies for long latency connections. We then present simulations to quantitatively assess buffer requirements for TCP over ATM. Section 4.9 presents a summary of this chapter.

4.2 TCP over UBR

In its simplest form, an ATM switch implements a tail drop policy for the UBR service category. When a cell arrives at the FIFO queue, if the queue is full, the cell is dropped, otherwise the cell is accepted. If a cell is dropped, the TCP source loses time waiting for the retransmission timeout. Even though TCP congestion mechanisms effectively recover from loss, the resulting throughput can be very low. It is also

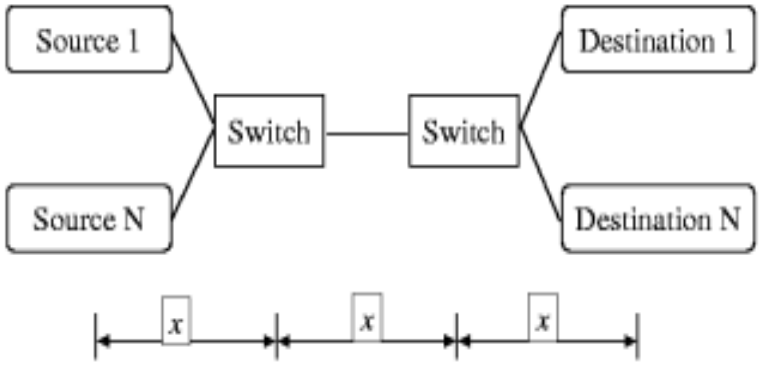


Figure 4.2: The N-source TCP configuration

known that FIFO buffering with tail drop results in excessive wasted bandwidth. Tail drop of ATM cells results in the receipt of incomplete segments. When part of a segment is dropped at the switch, the incomplete segment is dropped at the destination during reassembly. This wasted bandwidth further reduces the effective TCP throughput. In this section we describe our simulation results to exhibit the poor performance of TCP over UBR. We first describe our simulation model and performance metrics and then go on to discuss our simulation results.

4.2.1 Simulation Model

All simulations presented in this chapter are performed on the N source configuration shown in Figure 4.2. The configuration consists of N identical TCP sources that send data whenever allowed by the window. The switches implement UBR service with optional drop policies described in this chapter. Based on the recommendations in [24], we use the following simulation parameters

- The configuration consists of N identical TCP sources as shown in Figure 4.2.
- All sources are infinite TCP sources. The TCP layer always sends a segment as long as it is permitted by the TCP window.
- All link delays are 5 microseconds for LANs and 5 milliseconds for WANs[3]. Thus, the Round Trip Time due to the propagation delay is 30 microseconds and 30 milliseconds for LAN and WAN respectively.
- All link bandwidths are 155.52 Mbps. Peak Cell Rate is 146.9 Mbps after factoring in the SONET overhead.
- The traffic is unidirectional. Only the sources send data. The destinations send only acknowledgments.
- The TCP segment size is set to 512 bytes. This is the standard value used by many current TCP implementations. We experiment with larger values for satellite connections.
- TCP timer granularity is set to 100 ms. This affects the triggering of retransmission timeout due to packet loss. The values used in most TCP implementations are 500 ms and 100 ms. Several other studies [15, 69] have used a smaller TCP timer granularity and obtained higher efficiency values. However, the timer granularity is an important factor in determining the amount of time lost during congestion. Small granularity results in less time being lost waiting for the retransmission timeout to trigger. This results in faster recovery and higher

[3]In this dissertation, we refer to low latency connections as LAN connections and high latency connections as WAN connections. LAN and WAN do not refer to the legacy LAN/WAN architectures.

throughput. However, TCP implementations do not use timer granularities of less than 100 ms and producing results with lower granularity artificially increases the throughput. Moreover, the TCP RTT measurement algorithm is suited for coarse granularity timers. Moderate variation in RTT with a fine granularity timer can result in false TCP timeouts.

- TCP maximum receiver window size is 64K bytes for LANs. This is the default value used in TCP. For WANs, this value is not enough to fill up the pipe and reach full throughput. In the WAN simulations we use the TCP window scaling option to scale the window to the bandwidth delay product of approximately 1 RTT. The window size used for WANs is 600000 Bytes.

- TCP delay ack timer is NOT set. Segments are acked as soon are they are received.

- Duration of simulation runs is 10 seconds for LANs and 20 seconds for WANs.

- All TCP sources start sending at the same time and continue to send data through the duration of the simulation. This increases the probability of synchronization among connections and represents a worst case scenario.

4.2.2 Simulation Results

We simulated 5 and 15 TCP sources with finite buffered switches. The simulations were performed with two values of switch buffer sizes both for LAN and WAN links. For WAN experiments, we chose buffer sizes of approximately 1 and 3 times the round trip bandwidth-delay product of the connection. Thus, we selected WAN buffer sizes of 12000 and 36000 cells. For LANs, 1 round trip $\times$ bandwidth is a very small

number (11 cells) and is not practical as the size for the buffer. For LAN links, the buffer sizes chosen were 1000 and 3000 cells. These numbers are closer to the buffer sizes of current LAN switches and have been used by other studies on TCP over UBR [27, 70, 69]. The values for WANs were chosen in multiples of round trip time because most ABR feedback control mechanisms can achieve good steady state performance in a fixed number of round trip times and have similar buffer requirements for zero loss at the switch. Studies on TCP over ABR have used similar values of buffer sizes for both LANs and WANs [59]. It is interesting to assess the performance of TCP over UBR in this situation.

Column 4 of tables 4.1 and 4.2 show the efficiency and fairness values respectively for these experiments. Several results can be concluded from these observations.

Result 4.1 Fairness. TCP over vanilla UBR results in low fairness in both LAN and WAN configurations.

This is due to TCP synchronization effects. TCP connections are synchronized when their sources timeout and retransmit at the same time. This occurs because packets from all sources are dropped forcing them to enter the slow start phase. However, in this case, when the switch buffer is about to overflow, one or two connections get lucky and their entire windows are accepted while the segments from all other connections are dropped. All these connections wait for a timeout and stop sending data into the network. The connections that were not dropped send their next window and keep filling up the buffer. All other connections timeout and retransmit at the same time. This results in their segments being dropped again and the synchronization effect is seen. The sources that escape the synchronization get most of the bandwidth.

The synchronization effect is particularly important when the number of competing connections is small.

Result 4.2 Efficiency. The default TCP maximum window size leads to low efficiency in LANs.

LAN simulations have very low efficiency values (less than 50%) while WAN simulations have higher efficiency values. For LANs, the the TCP receiver window size (65535 Bytes) corresponds to more than 1500 cells at the switch for each source. For 5 sources and a buffer size of 1000 cells, the sum of the window sizes is almost 8 times the buffer size. For WAN simulations, with 5 sources and a buffer size of 12000 cells, the sum of the window sizes is less than 6 times the buffer size. Moreover, the larger RTT in WANs allows more cells to be cleared out before the next window is seen. As a result, the WAN simulations have higher throughputs than LANs. For LAN experiments with smaller window sizes (less than the default), higher efficiency values are seen.

4.2.3 Buffer Requirements For Zero Loss

TCP performs best when there is zero loss. In this situation, TCP is able to fill the pipe and fully utilize the link bandwidth. During the exponential rise phase (slow start), TCP sources send out two segments for every segment that is acked. For N TCP sources, in the worst case, a switch can receive a whole window's worth of segments from N-1 sources while it is still clearing out segments from the window of the Nth source. As a result, the switch can have buffer occupancies of up to the sum of all the TCP senders' maximum window sizes.

Config-uration	Number of Sources	Buffer Size (cells)	UBR	EPD	Selective Drop	FBA
LAN	5	1000	0.21	0.49	0.75	0.88
LAN	5	3000	0.47	0.72	0.90	0.92
LAN	15	1000	0.22	0.55	0.76	0.91
LAN	15	3000	0.47	0.91	0.94	0.95
Column Average			0.34	0.67	0.84	0.92
WAN	5	12000	0.86	0.90	0.90	0.95
WAN	5	36000	0.91	0.81	0.81	0.81
WAN	15	12000	0.96	0.92	0.94	0.95
WAN	15	36000	0.92	0.96	0.96	0.95
Column Average			0.91	0.90	0.90	0.92

Efficiency increases with increase in buffer size. The effect of drop policies is significant in LAN. Selective Drop and FBA significantly improve efficiency for LANs.

Table 4.1: Vanilla TCP over UBR : UBR enhancements (Efficiency)

Config-uration	Number of Sources	Buffer Size (cells)	UBR	EPD	Selective Drop	FBA
LAN	5	1000	0.68	0.57	0.99	0.98
LAN	5	3000	0.97	0.84	0.99	0.97
LAN	15	1000	0.31	0.56	0.76	0.97
LAN	15	3000	0.80	0.78	0.94	0.93
Column Average			0.69	0.69	0.92	0.96
WAN	5	12000	0.75	0.94	0.95	0.94
WAN	5	36000	0.86	1	1	1
WAN	15	12000	0.67	0.93	0.91	0.97
WAN	15	36000	0.77	0.91	0.89	0.97
Column Average			0.76	0.95	0.94	0.97

Tail drop and EPD can be unfair especially in low delay networks. Both Selective Drop and FBA improve fairness.

Table 4.2: Vanilla TCP over UBR : UBR enhancements (Fairness)

Number of Sources	Configuration	Efficiency	Fairness	Maximum Queue (Cells)
5	LAN	1	1	7591
15	LAN	1	1	22831
5	WAN	1	1	59211
15	WAN	1	1	196203

The maximum queue size for N symmetrical TCPs sharing a bottleneck is proportional to the sum of their maximum window sizes

Table 4.3: TCP over UBR: Buffer requirements for zero loss

Table 4.3 contains the simulation results for TCP running over the UBR service with infinite buffering in the bottleneck switch. The maximum queue length numbers give an indication of the buffer sizes required at the switch to achieve zero loss for TCP. The connections achieve 100% of the possible throughput and perfect fairness. For the five source LAN configuration, the maximum queue length is 7591 cells = 7591 / 12 segments = 633 segments $\approx$ 323883 Bytes. This is approximately equal to the sum of the TCP window sizes (65535 $\times$ 5 bytes = 327675 bytes). For the five source WAN configuration, the maximum queue length is 59211 cells = 2526336 Bytes. This is slightly less that the sum of the TCP window sizes (600000 $\times$ 5 = 3000000 Bytes). This is because the switch has 1 RTT to clear out almost 500000 bytes of TCP data (at 155.52 Mbps) before it receives the next window of data. In any case, the increase in buffer requirements is proportional to the number of sources in the simulation. The maximum queue is reached just when the TCP connections reach the maximum window. After that, the window stabilizes and TCP's self clocking congestion mechanism puts one segment into the network for each segment that leaves the network.

Result 4.3 Buffer Requirements for Zero Loss For a switch to guarantee zero loss for TCP over UBR, the amount of buffering required is equal to the sum of the TCP maximum window sizes of all the TCP connections.

Note that the maximum window size is determined by the minimum of the sender's congestion window and the receiver's window.

For smaller buffer sizes, efficiency typically increases with increasing buffer sizes (see table 4.1). Larger buffer sizes result in more cells being accepted before loss occurs and therefore higher efficiency. This is a direct result of the dependence of the buffer requirements to the sum of the TCP window sizes. The buffer sizes used in the LAN simulations reflect the typical buffer sizes used by other studies of TCP over ATM [27, 59, 70] and implemented in ATM workgroup switches.

From the simulation results in this section, it is clear that TCP over UBR can experience poor performance. In the next section, we study enhancements to the UBR service category that improve TCP performance over UBR.

4.3 Early Packet Discard

The Early Packet Discard (EPD) policy [87] has been suggested to remedy some of the problems with tail drop switches. EPD drops complete packets instead of partial packets. As a result, the link does not carry incomplete packets which would have been discarded during reassembly. A threshold R less than the buffer size, is set at the switches. When the switch queue length exceeds this threshold, all cells from any new packets are dropped. Packets which had been partly received before exceeding the threshold are still accepted if there is buffer space. In the worst case, the switch could have received one cell from all N connections before its buffer exceeded the

threshold. To accept all the incomplete packets, there should be additional buffer capacity of upto the sum of the packet sizes of all the connections. Typically, the threshold R should be set to the buffer size − N × the maximum packet size, where N is the expected number of connections active at any time.

The EPD algorithm used in our simulations is the one suggested by [70, 69] and is given in appendix B. Column 5 of tables 4.1 and 4.2 show the efficiency and fairness respectively of TCP over UBR with EPD. The switch thresholds are selected so as to allow one entire packet from each connection to arrive after the threshold is exceeded. We use thresholds of Buffer Size − 200 cells in our simulations. 200 cells are enough to hold one packet each from all 15 TCP connections. This reflects the worst case scenario when all the fifteen connections have received the first cell of their packet and then the buffer occupancy exceeds the threshold.

Result 4.4 Effect of EPD. EPD improves the efficiency of TCP over UBR, but it does not significantly improve fairness in LANs.

(see Tables 4.1 and 4.2.) This is because EPD indiscriminately discards complete packets from all connections without taking into account their current rates or buffer utilizations. When the buffer occupancy exceeds the threshold, all new packets are dropped. There is a more significant improvement in fairness for WANs because of the relatively larger buffer sizes.

4.4 Per-VC Accounting: Selective Drop and Fair Buffer Allocation

Intelligent buffer management schemes have been proposed that use per-VC accounting to maintain the current buffer utilization of each UBR VC. The Selective

Drop scheme proposed in this section is one such scheme. Selective Drop is similar to another per-VC accounting scheme called Fair Buffer Allocation [45]. In these schemes, a fair allocation is calculated for each VC and if the VC's buffer occupancy exceeds its fair allocation, its next incoming packet is dropped. Both schemes maintain a threshold R, as a fraction of the buffer capacity K. When the total buffer occupancy exceeds R×K, new packets are dropped depending on the VC's (say VC_i) buffer occupancy (Y_i).

Selective Drop keeps track of the activity of each VC by counting the number of cells from each VC in the buffer. A VC is said to be active if it has at least one cell in the buffer. A fair allocation is calculated as the current buffer occupancy divided by the number of active VCs.

Let the buffer occupancy be denoted by X and the number of active VCs be denoted by N_a. Then the fair allocation or fair share F_s for each VC is given by,

$$F_s = \frac{X}{N_a}$$

The ratio of the number of cells of a VC in the buffer to the fair allocation gives a measure of how much the VC is overloading the buffer i.e., by what ratio it exceeds the fair allocation. Let Y_i be the number of cells from VC_i in the buffer. Then the Load Ratio, L_i, of VC_i is defined as

$$L_i = \frac{Y_i}{F_s}$$

or

$$L_i = \frac{Y_i \times N_a}{X}$$

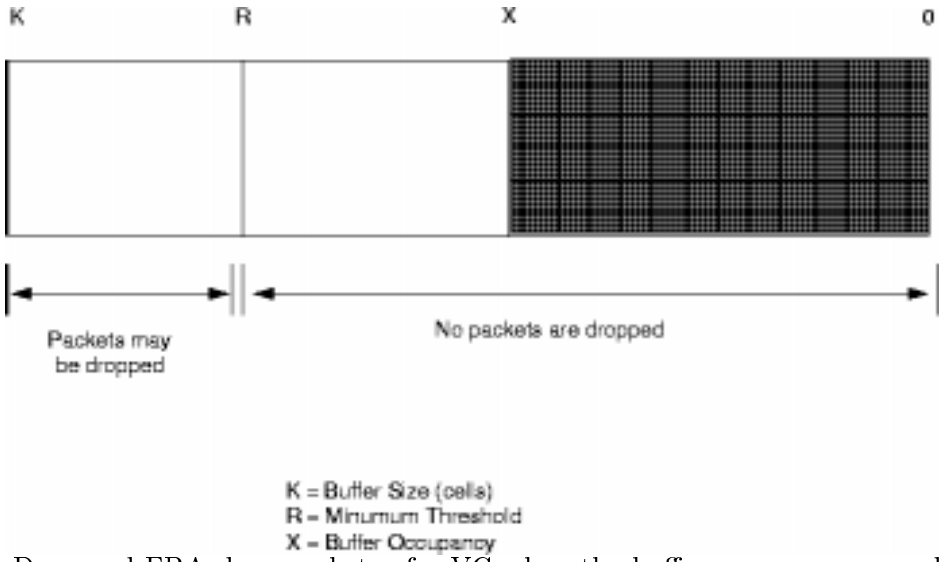
If the load ratio of a VC is greater than a parameter Z, then new packets from that VC are dropped in preference to packets of a VC with load ratio less than Z.

Thus, Z is used as a cutoff for the load ratio to indicate that the VC is overloading the switch.

Figure 4.3 illustrates the drop conditions for Selective Drop. For a given buffer size K (cells), the selective drop scheme assigns a static minimum threshold parameter R (cells). If the buffer occupancy X is less than or equal to this minimum threshold R, then no cells are dropped. If the buffer occupancy is greater than R, then the next new incoming packet of VC_i is dropped if the load ratio of VC_i is greater than Z.

We performed simulations to find the value of Z that optimizes the efficiency and fairness values. We first performed 5 source LAN simulations with 1000 cell buffers. We set R to 0.9 $\times$ the buffer size K. This ensured that there was enough buffer space to accept incomplete packets during congestion. We experimented with values of $Z =$ 2, 1, 0.9, 0.5 and 0.2. $Z = 0.9$ resulted in good performance. Further simulations of values of Z around 0.9 showed that $Z = 0.8$ produces the best efficiency and fairness values for this configuration. For WAN simulations, any Z value between 0.8 and 1 produced the best results.

The Fair Buffer Allocation Scheme proposed by [45] uses a more complex form of the parameter Z and compares it with the load ratio L_i of a VC. To make the cutoff smooth, FBA uses the current load level in the switch. The scheme compares the load ratio of a VC to another threshold that determines how much the switch is congested. For a given buffer size K, the FBA scheme assigns a static minimum threshold parameter R (cells). If the buffer occupancy X is less than or equal to this minimum threshold R, then no cells are dropped. When the buffer occupancy is greater than R, then upon the arrival of every new packet, the load ratio of the VC (to which the packet belongs) is compared to an allowable drop threshold T calculated



Selective Drop and FBA drop packets of a VC when the buffer occupancy exceeds R and the per-VC occupancy is greater than its fair share.

Figure 4.3: Selective Drop and FBA: Buffer occupancies for drop

as

$$T = Z \times \frac{K - R}{X - R}$$

In this equation Z is a linear scaling factor. The next packet from VC_i is dropped if

$$(X > R) \text{ AND } (\frac{Y_i \times N_a}{X} > Z \times \frac{K - R}{X - R})$$

Figure 4.3 shows the switch buffer with buffer occupancies X relative to the minimum threshold R and the buffer size K where incoming TCP packets may be dropped.

Note that when the current buffer occupancy X exceeds the minimum threshold R, it is not always the case that a new packet is dropped. The load ratio in the above equation determines if VC_i is using more than a fair amount of buffer space. X/N_a is used as a measure of a fair allocation for each VC and $Z \times ((K - R)/(X - R))$ is a drop threshold for the buffer. If the current buffer occupancy (Y_i is greater than this

dynamic threshold times the fair allocation (X/N_a), then the new packet of that VC is dropped.

4.4.1 Effect of Parameters

Effect of the minimum drop threshold R

Figure 4.4 shows the plots of ((K − R)/(X − R)) for buffer size K = 1000 cells and buffer occupancy X = 900 cells as the minimum threshold R varies from 100 to 850 cells. From the expression, Z((K − R)/(X − R)), as R increases, for a given value of the buffer occupancy X, the allowable drop threshold increases as expected. The load ratio threshold for dropping a complete packet is $Z((K-R)/(X-R))$. As R increases for a fixed value of the buffer occupancy X, $X-R$ decreases, which means that the drop threshold $((K-R)/(X-R))$ increases and each connection is allowed to have more cells in the buffer. Higher values of R provide higher efficiency by allowing higher buffer utilization. Lower values of R provide better fairness than higher values by dropping packets earlier.

Effect of the linear scale factor Z

The parameter Z scales the FBA drop threshold by a multiplicative factor. Z has a linear effect on the drop threshold, where lower values of Z lower the threshold and vice versa. Figure 4.5 illustrates how the drop threshold Z((K − R)/(X − R)) varies with the buffer occupancy. The buffer capacity K is set to 1000 cells and the minimum threshold R is set to 900 cells. The three lines show the effect of three values of Z as the buffer occupancy X varies from 900 to 950. Lower values of Z shift the curve down so that the drops occur earlier. Higher values of Z should increase

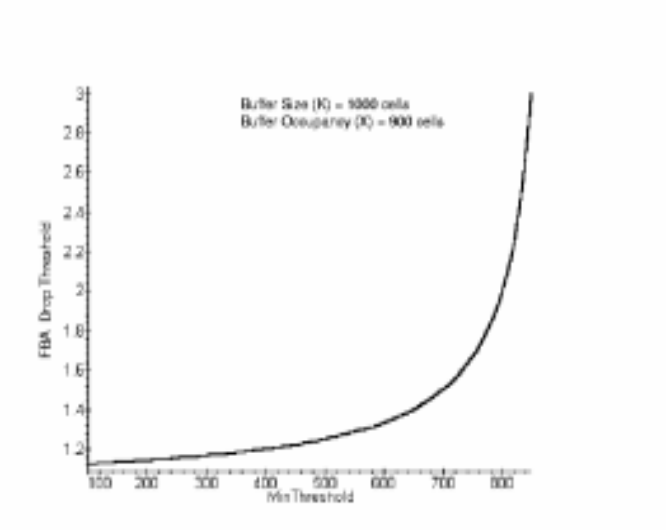


Figure 4.4: FBA: Effect of the minimum drop threshold

the efficiency of the connections. However, if Z is very close to 1, then cells from a connection may not be dropped until the buffer overflows.

4.4.2 Simulation Model

We performed a full factorial experiment [57] with the following parameter variations for both LANs and WANs. Each experiment was performed for N-source configuration.

- Number of sources, N = 5 and 15.
- Buffer capacity, K = 1000, 2000 and 3000 cells for LANs and 12000, 24000 and 36000 cells for WANs.
- Minimum drop threshold, R = 0.9×K, 0.5×K and 0.1×K.
- Linear scale factor, Z = 0.2, 0.5 and 0.8.

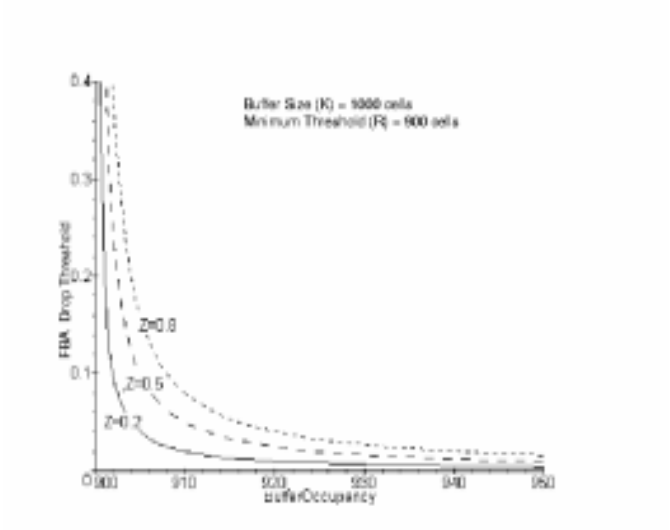


Figure 4.5: FBA: Effect of the linear scale factor

The remaining parameters were the same as before:

A set of 54 experiments were conducted to determine the values of R and Z that maximized efficiency and fairness among the TCP sources. We sorted the results with respect to the efficiency and fairness values.

We also performed a similar set of experiments with Selective Drop to assess the optimal value of its parameters.

The complete simulation results are listed in tables C.1, C.2, C.3, C.4, C.5 and C.6. The results are summarized below:

4.4.3 Simulation Results

Result 4.5 Efficiency versus Fairness. There is a tradeoff between efficiency and fairness.

The highest values of fairness (close to 1) have the lowest values of efficiency. The simulation data shows that these results are for low R and Z values. Higher values

of the minimum threshold R combined with low Z values lead to slightly higher efficiency. Efficiency is high for high values of R and Z. Lower efficiency values have either R or Z low and higher efficiency values have either of R or Z high. When R is low (0.1), the scheme drops packets when the buffer occupancy exceeds a small fraction of the capacity. When Z is low, a small rise in the load ratio results in packets being dropped. This improves the fairness of the scheme, but decreases the efficiency especially if R is also low. **For configurations simulated, we found that the best value of R was 0.9 and Z was 0.8.**

Result 4.6 Parameter sensitivity. The fairness of FBA is sensitive to parameters.

The simulation results showed that small changes in the values of R and Z can result in significant differences in the fairness results. With the increase of R and Z, efficiency shows an increasing trend. However there is considerable variation in the fairness numbers. We attribute this to TCP synchronization effects. Sometimes, a single TCP source can get lucky and its packets are accepted while all other connections are dropped. When the source finally exceeds its fair-share and should be dropped, the buffer is no longer above the threshold because all other sources have stopped sending packets and are waiting for timeout.

Result 4.7 Performance of SD and FBA Both Selective Drop and FBA improve both fairness and efficiency of TCP over UBR.

This is because cells from overloading connections are dropped in preference to underloading ones. As a result, Selective Drop and FBA are more effective in breaking TCP synchronization. When the buffer exceeds the threshold, only cells from overloading

connections are dropped. This frees up some bandwidth and allows the underloading connections to increase their window and obtain more throughput. In general, the average efficiency and fairness values for FBA (for optimal parameter values) are higher than the previously discussed options. Columns 6 and 7 of tables 4.1 and 4.2 show the fairness and efficiency values for Selective Drop and FBA with $R = 0.9$ and $Z = 0.8$ respectively.

Result 4.8 Effect of buffer size. Fairness and efficiency increase with increase in buffer size.

This supports the discussion in section 4.2.2 and shows that the performance improves with increasing buffer size for FBA and Selective Drop.

4.5 TCP Enhancements

In this section, we examine the effect of TCP enhancements discussed in chapter 2. We provide simulation results to compare relative TCP performance and present some analysis of the TCP behavior that explains the performance observed in the simulations.

4.5.1 TCP Reno: Simulation Results

Tables 4.4 and 4.5 list the simulation results of TCP Reno with each of the UBR options. Tables 4.8 and 4.9 compare the average efficiency and fairness values with the vanilla TCP results.

Result 4.9 Fast Retransmit and Recovery in WANs. For long latency connections (WAN), fast retransmit and recovery hurts the efficiency.

This is because congestion typically results in multiple packets being dropped. Fast retransmit and recovery cannot recover from multiple packet losses and slow start is triggered. The additional segments sent by fast retransmit and recovery (while duplicate ACKs are being received) may be retransmitted during slow start. In WAN links with large bandwidth delay products, the number of retransmitted segments can be significant. Thus, fast retransmit can add to the congestion and reduce throughput. Recall that figure 2.6 shows a case when three consecutive packets are lost from a window and the sender TCP incurs fast retransmit twice and then times out. At that time, SSTHRESH is set to one-eighth of the original congestion window value (CWND in the figure) As a result, the exponential phase lasts a very short time and the linear increase begins at a very small window. Thus, the TCP sends at a very low rate and loses much throughput.

Result 4.10 Fast Retransmit and Recovery in LANs. Fast retransmit and recovery improves the efficiency of TCP over UBR for the LAN configuration.

From table 4.4, the effect of multiple packet losses is much less visible in low latency connections because for a small RTT and large bandwidth, the linear increase very quickly fills up the network pipe. As a result it results in almost same efficiency as the exponential increase.

Result 4.11 EPD with Fast Retransmit and Recovery in LANs. The addition of EPD with fast retransmit and recovery results in a large improvement in both fairness for LANs.

Thus, the combination of EPD and fast retransmit can provide high throughput and fairness for low latency configurations.

Config-uration	Number of Sources	Buffer Size (cells)	UBR	EPD	Selective Drop	FBA
LAN	5	1000	0.53	0.97	0.97	0.97
LAN	5	3000	0.89	0.97	0.97	0.97
LAN	15	1000	0.42	0.97	0.97	0.97
LAN	15	3000	0.92	0.97	0.97	0.97
Column Average			0.69	0.97	0.97	0.97
WAN	5	12000	0.61	0.79	0.8	0.76
WAN	5	36000	0.66	0.75	0.77	0.78
WAN	15	12000	0.88	0.95	0.79	0.79
WAN	15	36000	0.96	0.96	0.86	0.89
Column Average			0.78	0.86	0.81	0.81

Fast retransmit and recovery with EPD can provide high efficiency in LANs. For WANs, fast retransmit and recovery provides low efficiency.

Table 4.4: Reno TCP over UBR (Efficiency)

4.5.2 SACK TCP: Simulation Results

We performed simulations for the LAN and WAN configurations for three drop policies – tail drop, Early Packet Discard and Selective Drop. Tables 4.6 and 4.7 show the efficiency and fairness values of SACK TCP with various UBR drop policies. Tables 4.8 and 4.9 show the comparative column averages for Vanilla, Reno and SACK TCP. Several results can be concluded from these tables:

Result 4.12 SACK Efficiency. For most cases, for a given drop policy, SACK TCP provides higher efficiency than the corresponding drop policy in vanilla TCP.

This confirms the intuition provided by the analysis of SACK that SACK recovers at least as fast as slow start when multiple packets are lost. In fact, for most cases, SACK recovers faster than both fast retransmit/recovery and slow start algorithms.

Config-uration	Number of Sources	Buffer Size (cells)	UBR	EPD	Selective Drop	FBA
LAN	5	1000	0.77	0.99	0.99	0.97
LAN	5	3000	0.93	0.99	1	0.99
LAN	15	1000	0.26	0.96	0.99	0.69
LAN	15	3000	0.87	0.99	0.99	1
Column Average			0.71	0.98	0.99	0.91
WAN	5	12000	0.99	1	0.99	1
WAN	5	36000	0.97	0.99	0.99	1
WAN	15	12000	0.91	0.96	0.99	0.95
WAN	15	36000	0.74	0.91	0.98	0.98
Column Average			0.90	0.97	0.99	0.98

Fairness is high for fast retransmit and recovery with EPD in the LAN configuration.

Table 4.5: Reno TCP over UBR (Fairness)

Result 4.13 LAN Efficiency. For LANs, the effect of drop policies is very important and can dominate the effect of SACK.

For UBR with tail drop, SACK provides a significant improvement over Vanilla and Reno TCPs. However, as the drop policies get more sophisticated, the effect of TCP congestion mechanism is less pronounced. This is because, the typical LAN switch buffer sizes are small compared to the default TCP maximum window of 64K bytes and so buffer management becomes a very important factor. Moreover, the degraded performance of SACK over Reno in LANs (see tables 4.8 and 4.9) is attributed to excessive timeout due to the retransmitted packets being lost. In this case SACK loses several round trips in retransmitting parts of the lost data and then times out. After the timeout, much of the data is transmitted again, resulting in wasted throughput. This result reinforces the need for a good switch drop policy for TCP over UBR.

Configuration	Number of Sources	Buffer (cells)	UBR	EPD	Selective Drop
LAN	5	1000	0.76	0.85	0.94
LAN	5	3000	0.98	0.97	0.98
LAN	15	1000	0.57	0.78	0.91
LAN	15	3000	0.86	0.94	0.97
Column Average			0.79	0.89	0.95
WAN	5	12000	0.90	0.88	0.95
WAN	5	36000	0.97	0.99	1.00
WAN	15	12000	0.93	0.80	0.88
WAN	15	36000	0.95	0.95	0.98
Column Average			0.94	0.91	0.95

Selective acknowledgments result in high efficiency for both LAN and WAN configurations.

Table 4.6: SACK TCP over UBR+: (Efficiency)

Configuration	Number of Sources	Buffer (cells)	UBR	EPD	Selective Drop
LAN	5	1000	0.22	0.88	0.98
LAN	5	3000	0.92	0.97	0.96
LAN	15	1000	0.29	0.63	0.95
LAN	15	3000	0.74	0.88	0.98
Column Average			0.54	0.84	0.97
WAN	5	12000	0.96	0.98	0.95
WAN	5	36000	1.00	0.94	0.99
WAN	15	12000	0.99	0.99	0.99
WAN	15	36000	0.98	0.98	0.96
Column Average			0.98	0.97	0.97

Fairness is unaffected by SACK and is dependent on the drop policy.

Table 4.7: SACK TCP over UBR+: (Fairness)

Result 4.14 WAN Efficiency. The throughput improvement provided by SACK is significant for wide area networks.

When the propagation delay is large, a timeout results in the loss of a significant amount of time during slow start from a window of one segment. With Reno TCP (with fast retransmit and recovery), performance is further degraded (for multiple packet losses) because timeout occurs at a much lower window size than vanilla TCP. With SACK TCP, a timeout is avoided most of the time and recovery is complete within a small number of roundtrips. Even if timeout occurs, the recovery is as fast as slow start but some time may be lost in the earlier retransmissions.

Result 4.15 SACK and Buffer Management. The performance of SACK TCP can be improved by intelligent drop policies like EPD and Selective drop.

This is consistent with our earlier results with Vanilla and Reno TCP . Thus, we recommend that intelligent drop policies be used in UBR service.

Result 4.16 SACK Fairness. The fairness values for selective drop are comparable to the values with the other TCP versions.

Thus, SACK TCP does not hurt the fairness in TCP connections with an intelligent drop policy like selective drop. The fairness of tail drop and EPD are sometimes a little lower for SACK TCP. This is again because retransmitted packets are lost and some connections time out. Connections which do not time out do not have to go through slow start and can utilize more of the link capacity. The fairness among a set of heterogeneous TCP connections is beyond the scope of this study.

4.5.3 SACK TCP: Analysis of Recovery Behavior

We now calculate a bound for the recovery behavior of SACK TCP and show that SACK TCP can recover from multiple packet losses more efficiently than Reno or vanilla TCP. This provides an intuitive explanation about the increased performance of SACK.

Suppose that at the instant when the sender learns of the first packet loss (from three duplicate ACKs), the value of the congestion window is CWND. Thus, the sender has CWND bytes of data waiting to be acknowledged. Suppose also that the network has dropped a block of data which is CWND/n bytes long (This typically results in several segments being lost). After one RTT of sending the first dropped segment, the sender receives three duplicate ACKs for this segment. It retransmits the segment, PIPE to CWND $-$ 3, and sets CWND to CWND/2. For each duplicate ACK received, PIPE is decremented by 1. When PIPE reaches CWND, then for each subsequent duplicate ACK received, another segment can be sent. All the ACKs from the previous window take 1 RTT to return. For half RTT nothing is sent (since PIPE $>$ CWND). For the next half RTT, if CWND/n bytes were dropped, then only CWND/2 $-$ CWND/n bytes (of retransmitted or new segments) can be sent.

Thus, all the dropped segments can be retransmitted in 1 RTT if

$$\frac{CWND}{2} - \frac{CWND}{n} \geq \frac{CWND}{n}$$

i.e., $n \geq 4$. Therefore, for SACK TCP to be able to retransmit all lost segments in one RTT, the network can drop at most CWND/4 bytes from a window of CWND.

Now, we calculate the maximum amount of data that can be dropped by the network for SACK TCP to be able to retransmit everything in two RTTs. Suppose again

that CWND/n bytes are dropped from a window of size CWND. Then, in the first RTT from receiving the 3 duplicate ACKs, the sender can retransmit upto CWND/2 − CWND/n bytes. In the second RTT, the sender can retransmit 2(CWND/2 − CWND/n) bytes. This is because for each retransmitted segment in the first RTT, the sender receives a partial ACK that indicates that the next segment is missing. As a result, PIPE is decremented by 2 and the sender can send 2 more segments (both of which could be retransmitted segments) for each partial ACK it receives.

Thus, all the dropped segments can be retransmitted in 2 RTTs if

$$\frac{CWND}{2} - \frac{CWND}{n} + 2(\frac{CWND}{2} - \frac{CWND}{n}) \geq \frac{CWND}{n}$$

i.e. $n \geq 8/3$. This means that at most 3×CWND/8 bytes can be dropped from a window of size CWND for SACK TCP to be able to recover in 2 RTTs.

Thus, the number of RTTs N_{rec} needed by SACK TCP to recover from a loss of CWND/n is given by

$$N_{rec} \leq log(\frac{n}{n-2}) \text{ for } 2 < n \leq 4$$

If less than one fourth of CWND is lost, then SACK TCP can recover in 1 RTT. If more than one half the CWND is dropped, then there are not enough duplicate ACKs for PIPE to become large enough to transmit any segments in the first RTT. Only the first dropped segment is retransmitted on the receipt of the third duplicate ACK. In the second RTT, the ACK for the retransmitted packet is received. This is a partial ACK and results in PIPE being decremented by 2 so that 2 packets can be sent. As a result, PIPE doubles every RTT and SACK recovers no slower than slow start [26, 32]. SACK is still advantageous because timeout is avoided unless a retransmitted packet were dropped.

Config-uration	UBR	EPD	Selective Drop
	LAN		
Vanilla TCP	0.34	0.67	0.84
Reno TCP	0.69	0.97	0.97
SACK TCP	0.79	0.89	0.95
	WAN		
Vanilla TCP	0.91	0.9	0.91
Reno TCP	0.78	0.86	0.81
SACK TCP	0.94	0.91	0.95

SACK TCP provides the best efficiency in most cases. For the WAN configuration, Reno TCP provides worst efficiency. EPD and Selective Drop significantly improve efficiency in the LAN configuration.

Table 4.8: TCP over UBR: Comparative Efficiencies

Config-uration	UBR	EPD	Selective Drop
	LAN		
Vanilla TCP	0.69	0.69	0.92
Reno TCP	0.71	0.98	0.99
SACK TCP	0.54	0.84	0.97
	WAN		
Vanilla TCP	0.76	0.95	0.94
Reno TCP	0.90	0.97	0.99
SACK TCP	0.98	0.97	0.97

Selective Drop provides high fairness in most cases. The effect of Selective Drop is more significant in LANs.

Table 4.9: TCP over UBR: Comparative Fairness

4.6 Effect of a Large Number of Sources

In workgroup and local area networks, the number of TCP connections active at any given time is small and can be realistically modeled by the above simulation results. However, in wide area networks, more than 15 TCP connections may be simultaneously active. It becomes interesting to assess the effectiveness of Selective Drop to provide high efficiency and fairness to a large number of sources.

Even with a large number of TCP connections, EPD does not significantly affect fairness over vanilla UBR, because EPD does not perform selective discard of packets based on buffer usage. However, with a large number of sources, the fairness metric can take high values even in clearly unfair cases. This is because, as the number of sources increases, the effect of a single source or a few sources on the the fairness metric decreases. As a result, vanilla UBR might have a fairness value of 0.95 or better even if a few TCP's receive almost zero throughput. The effect of unfairness is easily seen with a small number of sources. Vanilla UBR and EPD are clearly unfair, and the performance of Selective Drop needs to be tested with large number of sources for a more strict value of fairness. Jain [57] suggests that a value of 0.99 for the fairness metric reflects high fairness even for a large number of sources. Selective Drop should provide high efficiency and fairness in such cases.

We performed experiments with 50 and 100 TCP sources using SACK TCP and Selective Drop. The experiments were performed for WAN and satellite networks with the N source configuration. The simulations produced efficiency values of 0.98 and greater with fairness values of 0.99 and better for comparable buffer sizes. The simulations produced high efficiency and fairness for fixed buffer sizes, irrespective of the number of sources (5, 15, 50 or 100). The details of the simulation results with a

large number of sources are presented in section 4.8. The simulation results illustrate that SACK TCP with a per-VC buffer management policy like Selective Drop can produce high efficiency and fairness even for a large number of TCP sources.

From the simulation and analysis presented so far in this chapter, we know that vanilla TCP performs poorly because TCP sources waste bandwidth when they are waiting for a timeout. Reno TCP performs poorly in the case of multiple packet losses because of timeout and congestion avoidance at a very low window size. The effect of these behaviors is mitigated with a large number of sources. When a large number of sources are fairly sharing the link capacity, each TCP gets a small fraction of the capacity. The steady state window sizes of the TCPs are small. When packets are lost from a few TCPs, other TCPs increase their congestion widows to utilize the unused capacity within a few round trips. As a result, overall link efficiency improves, but at the expense of the TCPs suffering loss. The TCPs that lose packets recover the fastest with SACK TCP. Thus, SACK TCP can help in quickly achieving fairness after packet loss.

4.7 Effect of Long Latency: Satellite Networks

Since TCP congestion control is inherently limited by the round trip time, long delay paths have significant effects on the performance of TCP over ATM. A large delay-bandwidth link must be utilized efficiently to be cost effective. In this section, we present performance results of TCP over UBR and its enhancements, with satellite delays.

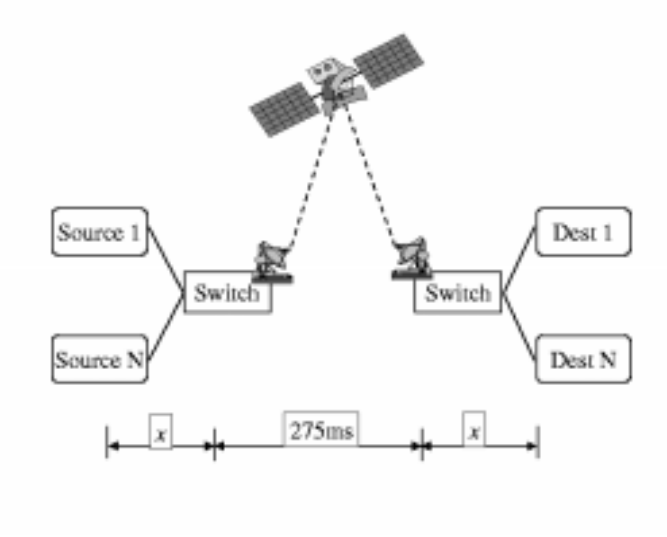


Figure 4.6: N-Source GEO Configuration

4.7.1 Simulation Model

The simulations use the N source satellite configurations with Geosynchronous (GEO) and Low Earth Orbit (LEO) delays as shown in Figures 4.6 and 4.7 respectively. All sources are identical and persistent TCP sources i.e., the sources always send a segment as long as it is permitted by the TCP window. Moreover, traffic is unidirectional so that only the sources send data. The destinations only send ACKs. The delayed acknowledgment timer is deactivated, i.e., the receiver sends an ACK as soon as it receives a segment. Each TCP is transported over a single VC. This enables the switch to perform per-TCP control using per-VC control (Selective Drop). When multiple TCPs are aggregated over a single VC, per-TCP accounting cannot be performed and the buffer management within a single VC becomes equivalent to EPD or RED. Aggregated TCP VCs are further discussed in chapter 6.

We consider the following factors while performing our experiments:

- **TCP mechanism:** We use TCP Vanilla, Reno and SACK.

- **Round Trip Latency: GEO (550 ms) and single hop LEO (30 ms).** Our primary aim is to study the performance of large latency connections. The typical one-way latency from earth station to earth station for a single LEO (700 km altitude, 60 degree elevation angle) hop is about 5 ms [100]. The one-way latencies for multiple LEO hops can easily be up to 50 ms from earth station to earth station. GEO one-way latencies are typically 275 ms from earth station to earth station. For GEO's, the link between the two switches in Figure 4.6 is a satellite link with a one-way propagation delay of 275 ms. The links between the TCP sources and the switches are 1 km long. This results in a round trip propagation delay of about 550 ms. The LEO configuration is modeled as an access uplink to the on board satellite switch, one or more intersatellite hops, and a downlink to the earth terminal. For the set of LEO simulations presented in this section, a single intersatellite link is used. Each link has a propagation delay of 5 ms, resulting in an end to end round trip time of 30 ms (see figure 4.7). The LEO delays also fit the model illustrated in figure 2.11. In this case, the uplink delay, the downlink delay and the delay through the terrestrial ISP network, are all 5 ms each. Note that the LEO RTT modeled in this section is the same as in our WAN simulations. As a result, the LEO results illustrated in this section are the same as the WAN results. Multiple hop LEO simulations are illustrated in section 4.8.

- **Switch Buffer Size.** The buffer sizes used in the switch are 200,000 cells and 600,000 cells for GEO and 12,000 and 36,000 cells for LEO. These buffer sizes

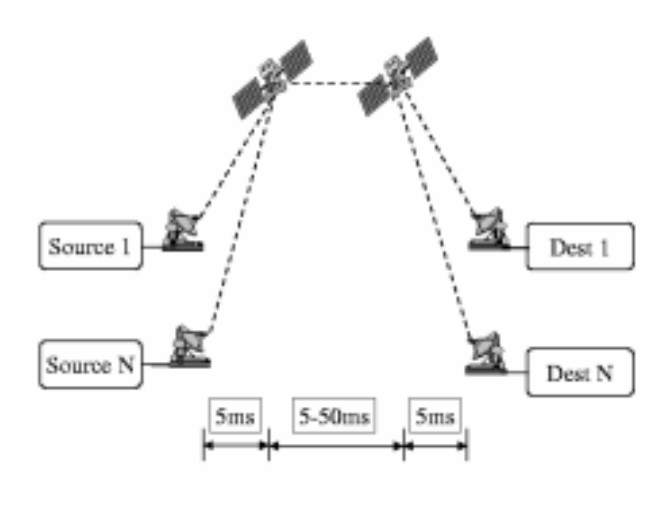


Figure 4.7: N-Source LEO Configuration

reflect approximate bandwidth-delay equivalents of 1 RTT and 3 RTTs respectively. Similar buffer sizes have been used in [61] for studying TCP performance over ABR and it is interesting to assess the performance of UBR in such situations. The relation between buffer sizes and round trip times is further explored in section 4.8.

- **Switch discard policy.** We use two discard policies, Early Packet Discard (EPD) and Selective Drop (SD).

We first present the results for LEO and GEO systems with the following parameters:

- The number of sources (N) is set to 5. In general, the typical number of simultaneous sources might be active, but our simulations give a good representation of the ability of the TCPs to recover during congestion. In section 4.8 we further extend these results to a large number of sources.

- Cells from all TCP sources share a single FIFO queue in the each outgoing link. The FIFO is scheduled according to link availability based on the data rate. In these experiments, no other traffic is present in the network. Cross traffic is introduced in the next chapter.
- The maximum value of the TCP receiver window is 8,704,000 bytes for GEO and 600,000 bytes for LEO (the window scale factor is used). These window size are sufficient to fill the 155.52 Mbps pipe.
- The TCP maximum segment size for GEO is 9180 bytes. A large value is used because most TCP connections over ATM with satellite delays are expected to use large segment sizes.
- The duration of simulation is 40 seconds. This is enough time for the simulations to reach steady state.
- All link bandwidths are 155.52 Mbps.
- The effects of channel access such as DAMA are ignored in our simulations. This simplifies the analysis and focuses on the properties of buffer management and end-system policies.

4.7.2 Simulation Results

Table 4.10 shows the efficiency values for TCP over UBR with 5 TCP sources. The table lists the efficiency values for three TCP types, 2 buffer sizes, 2 drop policies and the 2 round trip times. Several conclusions can be made from the table:

Result 4.17 Performance of SACK. For long delays, selective acknowledgments significantly improve the performance of TCP over UBR.

TCP Type	Buffer Size	Efficiency (LEO) EPD	Efficiency (LEO) SD	Efficiency (GEO) EPD	Efficiency (GEO) SD
SACK	1RTT	0.88	0.95	0.6	0.72
SACK	3RTT	0.99	0.99	0.99	0.99
Reno	1RTT	0.79	0.8	0.12	0.12
Reno	3RTT	0.75	0.77	0.19	0.22
Vanilla	1RTT	0.9	0.9	0.73	0.73
Vanilla	3RTT	0.81	0.81	0.81	0.82

Fast retransmit and recovery significantly degrades efficiency for long delay networks. SACK provides the best throughput.

Table 4.10: TCP over Satellite (UBR): Efficiency

For sufficient buffers, the efficiency values are typically higher for SACK than for Reno and vanilla TCP. This is because SACK often prevents the need for a timeout and can recover quickly from multiple packet losses. Under severe congestion, SACK can perform worse than Vanilla. This is because in these cases, retransmitted packets are dropped and the SACK sender experiences a timeout. As a result, all SACK information is discarded and the sender starts with a congestion window of 1. The lower efficiency is due to the bandwidth wasted in the aggressive fast retransmission due to SACK. Reduced SACK performance under severe congestion has also been reported in [37].

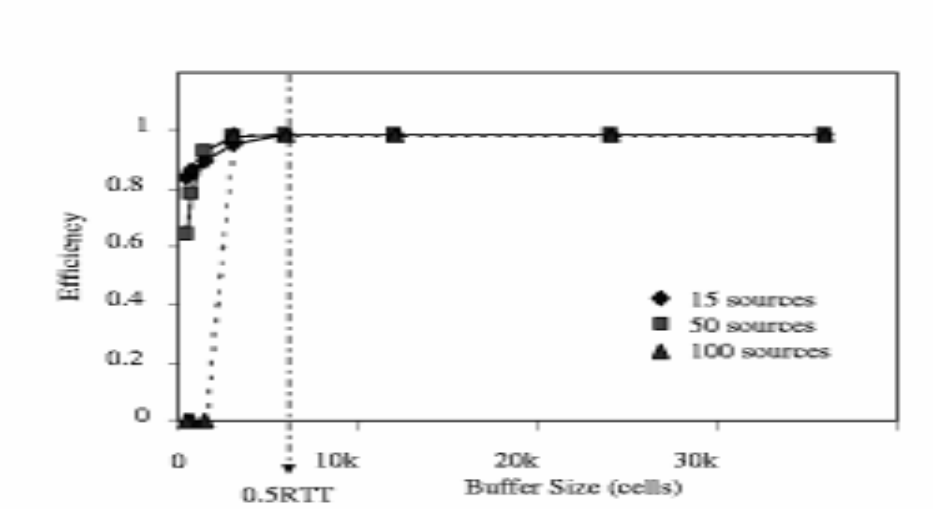
Result 4.18 Performance of fast retransmit and recovery. As delay increases, fast retransmit and recovery is detrimental to the performance of TCP.

The efficiency numbers for Reno TCP in table 4.10 are much lower than those of either SACK or Vanilla TCP. This problem with the fast retransmit and recovery algorithms in the presence of bursty packet loss was discussed in chapter 2. When

multiple packets are lost during a single round trip time (the same window), TCP Reno reduces its window by half for each lost packet. The reduced window size is not large enough to send new packets that trigger duplicate acks, resulting in a timeout. The timeout occurs at a very low window (because of multiple decreases during fast recovery) and the congestion avoidance phase triggers at a low window. For large RTT, the increase in window during congestion avoidance is very inefficient and results in much capacity being unused. For a large number of TCPs, the total throughput is greater, but our simulations reflect a worst case scenario and highlight the inefficiency of the congestion avoidance phase for large round trip times. This is an example of Reno's inability to accurately estimate the new steady state after a congestion episode. Vanilla TCP performs better, because the first packet loss triggers a timeout when the window is relatively large. The ensuing slow start phase quickly brings the window to half its original value before congestion avoidance sets in.

Result 4.19 Performance of buffer management. The effect of intelligent buffer management policies studied above is not significant in satellite networks.

We have shown that both EPD and Selective Drop improve the performance of TCP over UBR for LAN configurations. However, in these experiments, intelligent drop policies have little effect on the performance of TCP over UBR. The primary reason is that in our simulations, we have used adequate buffer sizes for high performance. Drop policies play a more significant role in improving performance in cases where buffers are a limited resource. These findings are further corroborated for WWW traffic in [37].



0.5RTT-bandwidth product is sufficient for high efficiency for TCP over UBR+.

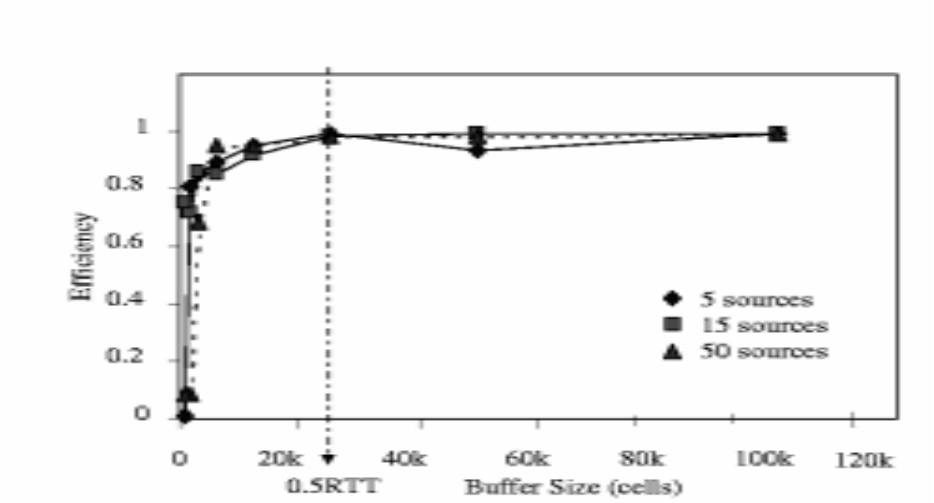
Figure 4.8: Buffer requirements for 30 ms RTT

4.8 Buffer Requirements for TCP over UBR

Our results have shown that small switch buffer sizes can result low TCP throughput over UBR. It is also clear that the buffer requirements increase with increasing delay-bandwidth product of the connections (provided the TCP window can fill up the pipe). However, the studies have not quantitatively analyzed the effect of buffer sizes on performance. *As a result, it is not clear how the increase in buffers affects throughput and what buffer sizes provide the best cost-performance benefits for TCP/IP over UBR.* In this section, we present simulation experiments to assess the buffer requirements for various satellite delays for TCP/IP over UBR.

4.8.1 Simulation Model

The N-source GEO and LEO configurations in figures 4.6 and 4.7 are used in these simulations. We study the effects of the following parameters:



0.5RTT-bandwidth product is sufficient for high efficiency for TCP over UBR+.

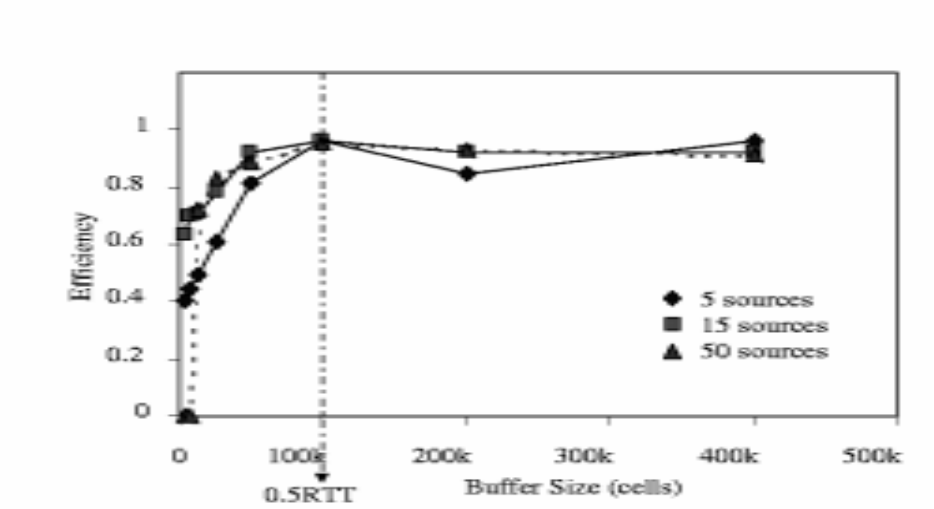
Figure 4.9: Buffer requirements for 120 ms RTT

- **Round trip latency.** In addition to GEO (550 ms round trip) and single hop LEO or WAN (30 ms round trip), we study a multi-hop LEO with a one way intersatellite delay of 50 ms. This results in a round trip time of 120 ms.

- **Number of sources.** To ensure that the results are scalable and general with respect to the number of connections, we use configurations with 5, 15 and 50 TCP connections on a single bottleneck link. For the single hop LEO configuration, we use 15, 50 and 100 sources.

- **Buffer size.** This is the most important parameter of this study. The set of values chosen are $2^{-k} \times$Round Trip Time (RTT), $k = -1..6$, (i.e., 2, 1, 0.5, 0.25, 0.125, 0.0625, 0.031, 0.016 multiples of the round trip delay-bandwidth product of the TCP connections.)

  The buffer sizes (in cells) used in the switch are the following:



0.5RTT-bandwidth product is sufficient for high efficiency for TCP over UBR+.

Figure 4.10: Buffer requirements for 550 ms RTT

- *LEO (30 ms)*: 375, 750, 1500, 3 K, 6 K, 12 K (=1 RTT), 24 K and 36 K.
- *Multiple LEO (120 ms)*: 780, 1560, 3125, 6250, 12.5 K, 50 K (=1 RTT) and 100 K.
- *GEO (550 ms)*: 3375, 6750, 12500, 25 K, 50 K, 100 K, 200 K (=1 RTT) and 400 K.

- **Switch drop policy.** We use the per-VC buffer allocation policy, Selective Drop to fairly allocate switch buffers to the competing connections.
- **End system policies.** We use SACK TCP for this study.
- **Start time:** All sources start at random times within the first 100 ms of the simulation. The previous simulations presented the worst case scenarios because all TCPs started at the same time. In this experiment, we are interested in buffer requirements that provide optimal throughput for a more realistic case.

The maximum values of the TCP receiver windows are 600000 bytes, 2500000 bytes and 8704000 bytes for single hop LEO, multi-hop LEO and GEO respectively. These window sizes are sufficient to fill the 155.52 Mbps links. The TCP maximum segment size is 9180 bytes. The duration of simulation is 100 seconds for multi-hop LEO and GEO and 20 secs for single hop LEO configuration. These are enough for the simulations to reach steady state. All link bandwidths are 155.52 Mbps and peak cell rate at the ATM layer is 149.7 Mbps after the SONET overhead.

We plot the buffer size against the achieved efficiency for different delay-bandwidth products and number of sources. The asymptotic nature of this graph provides information about the optimal buffer size for the best cost-performance tradeoff.

4.8.2 Simulation Results

Figures 4.8, 4.9 and 4.10 show the resulting TCP efficiencies for the 3 different latencies. Each point in the figures shows the efficiency (total achieved TCP throughput divided by maximum possible throughput) against the buffer size used. Each figure plots a different latency and each set of points (connected by a line) in a figure represents a particular value of N (the number of sources).

For very small buffer sizes, ($0.016\times$RTT, $0.031\times$RTT, $0.0625\times$RTT), the resulting TCP throughput is very low. In fact, for a large number of sources (N=50) , the throughput is sometimes close to zero. For moderate buffer sizes (less then 1 round trip delay-bandwidth), TCP throughput increases with increasing buffer sizes. TCP throughput asymptotically approaches the maximum value with further increase in buffer sizes. TCP performance over UBR for sufficiently large buffer sizes is scalable with respect to the number of TCP sources. The throughput is never 100%, but for

buffers greater than 0.5×RTT, the average TCP throughput is over 98% irrespective of the number of sources. Fairness (not shown here) is high for a large number of sources. This shows that SACK TCP sources with a good per-VC buffer allocation policy like selective drop, can effectively share the link bandwidth.

The knee of the buffer versus throughput graph is more pronounced for larger number of sources. For a large number of sources, TCP performance is very poor with small buffers, but jumps dramatically with sufficient buffering and then stays almost constant. For smaller number of sources, the increase in throughput with increasing buffers is more gradual.

For large round trip delays and a small number of sources, a buffer of 1 RTT or more can result in a slightly reduced throughput (see figures 4.9 and 4.10). This is because of the variability in the TCP retransmission timer value. When the round trip is of the order of the TCP timer granularity (100 ms in this experiment) and the queuing delay is also of the order of the round trip time, the retransmission timeout values become very variable. This may result in false timeouts and retransmissions thus reducing throughput.

Result 4.20 Buffer requirements for TCP over satellite. A buffer size of 0.5RTT at the bottleneck provides high efficiency and fairness to TCPs over UBR+ for satellite networks.

4.9 Chapter Summary

This chapter describes a set of techniques for improving the performance of TCP/IP over Asynchronous Transfer Mode (ATM) networks. Among the service categories provided by ATM networks, the most commonly used category for data traffic is

the Unspecified Bit Rate (UBR) service. UBR allows sources to send data into the network without any network guarantees or control.

Several issues arise in optimizing the performance of Transmission Control Protocol (TCP) when the ATM-UBR service is used over terrestrial and satellite networks. In this chapter, we studied several TCP mechanisms as well as ATM-UBR mechanisms to improve TCP performance over ATM networks. The UBR mechanisms that we studied in this chapter are:

- UBR with frame level discard policies,
- UBR with intelligent buffer management,

The following TCP mechanisms were studied:

- Vanilla TCP with slow start and congestion avoidance,
- TCP Reno with fast retransmit and recovery,
- TCP with selective acknowledgments (SACK)

We studied several combinations of these mechanisms using an extensive set of simulations and quantified the effect of each of these mechanisms. The following summarizes the list of conclusions drawn from our simulations:

Conclusion 4.1 (Buffer Requirements for zero loss) To achieve maximum possible throughput (or zero cell loss) for TCP over UBR, switches need buffers equal to the sum of the receiver windows of all the TCP connections.

Conclusion 4.2 (TCP over UBR) With limited buffer sizes, TCP performs poorly over vanilla UBR switches. TCP throughput is low and there is unfairness among the connections. The coarse granularity TCP timer is an important reason for low TCP throughput.

Conclusion 4.3 (Early Packet Discard) UBR with EPD improves the throughput performance of TCP. This is because partial packets are not being transmitted by the network and some bandwidth is saved. EPD does not have much effect on fairness because it does not drop segments selectively.

Conclusion 4.4 (Selective Drop) UBR with selective packet drop using per-VC accounting improves fairness over UBR+EPD. Connections with higher buffer occupancies are more likely to be dropped in this scheme. The efficiency values are similar to the ones with EPD.

Conclusion 4.5 (Fair Buffer Allocation) UBR with the Fair Buffer Allocation scheme improves TCP throughput and fairness. There is a tradeoff between efficiency and fairness and the scheme is sensitive to parameters. We found R = 0.9 and Z = 0.8 to produce best results for our configurations.

Conclusion 4.6 (Fast Retransmit and Recovery) Fast retransmit and recovery is detrimental to the performance of TCP over large delay-bandwidth links. This is because fast retransmit and recovery cannot effectively recover from multiple packet losses.

Conclusion 4.7 (Selective Acknowledgments) Selective Acknowledgments with TCP further improves the performance of TCP over UBR. SACK TCP results in better throughput than both vanilla and Reno TCP. The fairness and efficiency also increase with intelligent UBR drop policies.

Conclusion 4.8 (Long Latency) End-to-end policies have a more significant effect on large latency networks while drop policies have more impact on low latency networks.

Conclusion 4.9 (Number of Sources) SACK TCP with a per-VC accounting based buffer management policy like Selective Drop can produce high efficiency and fairness for TCP/IP over UBR even for a large number of TCP sources.

Conclusion 4.10 (Buffer Requirements) A buffer size equal to about half the round-trip delay-bandwidth product of the TCP connections provides high TCP throughput over satellite-UBR.

The results described above have been based on simulations using persistent TCP traffic. In [37], we have shown that the results also hold for world-wide web TCP traffic.

To summarize, TCP performance over UBR can be improved by either improving TCP using selective acknowledgments, or by introducing intelligent buffer management policies at the switches. Efficient buffer management has a more significant influence on LANs because of the limited buffer sizes in LAN switches compared to the TCP maximum window size. In WANs, the drop policies have a smaller impact because both the switch buffer sizes and the TCP windows are of the order of the bandwidth-delay product of the network. Also, the TCP linear increase is much slower in WANs than in LANs because the WAN RTTs are higher.

In this chapter, we have not presented a comprehensive comparative study of TCP performance with other per-VC buffer management schemes. This is a topic of further study and is beyond the scope of this chapter. Also, the present version of Selective Drop assigns equal weight to the competing connections. Selective Drop with weighted fairness is discussed in chapter 6.

CHAPTER 5

Guaranteed Rate: Effect of Higher priority Traffic

In the previous chapter, we have shown that intelligent drop policies can be used to enhance the performance of TCP over UBR. This enhanced version of UBR has been called UBR+. TCP performance over UBR+ can be degraded when high priority background traffic periodically uses up 100% of the link capacity. Providing a rate guarantee to the UBR+ class can ensure a continuous flow of TCP packets in the network. The Guaranteed Rate (GR) service provides such a guarantee to the UBR+ service category. In our proposal, guarantees are provided for the entire UBR+ class; per-VC guarantees are not provided. UBR+ with Guaranteed Rate requires no additional signaling requirements or standards changes, and can be implemented on current switches that support the UBR service. Another ATM service category called Guaranteed Frame Rate (GFR) has been proposed in [35]. The Guaranteed Frame Rate service provides per-VC rate guarantees to UBR+. This is more complex to implement and could significantly increase the cost of UBR+ switches. We discuss the GFR service in the next chapter.

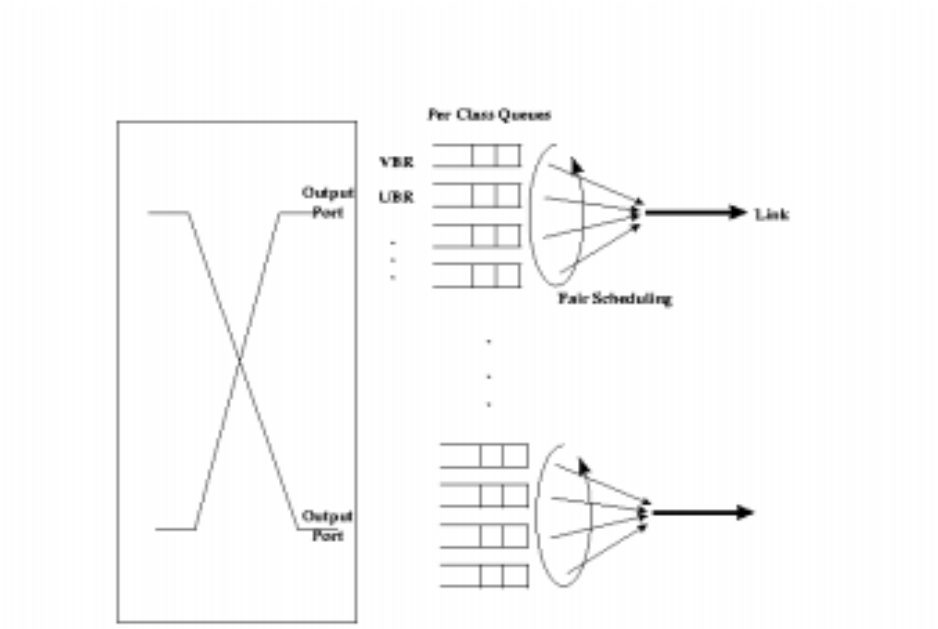
The Guaranteed Rate (GR) service is intended for applications that do not need any QoS guarantees, but whose performance depends on the availability of a continuous amount of bandwidth. GR guarantees a minimum rate to the UBR+ applications,

while maintaining the simplicity of the basic UBR+ service. This guarantee is made for the entire UBR+ class for each link in the switch. The goal of GR is to protect the UBR+ class from total bandwidth starvation and provide a continuous minimum bandwidth guarantee. In the presence of high load of higher priority Constant Bit Rate (CBR), Variable Bit Rate (VBR) and Available Bit Rate (ABR) traffic, TCP congestion control mechanisms are expected to benefit from a guaranteed minimum rate.

5.1 Chapter Goals

In this chapter, we discuss the implementation issues for providing guaranteed rates to UBR+ and analyze the performance of TCP over GR UBR+ in the presence of higher priority traffic. We first describe an architectural model for an ATM switch that supports multiple service categories. We describe the design of the GR service in the switch, based on the architectural model. We present simulation results that show how the performance of TCP over UBR+ can degrade in the presence of VBR, without any rate guarantees. We study the behavior of TCP over UBR+ with GR. Simulation results on the performance of TCP over UBR+ with and without GR are presented. The factors and their values used in our simulations are

1. TCP type
2. Round Trip Time
3. Buffer Size
4. Guaranteed Rate



The model assumes an output buffered switch with a single queue for each class. A fair scheduling mechanism provides a guaranteed rate to the UBR class.

Figure 5.1: Switch model for UBR+ with GR

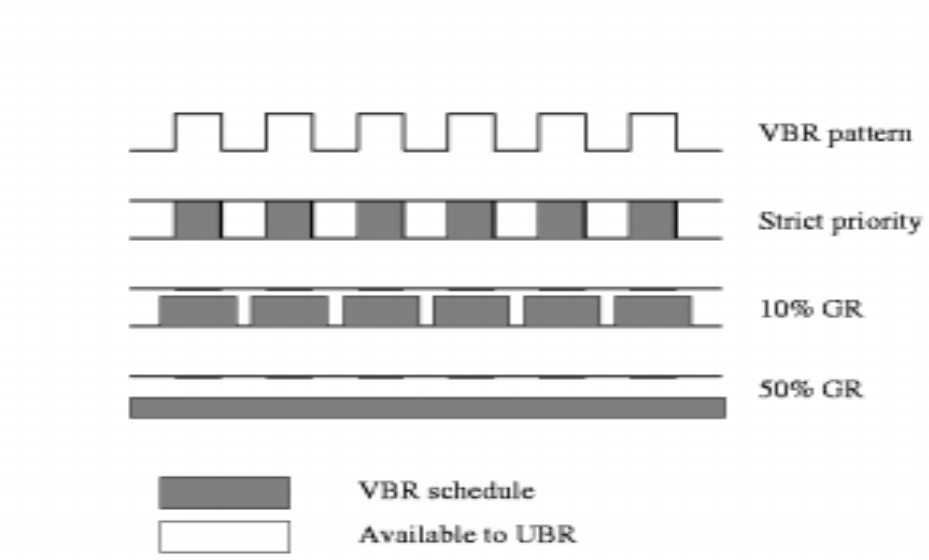
We analyze the effects of these factors using the analysis of variance technique from [57].

5.2 The UBR+ Guaranteed Rate Model

Our ATM switch model is output buffered, where each output port has a separate buffer for each service category. Figure 5.1 shows the switch model. The switch supports multiple service categories as shown in the figure. Each service category is provided with a bandwidth guarantee. In our examples, we consider only two classes – VBR and UBR. VBR typically has strict priority over UBR, but with GR, UBR is guaranteed a fraction (=GR) of the total link capacity.

To enforce a GR (as a fraction of the total link capacity), we perform fair scheduling among the queues on each port. Our fair scheduling algorithm ensures that when GR > 0.0, the UBR class is never starved, i.e., on the average, for every N cells transmitted on to the link, GR $\times$ N cells are from the UBR queue. This means that the VBR cells could be queued if the VBR connections are using more than (1−GR) of the link capacity. Any unused capacity by VBR is also allocated to UBR. The cell level minimum rate guarantee translates directly to a packet level guarantee for the TCP connections.

Figure 5.2 shows the link capacity allocations for three values of GR. There is a single VBR source with an on/off burst pattern. The VBR source uses up 100% of the link capacity during the on period and zero capacity during the off period. In the figure, the VBR on and off times are equal, so the average bandwidth requirements for VBR is 50% of the link capacity. When GR is 0, the VBR service is assigned strict priority over the UBR service. UBR is not guaranteed any rate and must use whatever capacity is left over by the VBR source. The VBR bursts are scheduled just as they arrive and VBR cells are not queued. When GR = 0.1, 10% of the link capacity is guaranteed to the UBR service class. This 10% must be shared by all the UBR connections going through the link. In this case, the VBR bursts may be queued in the VBR buffer to allow for UBR cells to be scheduled. As a result, the VBR bursts are flattened out with the VBR allocated Peak Cell Rate equal to 90% of the link capacity. Any link capacity unused by the VBR source is also available for UBR to use.



An on-off VBR arrival pattern may starve UBR traffic unless a guaranteed rate is provided to UBR. The figure shows the scheduling pattern of the VBR and UBR traffic as a fraction of the link capacity.

Figure 5.2: Link Capacity allocations for VBR and UBR with GR

When GR = 0.5, the VBR is further smoothed out so that it is now allocated a steady rate of 50% of the link capacity. On the average, the VBR queues are zero, but the on/off pattern results in temporary queues until the burst can be cleared out.

In each of the three GR allocations, VBR uses up only 50% of the link capacity. As a result, UBR can use up to the remaining 50%. The difference between the three configurations is the way in which UBR is given the 50% capacity. With GR = 0, UBR is starved for the time VBR is using up 100% of the link. With positive values of GR, UBR is guaranteed a continuous flow of bandwidth and is never completely starved.

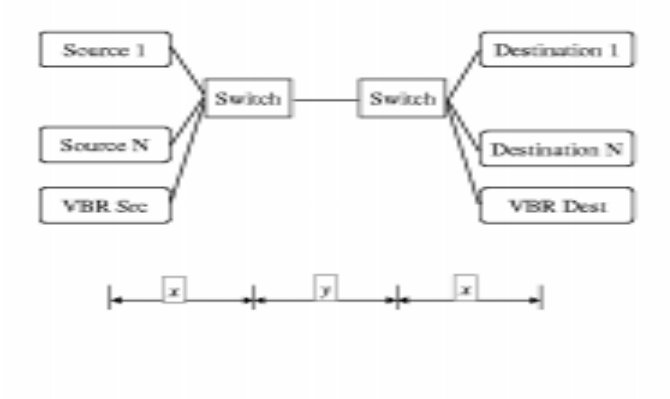


Figure 5.3: The N source TCP configuration with VBR

In this work, we experiment with a per-port bandwidth guarantee for UBR. The study of UBR with per-VC rate guarantees is a subject of the next chapter.

5.3 TCP over UBR+ with VBR background

We first introduce a Variable Bit Rate source into our N-source configuration and observe the performance of TCP with three different on-off periods.

5.3.1 Simulation Model

All simulations use the N-source VBR configuration shown in figure 5.3. Note the presence of an additional VBR source shown in the figure. The VBR source is also an end-to-end VBR source like the other TCP connections. All TCP sources are identical and infinite TCP sources. The TCP layer always sends a segment as long as it is permitted by the TCP window. Moreover, traffic is unidirectional so that only the sources send data. The destinations only send ACKs.

Link delays are 5 microseconds for LAN configurations and 5 milliseconds for WAN configurations. This results in a round trip propagation delay of 30 microseconds for

LANs and 30 milliseconds for WANs respectively. For GEO satellite configurations, the propagation delay between the two switches is 275 milliseconds and the distance between the TCPs and the switches is 1 km. The round trip propagation delay for GEO satellite networks is about 550 milliseconds.

The TCP segment size is set to 512 bytes for LAN and WAN configurations. This is the common segment size used in most current TCP implementations. For satellite networks, we use a segment size of 9180 bytes. For the LAN configurations, the TCP maximum window size is limited by a receiver window of 64K bytes. This is the default value specified for TCP implementations. For WAN configurations, a window of 64K bytes is not sufficient to achieve 100% utilization. We thus use the window scaling option to specify a maximum window size of 600000 Bytes. For satellite configurations, this value is further scaled up to 8704000 Bytes.

All link bandwidths are 155.52 Mbps and Peak Cell Rate at the ATM layer is 149.7 Mbps. The duration of the simulation is 10 seconds for LANs, 20 seconds for WANs and 40 seconds for satellites. This allows for adequate round trips for the simulation to give stable results.

5.3.2 Simulation Results

When higher priority VBR traffic is present in the network, TCP over UBR may get considerably lower link capacity than without VBR. Moreover, the presence of VBR traffic could result in the starvation of UBR traffic for periods of time for which VBR uses up the entire link capacity. When VBR has strict priority over UBR, TCP (over UBR) traffic is transmitted in bursts and the round trip time estimates for the TCP connection are highly variable. An underestimation of the RTT may cause a

Config-uration	Buffer (cells)	VBR period (ms)	UBR	EPD	Selective Drop
LAN	1000	300	0.71	0.88	0.98
LAN	3000	300	0.83	0.91	0.92
LAN	1000	100	0.89	0.97	0.95
LAN	3000	100	0.96	0.95	0.96
LAN	1000	50	0.97	0.93	0.93
LAN	3000	50	0.95	0.97	0.97
WAN	12000	300	0.42	0.43	0.61
WAN	36000	300	0.55	0.52	0.96
WAN	12000	100	0.72	0.58	0.70
WAN	36000	100	0.95	0.97	0.97
WAN	12000	50	0.97	0.65	0.73
WAN	36000	50	0.97	0.98	0.98

Longer VBR on-off periods result in low throughput, especially in WANs.

Table 5.1: SACK TCP with VBR (strict priority) : Efficiency

false timeout in the TCP indicating congestion even though the TCP packet is queued behind a VBR burst. An overestimation of the RTT may result in much time being wasted waiting for a timeout when a packet is dropped due to congestion.

The effect of UBR starvation is seen in table 5.1. In this set of simulations, we used five source LAN and WAN configurations with SACK TCP. Three different VBR on/off periods were simulated – 300ms, 100ms and 50ms. In each case, the on times were equal to the off times and during the on periods, the VBR usage was 100% of the link capacity. VBR was given strict priority over UBR, i.e., GR for UBR was 0.

From the tables we can see that longer VBR bursts (for the same average VBR usage of 50%) result in lower throughput for TCP over UBR+.

For the WAN configuration, the effect of VBR frequency is very clear from the table. The performance is good for lower VBR on/off frequencies (50 and 100 ms).

With 300 ms VBR, TCP performance for WANs is poor. This is because, the VBR burst time is of the order of the TCP timeout value (2 to 3 ticks of 100 ms each). As a result the TCP source is starved long enough that a retransmission timeout occurs. Much time (several roundtrips of at least 30 ms each) is then wasted in recovering from the timeout during the slow start phase. This causes poor utilization of the link and lower efficiency values. When VBR on/off times are smaller compared to the retransmission timeout value, the VBR delay is not enough to cause a TCP timeout and higher throughput results.

For LANs, the above argument also holds, but other factors are more dominant. The LAN results show that the effects of the switch drop policy and the buffer size are also important. The selective drop policy significantly improves the LAN performance of TCP over UBR+. This is because the round trip time is very small and even during the congestion avoidance phase, the recovery is very fast. The TCP timeouts are often in phase with the VBR burst times. As a result, when TCP is waiting for the timer to expire and not utilizing the link, VBR is using the link at 100% capacity. When TCP times out and starts to send segments, the congestion window increases very fast. Also packets that were not dropped but queued behind the VBR burst are not always retransmitted because the receiver TCP performs out of order caching.

Result 5.1 Effect of Higher Priority Traffic on TCP. When strict priority traffic starves TCP traffic, throughput may be degraded.

The effect is more severe for longer starvation periods. The effect is also more significant for long delay terrestrial networks when the starvation period is comparable to the length of the retransmission timeout.

5.4 Guaranteed Rate

We now experiment with the guaranteed rate model to improve the performance of TCP. In our experiments, we vary the following parameters.

1. **TCP type**: Vanilla, Reno and SACK
2. **Round Trip Time**: LAN, WAN and GEO
3. **Buffer Size**: 1RTT and 3RTT. Buffer size = 1000 cells and 3000 cells for LANs, 12000 cells and 36000 cells for WANs; and 200000 cells and 600000 cells for satellites.
4. **Drop Policy**: Tail Drop (UBR), EPD and SD.
5. **Guaranteed Rate**: 0%, 10% and 50% of the link capacity.

5.4.1 Simulation Results

The tables in Appendix C list the complete results of the simulations.

To quantify the effect of each factor in our experiment we use the allocation of variation technique described in [57]. Only a brief description of the methodology is presented here. For a complete description, the reader is referred to [57].

The goal of analyzing results the experiments is to calculate the individual effects of contributing factors and the interactions between the factors. These effects can also help us in drawing meaningful conclusions about the optimum values for different factors. In our experiments, we analyze the effects of the TCP flavors, buffer sizes, drop policies and guaranteed rate in determining the efficiency and fairness for LAN, WAN and GEO links. In this experiment, there were 4 factors – Drop policy (A),

TCP flavor (B), switch buffer (C) and guaranteed rate (C). The values a factor can take are called *levels* of the factor. For example, EPD and SD are two levels of the factor *Drop Policy*. For factors A, B, C and D, the levels are indicated by i, j, k and l respectively. Each simulation corresponds to a combination of the levels of the factors. A *full factorial* experiment calculates performance results (in our case efficiency and fairness) for all such combinations. In this case, the total number of experiments is 162. 54 experiments were run for each configuration (LAN, WAN and GEO) for a total of 162 simulations. The analysis is done separately for LAN, WAN and GEO configurations. Each experiment set contains 54 experimental values of efficiency and fairness each.

We assume an additive model given by the following equation:

$$y = \mu + \alpha_i + \beta_j + \zeta_k + \delta_l + \gamma_{ij} + \gamma_{ik} + \gamma_{il} + \gamma_{jk} + \gamma_{jl} + \gamma_{kl} + \gamma_{ijk} + \gamma_{ikl} + \gamma_{jkl} + \gamma_{ijkl} + \epsilon_{ijkl}$$

The model (y) consists of the sum of the mean response (μ), 4 main effects ($\alpha_i, \beta_j, \zeta_k$ and δ_l), 6 first order interactions ($\gamma_{ij}, \gamma_{ik}, \gamma_{il}, \gamma_{jk}, \gamma_{jl}$ and γ_{kl}), 3 second order interactions ($\gamma_{ijk}, \gamma_{ikl}, \gamma_{jkl}$ and γ_{ijkl}), 1 third order interaction (γ_{ijkl}) and an experimental error term (ϵ_{ijkl}). We assume that only first order interactions are significant; second and third order interactions are ignored.

We calculate the following quantities:

Observation or response, (y_{ijkl}). This is the observed value of efficiency or fairness from an experiment with the levels of individual factors as i, j, k and l respectively.

Sum of squares of responses, (SSY). This is the sum of squares of the individual results above.

Sum of squares of the overall mean, (SS0). This consists of the calculation of the overall mean, $\bar{y}$, of the results y_{ijkl} and multiplying its square by the total number of experiments.

Total variation, (SST). This represents the variation in the result values (efficiency or fairness) around the overall mean.

$$SST = SSY - SS0.$$

Sum of squares of main effects, (SSA, SSB, SSC and SSD). The main effects given by $\alpha_i, \beta_j, \zeta_k$ and δ_l are the individual contributions of a level of each factor (A, B, C and D) to the overall result. A particular main effect is associated with a level of a factor and indicates how much variation around the overall mean is caused by the level.

$$\begin{aligned} \alpha_i &= \bar{y}_{.jkl} - \mu \\ \beta_j &= \bar{y}_{i.kl} - \mu \\ \zeta_k &= \bar{y}_{ij.l} - \mu \\ \delta_l &= \bar{y}_{ijk.} - \mu \end{aligned}$$

$$\begin{aligned} SSA &= bcd \times \sum_i {\alpha_i}^2 \\ SSB &= acd \times \sum_j {\beta_j}^2 \\ SSC &= abd \times \sum_k {\zeta_k}^2 \\ SSD &= abc \times \sum_l {\delta_l}^2 \end{aligned}$$

where a, b, c and d are the number of levels of factors A, B, C and D respectively.

First order interactions, (SSAB, SSAC ...). These are the interactions between levels of two factors. In the example, there are first order interactions between each TCP flavor and buffer size, between each drop policy and TCP flavor, between each buffer size and drop policy, TCP flavor and guaranteed rate and so on. For example, the first order interaction term between drop policy (A) and TCP flavor (B) is given by:

$$\gamma_{ij} = \bar{y}_{ij..} - \bar{y}_{i...} - \bar{y}_{.j..} - \mu$$

$$SSAB = cd \times \sum_{i,j} (\gamma_{ij})^2$$

Sum of Squares of overall standard error, (SSE). This represents the experimental error associated with each result value. The overall standard error is also used in the calculation of the confidence intervals for each effect.

$$\begin{aligned} SSE &= SSY - SS0 - SSA - SSB - SSC - SSD - SSAB - SSAC - \\ &\quad SSAD - SSBC - SSBD - SSCD \end{aligned}$$

Allocation of variation. This is used to explain how much each effect contributes to the total variation (SST).

$$\begin{aligned} SST &= SSA + SSB + SSC + SSD + SSAB + SSAC + \\ &\quad SSAD + SSBC + SSBD + SSCD + SSE \end{aligned}$$

Each term on the right of the above equation contributes to the total variation. An effect (a factor or interaction), which explains a large fraction of the total variation, is said to be important.

	LAN		WAN		GEO	
Component	Value	% of SST	Value	% of SST	Value	% of SST
SST	2.51	100	1.27	100	3.31	100
SSY	29.00		38.57		29.50	
SS0	26.48		37.30		26.19	
SSA	1.12	44.8	0.03	2.77	0.15	4.67
SSB	0.21	8.44	0.02	2.03	2.45	74.17
SSC	0.58	23.10	0.37	29.21	0.09	2.78
SSD	0.06	2.60	0.52	41.28	0.09	2.74
SSAB	0.05	2.10	0.01	1.14	0.21	6.58
SSAC	0.05	1.99	0.008	0.6	0.002	0.06
SSAD	0.04	1.91	0.01	1.12	0.02	0.76
SSBC	0.08	3.30	0.03	2.80	0.02	0.81
SSBD	0.02	0.82	0.09	7.20	0.09	2.89
SSCD	0.04	1.84	0.09	7.44	0.02	0.83
SSE	0.22	9.00	0.05	4.00	0.12	5.00

In LANs, drop policy and buffer size (A and C) explain most of the variation. In WANs, guaranteed rate (D) is the most important factor, while buffer size also explains some variation. For GEO satellites, TCP flavor (B) is the most important factor in the allocation of variation.

Table 5.2: Allocation of Variation:Efficiency

Confidence intervals for main effects. The 90% confidence intervals for each main effect are calculated. If a confidence interval contains 0, then the corresponding level of the factor is not statistically significant. If confidence intervals of two levels overlap, then the effects of both levels are assumed to be similar.

Tables 5.2 and 5.3 show the results of the allocation of variation for efficiency and fairness respectively. The results show that the model is applicable to efficiency because most of the variation is explained by the main effects and the interactions. The fairness is not explained by the effects or the interactions and so the model does

	LAN		WAN		GEO	
Component	Value	% of SST	Value	% of SST	Value	% of SST
SST	2.71	100	0.57	100	0.43	100
SSY	29.83		42.60		47.31	
SS0	27.12		42.02		46.87	
SSA	0.52	19.4	0.01	2.01	0.00	0.94
SSB	0.06	2.50	0.07	12.57	0.03	7.05
SSC	0.78	28.74	0.05	8.82	0.04	9.38
SSD	0.48	17.81	0.02	4.33	0.00	0.27
SSAB	0.08	3.13	0.03	5.19	0.008	1.92
SSAC	0.008	0.29	0.00	0.16	0.02	6.58
SSAD	0.04	1.6	0.03	5.41	0.03	8.10
SSBC	0.03	1.20	0.00	0.10	0.01	2.64
SSBD	0.06	2.35	0.07	12.67	0.01	3.22
SSCD	0.10	4.03	0.05	9.04	0.01	3.30
SSE	0.51	19.4	0.22	39.00	0.24	56.00

In LANs, the buffer size (C) and the drop policy (A) explain some variation. The experimental error is high and the results cannot explain the effect of the factors on fairness.

Table 5.3: Allocation of Variation:Fairness

not give us any information about the effects of the factors on fairness. The following discussions are about the efficiency metric.

For LANs, the most important factors are the drop policy that explains 44% of the variation and the buffer size which explains 23% of the variation. The results show that large buffer sizes with selective drop produce the best efficiency. For WAN, the most important factors are the buffer size (41% of variation) and the TCP type (29% of variation). Large buffer and SACK produce the best performance. For GEO, TCP type is the most important factor (explains 74% of the variation). SACK provides the best performance. The interactions between the factors are insignificant.

The following results can be summarized for the TCP over UBR+ with GR in the presence of high priority background traffic.

Result 5.2 LAN Performance. For LANs, the dominating factors that effect the performance are the switch drop policy and the buffer size.

The selective drop policy improves the performance irrespective of most TCP and GR parameters. This result holds with or without the presence of background VBR traffic. In LANs, the switch buffer sizes are of the order of 1000 and 3000 cells. This is very small in comparison with the maximum TCP receiver window. As a result, TCP can easily overload the switch buffers. This makes buffer management very important for LANs.

Result 5.3 WAN Performance. For WANs, the dominating factor is the GR. A GR of 0 hurts the TCP performance.

GR values of 0.5 and 0.1 produce the highest efficiency values. A constant amount of bandwidth provided by GR ensures that TCP keeps receiving ACKs from the

destination. This reduces the variability in the round trip times. Consequently, TCP is less likely to timeout. Buffer management policies do have an impact on TCP performance over WANs, but the effect is less than in LANs. This is because the buffer sizes of WAN switches are comparable to the bandwidth $\times$ round trip delays of the network. The TCP maximum windows are also usually based on the round trip times. As a result, buffers are more easily available and drop policies are less important.

The results for LAN and WAN confirm our intuition based on table 5.1. We saw that without GR, VBR had the most effect on the WAN configuration. The performance of TCP with strict priority VBR suffered most in this case. As a result, there was more room for improvement in performance using GR. In LAN, selective drop was sufficient to improve performance to a near optimal level.

Result 5.4 Satellite Performance. For satellite networks, the TCP congestion control mechanism makes the most difference; SACK TCP produces the best results and Reno TCP results in the worst performance.

SACK TCP ensures quick recovery from multiple packet losses, whereas fast retransmit and recovery is unable to recover from multiple packet drops. The satellite buffer sizes are quite large and so the drop policies do not make a significant difference. The GR fractions do not significantly affect the TCP performance over satellite networks because in our simulations, the VBR burst durations are smaller than the round trip propagation delays. The retransmission timeout values are typically close to 1 second and so a variation of the RTT by 300 milliseconds can be tolerated by the TCP. GR may have more impact on satellite networks in cases where UBR is starved for times

larger than the retransmission timeout value of the connection. However, a VBR on-off period of more than 1 sec is not a realistic model.

5.5 Chapter Summary

In this chapter, we examined the effect of higher priority VBR traffic on the performance of TCP over UBR+. Several factors can effect the performance of TCP over UBR in the presence of higher priority VBR traffic. These factors include:

- The propagation delay of the TCP connection.
- The TCP congestion control mechanisms.
- The UBR switch drop policies.
- The Guaranteed Rate provided to UBR.

For large propagation delays, end-to-end congestion control is the most important factor. For small propagation delays, the limited switch buffers makes buffer management very important. A minimum bandwidth guarantee improves TCP performance over UBR when the TCP connection may be starved for periods longer than the round trip propagation delay. The minimum bandwidth scheme explored here provides a minimum rate to the entire UBR class on the link. Per-VC mechanisms are explored in the next chapter.

CHAPTER 6

Guaranteed Frame Rate: Providing per-VC Minimum Rate Guarantees

In current internetworking architectures, enterprise networks are interconnected to each other, or to their service providers via backbone ATM VCs. Most ATM networks provide best effort UBR connections for TCP/IP traffic. The ATM Guaranteed Frame Rate (GFR) service is a best effort service that provides minimum rate guarantees to ATM VCs. Edge devices connecting IP LANs to an ATM network can use GFR VCs to transport TCP/IP traffic. Buffer management techniques are essential to the realization of a robust GFR implementation. In this chapter, we show how rate guarantees can be provided to VCs carrying TCP traffic using buffer management on a FIFO buffer. We present a buffer management scheme called Differential Fair Buffer Allocation (DFBA) that provides Minimum Cell Rate (MCR) guarantees to Guaranteed Frame Rate (GFR) VCs carrying TCP/IP traffic. DFBA allocates buffer space in proportion to MCR and probabilistically drops TCP packets to control congestion and maintain MCR. DFBA can be used on a FIFO buffer shared by several VCs. Each VC can carry traffic from one or more TCP connections.

6.1 Chapter Goals

In this chapter, we present:

- An overview of the GFR service and its implementation options.
- Issues in providing minimum rate guarantees to TCP traffic with FIFO buffers.
- A buffer management scheme for MCR guarantees to VCs with aggregate TCP flows.

We begin in section 6.2 with a description of the GFR service category and propose implementation options for GFR. Section 6.3 discusses results of a study on the behavior of TCP traffic with controlled windows. This provides insight into controlling TCP rates by controlling TCP windows. Section 6.4 describes the effect of buffer occupancy and thresholds on TCP throughput. The focus of these sections is to present empirical simulation based analysis of intuitive ideas on controlling TCP rates using buffer management. Section 6.5 presents a dynamic threshold-based buffer management policy to provide TCP throughputs in proportion to buffer thresholds for per-VC rate allocations. This scheme assumes that each GFR VC may carry multiple TCP connections. We then present simulation results with TCP traffic over LANs interconnected by an ATM network.

6.2 The Guaranteed Frame Rate Service

Guaranteed Frame Rate (GFR) has been recently proposed in the ATM Forum as an enhancement to the UBR service category. Guaranteed Frame Rate will provide a minimum rate guarantee to VCs at the frame level. The GFR service also allows for the fair usage of any extra network bandwidth. GFR requires minimum signaling and

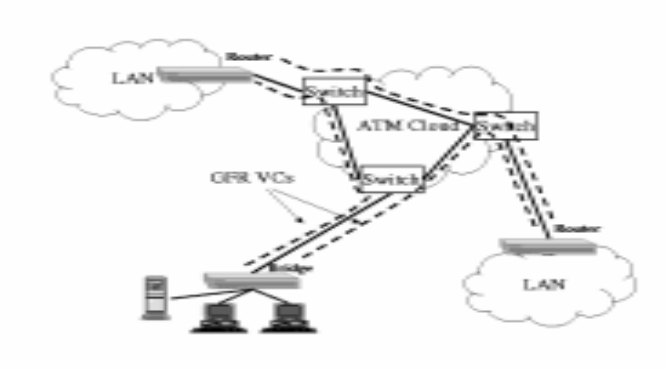


Figure 6.1: Use of GFR in ATM connected LANs

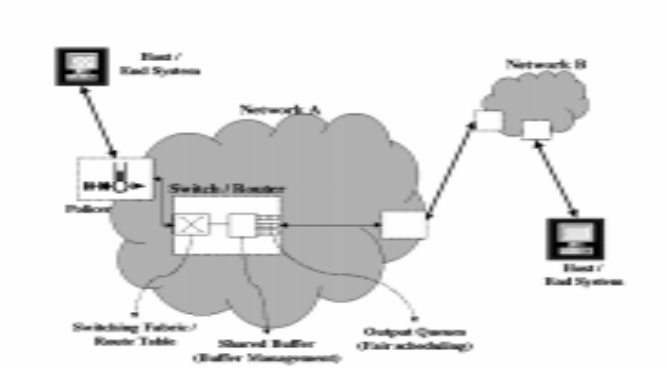
connection management functions and depends on the network's ability to provide a minimum rate to each VC. GFR is likely to be used by applications that can neither specify the traffic parameters needed for a VBR VC, nor have the capability for ABR (for rate based feedback control). Current internetworking applications fall into this category and are not designed to run over QoS based networks. These applications could benefit from a minimum rate guarantee by the network, along with an opportunity to fairly use any additional bandwidth left over from higher priority connections. In the case of LANs connected by ATM backbones, network elements outside the ATM network could also benefit from GFR guarantees. For example, IP routers separated by an ATM network could use GFR VCs to exchange control messages. Figure 6.1 illustrates such a case where the ATM cloud connects several LANs and routers. ATM end systems may also establish GFR VCs for connections that can benefit from a minimum throughput guarantee.

The original GFR proposals [43, 42] give the basic definition of the GFR service. GFR provides a minimum rate guarantee to the **frames** of a VC. The guarantee requires the specification of a maximum frame size (MFS) of the VC. If the user

sends packets (or frames) smaller than the maximum frame size, at a rate less than the minimum cell rate (MCR), then all the packets are expected to be delivered by the network with minimum loss. If the user sends packets at a rate higher than the MCR, it should still receive at least the minimum rate. The minimum rate is guaranteed to the untagged (CLP=0) frames of the connection. In addition, a connection sending in excess of the minimum rate should receive a fair share of any unused network capacity. The exact specification of the fair share has been left unspecified by the ATM Forum. Although the exact GFR specification has some more detail, the above discussion captures the essence of the service.

There are three basic design options that can be used by the *network* to provide the per-VC minimum rate guarantees for GFR – tagging, buffer management and queuing:

1. **Tagging:** *Network based tagging* (or policing) can be used as a means of marking non-eligible packets before they enter the network. This form of tagging is usually performed when the connection enters the network. Figure 6.2 shows the role of network based tagging in providing a minimum rate service in a network. Network based tagging on a per-VC level requires some per-VC state information to be maintained by the network and increases the complexity of the network element. Tagging can isolate eligible and non-eligible traffic of each VC so that other rate enforcing mechanisms can use this information to schedule the eligible traffic in preference to non-eligible traffic. In a more general sense, policing can be used to discard non-eligible packets, thus allowing only eligible packets to enter the network.



GFR can be implemented by a combination of tagging, buffer management and scheduling mechanisms.

Figure 6.2: Network Architecture with tagging, buffer management and scheduling

2. **Buffer management:** Buffer management is typically performed by a network element (like a switch or a router) to control the number of packets entering its buffers. In a shared buffer environment, where multiple VCs share common buffer space, per-VC buffer management can control the buffer occupancies of individual VCs. Per-VC buffer management uses per-VC accounting to keep track of the buffer occupancies of each VC. Figure 6.2 shows the role of buffer management in the connection path. Examples of per-VC buffer management schemes are Selective Drop and Fair Buffer Allocation. Per-VC accounting introduces overhead, but without per-VC accounting it is difficult to control the buffer occupancies of individual VCs (unless non-conforming packets are dropped at the entrance to the network by the policer). Note that per-VC buffer management uses a single FIFO queue for all the VCs. This is different from per-VC queuing and scheduling discussed below.

3. **Scheduling:** Figure 6.2 illustrates the position of scheduling in providing rate guarantees. While tagging and buffer management control the entry of packets into a network element, queuing strategies determine how packets are scheduled onto the next hop. FIFO queuing cannot isolate packets from various VCs at the egress of the queue. As a result, in a FIFO queue, packets are scheduled in the order in which they enter the buffer. Per-VC queuing, on the other hand, maintains a separate queue for each VC in the buffer. A scheduling mechanism can select between the queues at each scheduling time. However, scheduling adds the cost of per-VC queuing and the service discipline. For a simple service like GFR, this additional cost may be undesirable.

A desirable implementation of GFR is to use a single queue for all GFR VCs and provide minimum rate guarantees by means of intelligent buffer management policies on the FIFO. Several proposals have been made to provide rate guarantees to TCP sources with FIFO queuing in the network [9, 14, 38] . The bursty nature of TCP traffic makes it difficult to provide per-VC rate guarantees using FIFO queuing. These proposals recommend the use of per-VC queuing and scheduling to provide rate guarantees to TCP connections. However, all these studies were performed at high target network utilization, i.e., most of the network capacity was allocated to the MCRs. Moreover, these proposals are very aggressive in dropping TCP packets causing TCP to timeout and lose throughput.

All the previous studies have examined TCP traffic with a single TCP per VC. Per-VC buffer management for such cases reduces to per-TCP buffer management. However, routers that use GFR VCs, will multiplex many TCP connections over a single VC. For VCs with several aggregated TCPs, per-VC control is unaware of

each TCP in the VC. Moreover, aggregate TCP traffic characteristics and control requirements may be different from those of single TCP streams.

6.3 TCP Behavior with Controlled Windows

TCP uses a window based mechanism for flow control. The amount of data sent by a TCP connection in one round trip is determined by the window size of the TCP connection. The window size is the minimum of the sender's congestion window (CWND) and the receiver's window (RCVWND). TCP rate can be controlled by controlling the window size of the TCP connection.

However, a window limit is not enforceable by the network to control the TCP rate. The only form of control available to the network is to drop TCP packets. TCP sources respond to packet loss by reducing the source congestion window by one-half and then increasing it by one segment size every round trip. As a result, the average TCP window can be controlled by intelligent packet discard.

For TCP window based flow control, the throughput (in Mbps) can be calculated from the average congestion window (in bytes) and the round trip time (in seconds) as:

$$\text{Throughput (Mbps)} = \frac{8 \times 10^{-6} \times \text{CWND}_{\text{avg}}}{\text{Round Trip Time}} \tag{6.1}$$

Where CWND_{avg} is the average congestion window in bytes and Round Trip Time is in seconds. The factor 8×10^{-6} converts the throughput from bytes per sec to Megabits per sec.

Suppose the network capacity allows the TCP window to increase to CWND_{max}, at which point TCP detects a packet loss and reduces its window to $\text{CWND}_{\text{max}}/2$. The window then increases linearly to CWND_{max} when a packet is dropped again.

The average TCP CWND during the linear increase phase can be calculated as:

$$\text{CWND}_{\text{avg}} = \frac{\sum_{i=1}^{T} \text{CWND}_{\text{max}}/2 + \text{MSS} \times i}{\text{T}}$$

where T is the number of round trip times for the congestion window to increase from $\text{CWND}_{\text{max}}/2$ to CWND_{max}. Since CWND increases by 1 MSS every RTT, $\text{T} = \text{CWND}_{\text{max}}/(MSS * 2)$. The average window simply reduces to

$$\text{CWND}_{\text{avg}} = 0.75 \times \text{CWND}_{\text{max}} \tag{6.2}$$

Note that this equation assumes that during the linear increase phase, the TCP window increases by one segment every round trip time. However, when the TCP delayed acknowledgment option is set, TCP might only send an ACK for every two segments. In this case, the window would increase by 1 segment every 2 RTTs.

Figure 6.3 shows how the source TCP congestion window varies when a single segment is lost at a particular value of the congestion window. The figure is the CWND plot of the simulation of the configuration shown in Figure 6.4 with a single SACK TCP source (N=1). The figure shows four different values of the window at which a packet is lost. The round trip latency (RTT) for the connection is 30 ms. The window scale factor is used to allow the TCP window to increase beyond the 64K limit.

From Figure 6.3 and equation 6.2, the average congestion windows in the linear phases of the four experiments are approximately 91232 bytes, 181952 bytes, 363392 bytes and over 600000 bytes. As a result, the average calculated throughputs from equation 6.1 are 24.32 Mbps, 48.5 Mbps, 96.9 Mbps and 125.6 Mbps (126 Mbps is the maximum possible TCP throughput for a 155.52 Mbps link with 1024 byte TCP segments). The empirical TCP throughputs obtained from the simulations of the four

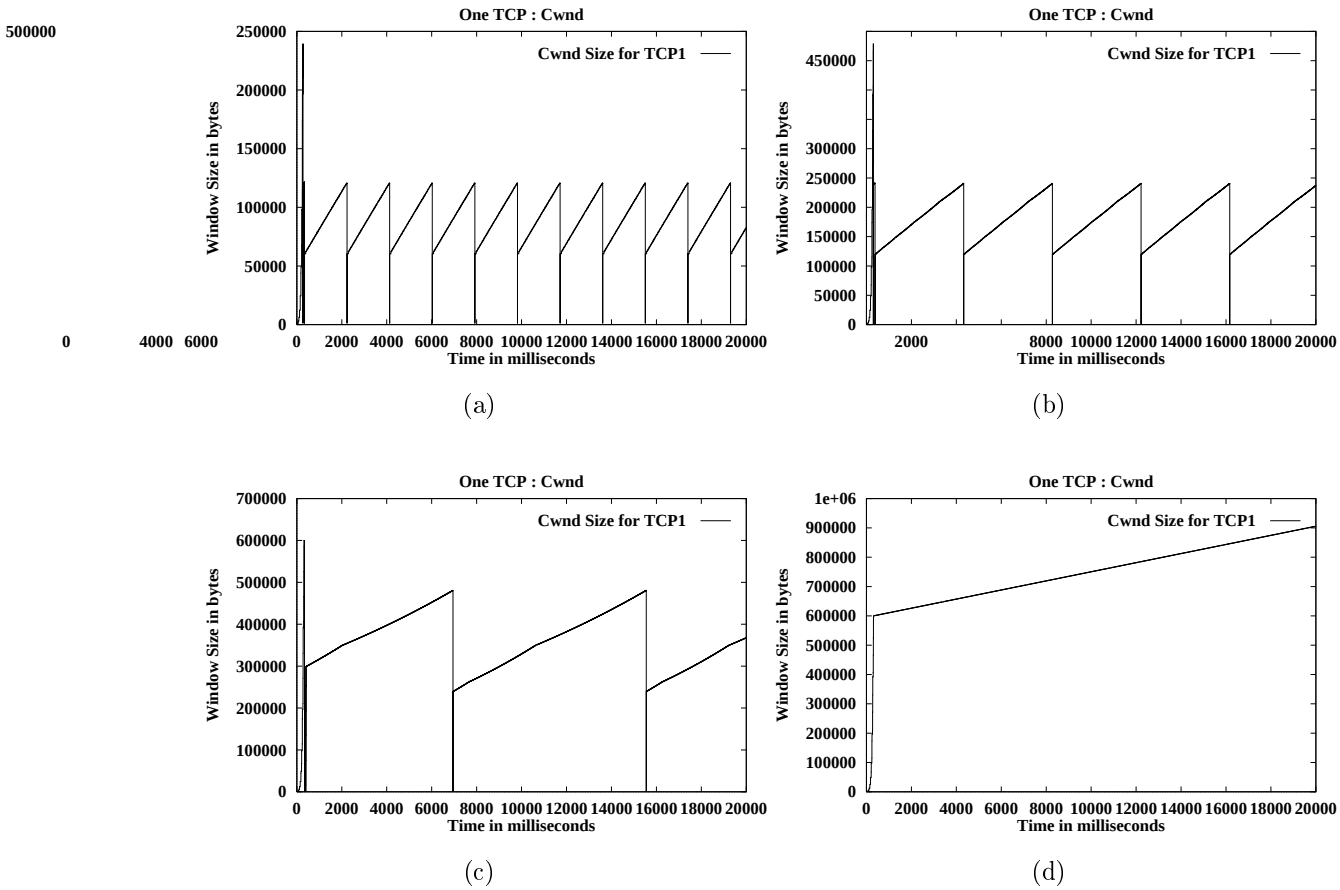


Figure 6.3: Single TCP Congestion Window Control. Drop thresholds (bytes of window size) = 125000, 250000, 500000, None

cases are 23.64 Mbps, 47.53 Mbps, 93.77 Mbps and 125.5 Mbps respectively. The throughput values calculated from equation 6.2 are very close to those obtained by simulation. This shows that controlling the TCP window so as to maintain a desired average window size enables the network to control the average TCP throughput.

6.4 TCP Rate Control using Buffer Management

In the previous section, an artificial simulation was presented where the network controlled the TCP rate by dropping a packet every time the TCP window reached a particular value. In practice, the network knows neither the size of the TCP window, nor the round trip time of the connection. In this section, we show how a switch can use per-VC accounting in its buffer to estimate the bandwidth used by the connection relative to other connections in the same buffer.

In a FIFO buffer, the average output rate of a connection is determined by the relative proportion of packets from the connection in the buffer. Let μ_i and x_i be the output rate and the buffer occupancy respectively of VC_i. Let μ and x be the total output rate and the buffer occupancy (total number of cells from all connections in the buffer) of the FIFO buffer respectively. Note that these numbers are averages over a long enough time period. Then, because the buffer is a FIFO,

$$\mu_i = \frac{x_i}{x}\mu$$

$$\text{or } \frac{x_i/x}{\mu_i/\mu} = 1$$

If the buffer occupancy of every active VC is maintained at a desired relative threshold, then the output rate of each VC can also be controlled. In other words, if a VC always has at least x_i cells in the buffer with a total occupancy of x cells, its average output rate will be at least $\mu x_i/x$.

Adaptive flows like TCP respond to segment loss by reducing their congestion window. A single packet loss is sufficient to reduce the TCP congestion window by one-half. Consider a drop policy that drops a single TCP packet from a connection every time the connection's buffer occupancy crosses a given threshold from below.

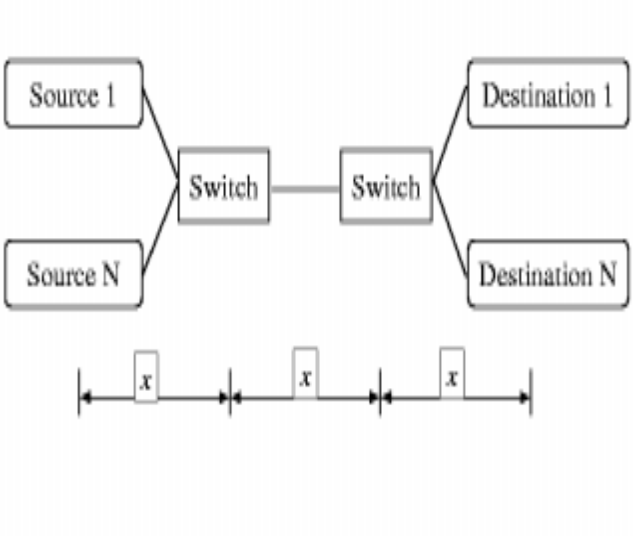


Figure 6.4: N source configuration

The drop threshold for a connection effectively determines the maximum size to which the congestion window is allowed to grow. Because of TCP's adaptive nature, the buffer occupancy reduces after about 1 RTT. The drop policy drops a single packet when the TCP's buffer occupancy crosses the threshold and then allows the buffer occupancy to grow by accepting the remainder of the TCP window. On detecting a loss, TCP reduces its congestion window by 1 segment and remains idle for about one-half RTT, during which the buffer occupancy decreases below the threshold. Then the TCP window increases linearly (and so does the buffer occupancy) and a packet is again dropped when the buffer occupancy crosses the threshold. In this way, TCP windows can be controlled quite accurately to within one round trip time. As a result, the TCP throughput can also be controlled by controlling the TCP's buffer occupancy.

Using this drop policy, we performed simulations of the TCP configuration in figure 6.4 with fifteen TCP sources divided into 5 groups of 3 each. Each TCP source was a separate UBR VC. Five different buffer thresholds (r_i) were selected and each of three TCP's in a group had the same buffer threshold. Table 6.1 lists the buffer thresholds for the VC's in the FIFO buffer of the switches. We performed experiments with 4 different sets of thresholds as shown by the threshold columns. The last row in the table shows the total buffer allocated ($r = \Sigma r_i$) to all the TCP connections for each simulation experiment. The total buffer size was large (48000 cells) so that there was enough space for the buffers to increase after the single packet drop. For a buffer size of 48000 cells, the total target buffer utilizations were 29%, 43%, 57%, 71% in the 4 columns of table 6.1, respectively. The selected buffer thresholds determine the MCR achieved by each connection. For each connection, the ratios of the thresholds to the total buffer allocation should be proportional to the ratios of the achieved per-VC throughputs to the total achieved throughput. In other words, if μ_i, μ, r_i and r represent the per-VC achieved throughputs, total throughput, per-VC buffer

Experiment #	1	2	3	4
TCP number	Threshold per TCP (cells) (r_i)			
1-3	305	458	611	764
4-6	611	917	1223	1528
7-9	917	1375	1834	2293
10-12	1223	1834	2446	3057
13-15	1528	2293	3057	3822
Tot. Thr (r)	13752	20631	27513	34392

Thresholds are allocated based on the desired throughput of each TCP group

Table 6.1: Fifteen TCP buffer thresholds

Experiment #	1	2	3	4	
TCP number	Achieved throughput per TCP (Mbps) (μ_i)				Expected Throughput
					($\mu_i^e = \mu \times r_i/\Sigma r_i$)
1-3	2.78	2.83	2.95	3.06	2.8
4-6	5.45	5.52	5.75	5.74	5.6
7-9	8.21	8.22	8.48	8.68	8.4
10-12	10.95	10.89	10.98	9.69	11.2
13-15	14.34	13.51	13.51	13.93	14.0
Tot. throughput (μ)	125.21	122.97	125.04	123.35	126.0

Achieved throughputs are close to the expected throughputs. Throughputs are proportional to the buffer allocations.

Table 6.2: Fifteen TCP throughputs

thresholds and total buffer threshold respectively, then we should have

$$\mu_i/\mu = r_i/r$$

or the expected per-VC throughput is

$$\mu_i^e = \mu \times r_i/r$$

The above formula holds when all TCPs are greedy and are trying to use up their allocated thresholds by growing their congestion window. For non-greedy sources, or sources that may be bottlenecked elsewhere in the network the thresholds must be allocated relative to the current buffer occupancy and not statically as above. This is further discussed in section 6.5.1.

Table 6.2 shows the average throughput obtained per TCP in each group for each of the four simulations. The TCP throughputs were averaged over each group to reduce the effects of randomness. The last row of the table shows the total throughput obtained in each simulation. Based on the TCP segment size (1024 bytes) and the

Experiment #	1	2	3	4
Buff. alloc.	30%	45%	60%	75%
TCP number	Ratio (μ_i/μ_i^e)			
1-3	1.00	1.03	1.02	1.08
4-6	0.98	1.01	1.03	1.04
7-9	0.98	1.00	1.00	1.02
10-12	0.98	0.99	0.98	0.88
13-15	1.02	0.98	0.97	1.01

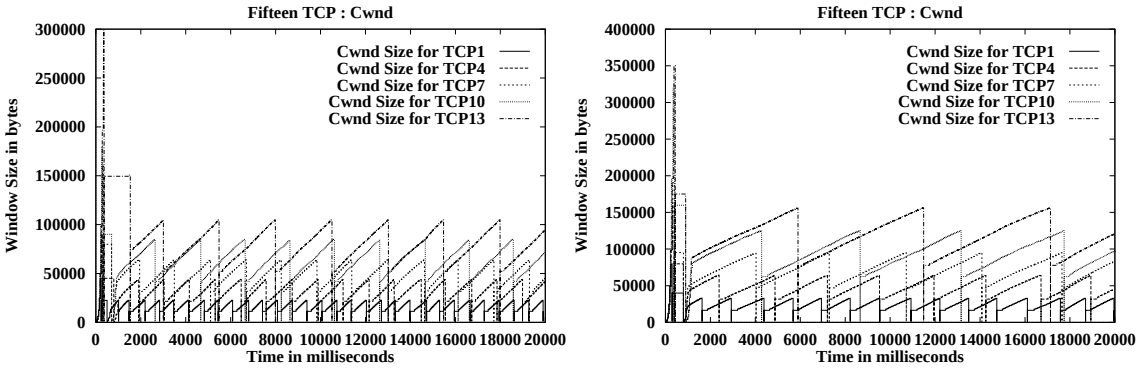
For low buffer utilization, achieved throughputs are close to the expected values. For high buffer allocation (last column), larger variation is observed due to bursty TCP dynamics.

Table 6.3: Fifteen TCP buffer:throughput ratio

ATM overhead, it is clear that the TCPs were able to use almost the entire available link capacity (approximately 126 Mbps at the TCP layer).

The proportion of the buffer usable by each TCP (r_i/r) before the single packet drop should determine the proportion of the throughput achieved by the TCP. Table 6.3 shows the ratios (μ_i/μ_i^e) for each simulation. All ratios are close to 1. This indicates that the TCP throughputs are indeed proportional to the buffer allocations. The variations (not shown in the table) from the mean TCP throughputs increased as the total buffer thresholds increased (from left to right across the table). This is because the TCPs suffered a higher packet loss due to the reduced room to grow beyond the threshold. Thus, high buffer utilization produced more variation in achieved rate (last column of Table 6.3), whereas in low utilization cases, the resulting throughputs were in proportion to the buffer allocations.

Figure 6.5 shows the congestion windows of one TCP from each group for each of the four simulations. The graphs illustrate that the behaviors of the TCP congestion



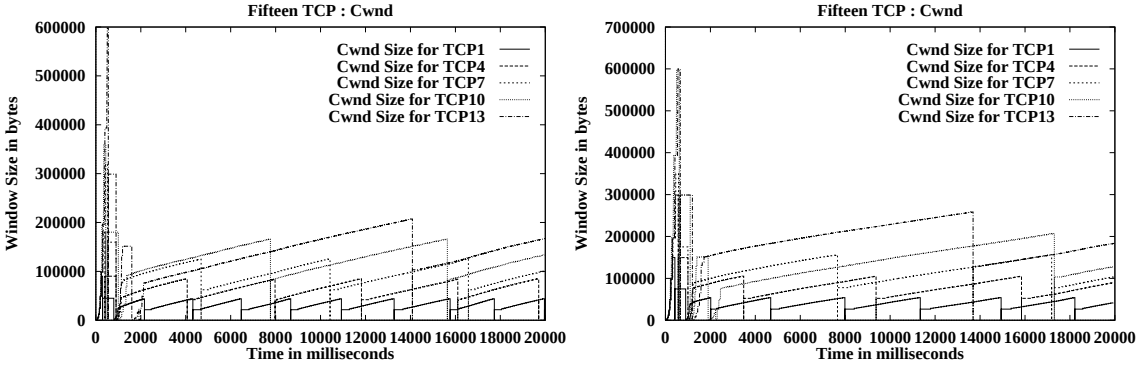


Figure 6.5: 15 TCP rate control by packet drop

windows are very regular in these cases. The average throughput achieved by each TCP can be calculated from the graphs using equations 6.1 and 6.2.

An interesting observation is that for each simulation, the slopes of the graphs during the linear increase are approximately the same for each TCP, i.e., for a given simulation, the rate of increase of CWND is the same for all TCPs regardless of their drop thresholds. We know that TCP windows increase by 1 segment every round trip time. Also, in the experiment, all TCPs have the same propagation delay and the same segment size. Thus, we can conclude that for a given simulation, TCPs sharing the FIFO buffer experience similar queuing delays regardless of the individual per-connection thresholds at which their packets are dropped. This can be explained as follows. In the scheme described above, the buffer occupancies of cells from all TCPs are maintained close to their respective thresholds (r_i). As a result, when a packet arrives at the buffer, it is queued behind cells from $\Sigma(r_i)$ packets regardless of the connection to which it belongs. Consequently, each TCP experiences the same average queuing delay.

However, as the total buffer threshold increases (from experiment (a) to (d)), the round trip time for each TCP increases because of the larger total queue size. The larger threshold also results in a larger congestion window at which a packet is dropped. A larger congestion window means that TCP can send more segments in one round trip time. However, the round trip time also increases proportionally to the increase in CWND (due to the increasing queuing delay of the 15 TCPs bottlenecked at the first switch). As a result, the average throughput achieved by a single TCP remains almost the same (see table 6.2) across the simulations.

The following list summarizes the results from the graphs:

Result 6.1 Controlling TCP throughput. TCP throughput can be controlled by controlling its congestion window, which in turn, can be controlled by setting buffer thresholds to drop packets.

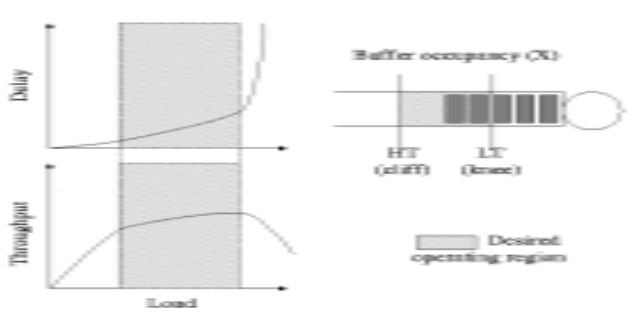
Result 6.2 FIFO buffers and throughput. With a FIFO buffer, the average throughput achieved by a connection is proportional to the fraction of the buffer occupancy of the connection's cells.

The achieved TCP throughput is independent of the absolute number of bytes from that TCP in the buffer and depends on the relative proportion of bytes it the buffer.

At a very high buffer utilization, packets may be dropped due to buffer unavailability. This results in larger variations in TCP throughputs. At very high thresholds, the queuing delay also increases significantly and may cause the TCP sources to timeout. At very low buffer thresholds (high loss rates), TCP sources become unstable and tend to timeout. Also, very low buffer occupancies result in low network utilization. Since TCP can maintain a flow of 1 CWND worth of packets each round trip time, a total buffer occupancy of 1 bandwidth-delay product should provide good utilization [65].

6.5 The Differential Fair Buffer Allocation Scheme

In this section, we describe the DFBA buffer management scheme that provides minimum rate guarantees to TCP/IP traffic. The scheme is based on the principles described above and uses per-VC accounting and thresholds to control TCP rates. The scheme stores the number of cells of each active VC in the buffer, where active VCs are those with at least one cell in the buffer. As a result, the DFBA scheme is scalable with respect to the number of VCs and only maintains fairness among the



The load versus delay-throughput graph illustrates the desired operating regions for the network. Thresholds in DFBA are selected based on the knee and cliff in the graphs.

Figure 6.6: DFBA Target Operating Region

active VCs. Another feature of DFBA is that it uses dynamic thresholds to determine the fairness of the individual VCs. Dynamic thresholds allow the scheme to maintain approximate max-min fairness among remaining VCs when other active VCs are not using their entire guaranteed rates.

6.5.1 DFBA Description

The Differential Fair Buffer Allocation (DFBA) scheme uses the current queue length (buffer occupancy) as an indicator of network load. The scheme tries to maintain an optimal load so that the network is efficiently utilized, yet not congested. Figure 6.6 illustrates the operating region for DFBA. The high threshold (H) and the low threshold (L) represent the cliff and the knee respectively of the classical load versus delay/throughput graph. The goal is to operate between the knee and the cliff.

In addition to efficient network utilization, DFBA is designed to allocate buffer capacity fairly amongst competing VCs. This allocation is proportional to the MCRs

of the respective VCs. The following variables are used by DFBA to fairly allocate buffer space:

X = Total buffer occupancy at any given time

L = Low buffer threshold

H = High buffer threshold

MCR_i = MCR guaranteed to VC_i

W_i = Weight of $VC_i = MCR_i/(\text{GFR capacity})$

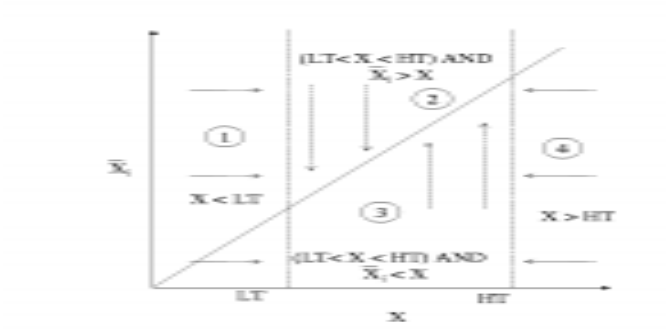
$W = \Sigma W_i$

X_i = Per-VC buffer occupancy ($X = \Sigma X_i$)

Z_i = Parameter ($0 \leq Z_i \leq 1$)

DFBA maintains the total buffer occupancy (X) between L and H. When X falls below L, the scheme attempts to bring the system to efficient utilization by accepting all incoming packets. When X rises above H, the scheme tries to control congestion by performing EPD. When X is between L and H, DFBA attempts to allocate buffer space in proportion to the MCRs, as determined by the W_i for each VC. When X is between L and H, the scheme also drops low priority (CLP=1) packets so as to ensure that sufficient buffer occupancy is available for CLP=0 packets.

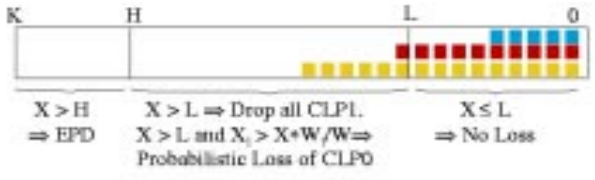
Figure 6.7 illustrates the four operating regions of DFBA. The graph shows a plot of the current buffer occupancy X versus the normalized fair buffer occupancy for VC_i. If VC_i has a weight W_i, then its target buffer occupancy (X_i) should be $X \times W_i/W$. Thus, the normalized buffer occupancy of VC_i can be defined as



X axis = total buffer occupancy (X). Y axis = normalized per-VC buffer occupancy ($X_i \times W/W_i$). In region 1 the queue is underloaded and all packets are accepted. In region 2, all low priority packets are dropped and packets from VCs whose buffer occupancy is larger than its fair share are probabilistically dropped. In region three only low priority packets are dropped. Region 4 denotes severe congestion and EPD is performed.

Figure 6.7: DFBA Drop Regions

$\bar{X} = X_i \times W/W_i$. The goal is to keep this normalized occupancy as close to X as possible, as indicated by the solid $y = x$ line in the graph. Region 1 is the underload region, in which the current buffer occupancy is less than the low threshold L. In this case, the scheme tries to improve efficiency. Region 2 is the region with mild congestion because X is above L. As a result, any incoming packets with CLP=1 are dropped. Region 2 also indicates that VC_i has a larger buffer occupancy than its fair share (since $X_i > X \times W_i/W$). In this region, the scheme drops some incoming CLP=0 packets of VC_i, as an indication to the VC that it is using more than its fair share. In region 3, there is mild congestion, but VC_i's buffer occupancy is below its fair share. As a result, only CLP=1 packets of a VC are dropped when the VC is in region 3. Finally, region 4 indicates severe congestion and EPD is performed here.



When the total buffer occupancy is below L, no packets are dropped. When it is above H, all packets are dropped. Between L and H, drop behavior is based on fairness and priority.

Figure 6.8: DFBA Buffer Occupancies for Drop

In region 2, the packets of VC_i are dropped in a probabilistic manner. This drop behavior is controlled by the drop probability function P{drop}. This is further discussed below. Figure 6.8 illustrates the drop conditions for DFBA.

Figures B.7, B.9 and B.8 in the appendix contain the complete pseudocode for DFBA.

6.5.2 DFBA Drop Probability

The probability for dropping packets from a VC when it is in region 2 can be based on several factors. Probabilistic drop is used by several schemes including RED and FRED. The purpose of probabilistic drop is to notify TCP of congestion so that TCP backs off without a timeout. An aggressive drop policy will result in a TCP timeout. Different drop probability functions have different effects on TCP behavior. In general, a simple probability function can use RED like drop, while a more complex function can depend on all the variables defined in the previous section.

For example, a sample drop probability can be defined using two main components

1. A fairness component,

2. An efficiency component.

Thus, P{drop} = fn(Fairness component, Efficiency component). The contribution of the fairness component increases as the VC's buffer occupancy X_i increases above its fair share. The contribution of the efficiency component increases as the total buffer occupancy increases above L. A sample function could increase linearly as X_i increases from $X \times W_i/W$ to X and as X increases from L to H. As a result, the drop probability is given by

$$P\{drop\} = Z_i \times (\alpha \times \frac{X_i - X \times W_i/W}{X \times (1 - W_i/W)} + (1 - \alpha)\frac{X - L}{H - L})$$

The parameter α is used to assign appropriate weights to the fairness and efficiency components of the drop probability. Z_i allows the scaling of the complete probability function based on per-VC characteristics. It has been shown that for a given TCP connection, a higher packet loss rate results in a lower average TCP window. As a result, a higher drop probability also results in a lower TCP window. Mathis et. al. [75] have shown that for random packet loss the average TCP window size is inversely proportional to the square root of the packet loss probability. As a result, the average TCP data rate D is given by

$$D \propto \frac{\text{MSS}}{\text{RTT} \times \sqrt{P\{\text{drop}\}}}$$

This relationship can have a significant impact on TCP connections with either a high data rate or a large latency or both. To maintain high TCP data rate or when the RTT is large, one must choose a large TCP MSS and/or must ensure that the average loss rate is low. As a result, DFBA can be tuned to choose a small Z_i for

large latency VCs, as in the case of switches connected to satellite hops, or for VCs with high MCRs.

6.5.3 DFBA Thresholds

The operation of DFBA is based on two static thresholds (L and H) and the per-VC dynamic thresholds $X \times W_i/W$. These thresholds determine the overall and per-VC average buffer occupancies. To maintain high throughput, the average total buffer occupancy must be close to the bandwidth-delay products of the TCP connections [65].

On a per-VC level, DFBA employs a dynamic threshold strategy as opposed to a static threshold. When all sources are infinitely greedy, static thresholds can sometimes provide equivalent guarantees. However, when the number and the data rate of sources are dynamic, static thresholds cannot provide max-min fairness among competing connections. As an example, consider a scheme that allocates a static fraction of the buffer capacity to VCs depending on their MCRs. Consider a buffer size of 100 cells, a link data rate of 100 cells per sec and three active VCs allocated 0.5, 0.25 and 0.25 of the capacity. The static thresholds in this case are 50, 25 and 25 cells respectively.

Consider two possible schemes. The first scheme drops incoming packets of a given VC as soon as the VCs buffer occupancy exceeds its static threshold. Suppose that the first two VCs are bottlenecked elsewhere in the network and only have 1 cell each in the buffer (so they are counted as active). Regardless of the traffic condition, VC3's cells are dropped probabilistically as soon as its buffer occupancy exceeds 25 cells. However, if the bandwidth-delay product of VC3 is more than 25 cells, then

the buffer will empty out before the TCPs in VC3 have a chance to replenish it with the next window of packets. As a result, the link will be underutilized.

The second scheme fixes the underutilization problem by using a low threshold L like DFBA, so that if the total buffer occupancy is less than L, then all packets are accepted. When the total buffer occupancy exceeds L, then incoming packets are checked and if their per-VC buffer occupancies exceed the static threshold, then the packets are dropped. Consider the above example again. This time suppose only VC1 is bottlenecked elsewhere, while VC2 and VC3 are vying for a fair share of the capacity. Suppose that based on the network configuration the threshold L is set to 60 cells, and both VC2 and VC3 currently have 25 cells each in the buffer while VC1 has 1 cell (negligible). Now, more cells from VC2 arrive and since the total buffer occupancy (50 cells) is less than L, these cells are accepted. When the buffer occupancy crosses L, VC2 has 35 cells and VC1 has 25 cells in the buffer. Now if cells of VC3 arrive, these cells are dropped because X > L and VC3's buffer occupancy is more than the static threshold. This will result in unfairness between VC2 and VC3 because VC2 will get more than VC3 although both were guaranteed an equal amount of the capacity.

In case of a dynamic threshold like in DFBA, VC3's cells will not be dropped because its per-VC buffer occupancy (25 cells) is less than its proportional share (30 cells) of the total buffer occupancy (60 cells). It will be able to share the unused capacity equally with VC2.

6.6 Simulation Model

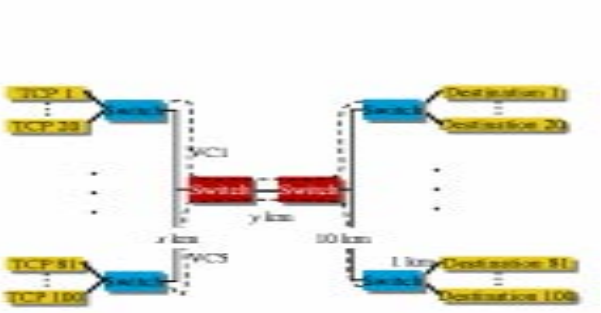
The test results presented here are with DFBA for ATM interconnected TCP/IP networks. Figure 4 illustrates the basic test configuration. The figure shows 5 pairs of local IP/ATM edge switches[4] connected to two backbone ATM switches that implement GFR. Each local switch carries traffic from multiple TCPs as shown in the figure. The backbone link carries 5 GFR VCs, one from each local network. Each VC thus carries traffic from several TCP connections. The length of the local hop is denoted by x km and the length of the backbone hop is denoted by y km. We present results with x=10 km (5 μs) and y=1000 km (5 ms). The GFR capacity was fixed to the link rate of 155.52 Mbps (approx. 353207 cells per sec). α is fixed to 0.5 in this study. All TCP sources were persistent TCPs with SACK. The SACK implementation is based on [26]. Based on previous studies, [40], we set the thresholds L and H to 0.5 and 0.9 of the buffer capacity respectively. The goal was to operate with an average buffer occupancy of about 0.5RTT as described in section 4.8.

In our simulations, we varied the following key parameters:

- **Per-VC MCR allocations**: Two sets of MCRs were chosen. In the first set, the MCR values were 12, 24, 36, 48 and 69 kcells/sec for VCs 1...5 respectively. This resulted in a total MCR allocation of about 50% of the GFR capacity. In the second set, the MCRs were 20, 40, 60, 80 and 100 kcells/sec for VCs 1...5 respectively, giving a total MCR allocation of 85% of the GFR capacity.

- **Number of TCPs**: We used 10 TCPs per VC and 20 TCPs per VC for a total of 50 and 100 TCPs respectively.

[4]Only two pairs of local switches are actually shown in the figure. Our simulations use 5 pairs.



20 TCPs are multiplexed over each local switch. GFR VCs connect local switches over backbone switches.

Figure 6.9: DFBA Simulation Configuration

- **Buffer size**: We first used a large buffer size of 25 kcells in the bottleneck backbone switch. We also analyzed DFBA performance with buffer sizes of 6 kcells and 3 kcells.

- **Heterogeneous round trip time**: Since TCP throughput is inherently dependent on the RTT, we tested the performance of DFBA for VCs with heterogeneous RTTs, by increasing the RTT of VC3 to 60 ms. All other RTTs were 10 ms.

- **Long round trip time**: We changed the backbone link to a GEO link with RTT of 550 ms. The MSS was also changed to 9180 bytes.

- Z_i: In most cases, the value of Z_i was chosen to be 1. We studied the effect of Z_i by decreasing it with increasing W_i.

MCR	Expected Throughput	Achieved Throughput	Excess Throughput	Excess/Expected
4.61	4.16	11.86	7.7	1.85
9.22	8.20	18.63	10.43	1.27
13.82	12.29	24.80	12.51	1.01
18.43	16.40	32.99	16.59	1.01
23.04	20.50	38.60	18.1	0.88
69.12	61.55	126.88	65.33	

For low MCR allocation, DFBA meets the requested guarantees. All achieved throughputs are greater than the respective expected throughputs.

Table 6.4: DFBA: 50 TCPs 5 VCs, 50% MCR Allocation

6.7 Simulation Results

Table 6.4 shows achieved throughput for a 50 TCP configuration. The total MCR allocation is 50% of the GFR capacity. The W_i values for the VCs are 0.034, 0.068, 0.102, 0.136 and 0.170. The achieved throughput column shows the total end to end TCP throughput for all the TCPs over the respective VC. The table shows that the VCs achieve the guaranteed MCR.

The table also shows that VCs with larger MCRs get a larger share of the unused capacity. The last column of the table indicates that the excess bandwidth is not shared in proportion to MCR.

The total efficiency (achieved throughput over maximum possible throughput) is close to 100%.

Result 6.3 MCR Guarantees. DFBA satisfied the MCR guarantees for 50 TCPs with heterogeneous MCRs and 50% MCR allocation.

MCR	Expected Throughput	Achieved Throughput	Excess Throughput	Excess/Expected
7.68	6.83	12.52	5.69	0.83
15.36	13.67	18.29	4.62	0.33
23.04	20.50	25.57	5.07	0.24
30.72	27.34	31.78	4.44	0.16
38.40	34.17	38.72	4.55	0.13
115.2	102.51	126.88	24.37	

DFBA meets the requested guarantees for high MCR allocations. Excess capacity is shared approximately equally.

Table 6.5: DFBA: 50 TCPs 5 VCs, 85% MCR Allocation

Result 6.4 Overall Efficiency. The unallocated link capacity is utilized efficiently. The overall system efficiency with DFBA is high, even with low MCR allocation.

MCR Allocation

Table 6.5 illustrates the performance of DFBA when 85% of the GFR capacity is allocated to the MCRs. In this case, the W_i's are 0.057, 0.113, 0.17, 0.23 and 0.28 for VC's 1...5 respectively. The table again shows that DFBA meets the MCR guarantees for VCs carrying TCP/IP traffic.

Result 6.5 MCR Allocation. DFBA provides MCR guarantees with high MCR allocation.

Number of TCPs

Table 6.6 validates the scheme for a larger number of TCPs. Each VC now carries traffic from 20 TCP connections, for a total of 100 TCPs. The total MCR allocation is 85% of the GFR capacity. All MCRs guarantees are met for a large number of TCPs and high MCR allocation.

MCR	Expected Throughput	Achieved Throughput	Excess Throughput	Excess/Expected
7.68	6.83	11.29	4.46	0.65
15.36	13.67	18.19	4.52	0.33
23.04	20.50	26.00	5.5	0.26
30.72	27.34	32.35	5.01	0.18
38.40	34.17	39.09	4.92	0.14
115.2	102.51	126.92	24.41	

DFBA meets MCR guarantees for a larger number of TCP sources.

Table 6.6: DFBA: 100 TCPs 5 VCs, 85% MCR Allocation

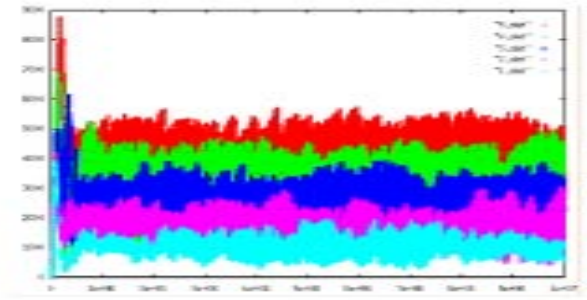


Figure 6.10: Per-VC Buffer Occupancy Levels

Result 6.6 Number of TCPs. MCR guarantees provided by DFBA are not dependent on the number of TCP connections in the network.

Figure 6.10 illustrates the buffer occupancies of the 5 VCs in the bottleneck backbone switch. The figure shows that DFBA controls the switch buffer occupancy so that VCs with a lower MCR have a lower buffer occupancy than VCs with a higher MCR. In fact, the average buffer occupancies are in proportion to the MCR values so that FIFO scheduling can ensure a long-term MCR guarantee.

MCR	Expected Throughput	Achieved Throughput	Excess Throughput	Excess/Expected
7.68	6.83	11.79	4.96	0.72
15.36	13.67	18.55	4.88	0.35
23.04	20.50	25.13	4.63	0.22
30.72	27.34	32.23	4.91	0.17
38.40	34.17	38.97	4.8	0.14
115.2	102.51	126.67	24.16	

DFBA meets MCR guarantees for small buffer sizes.

Table 6.7: DFBA: Effect of Buffer Size (6 kcells)

MCR	Expected Throughput	Achieved Throughput	Excess Throughput	Excess/Expected
7.68	6.83	10.02	3.19	0.46
15.36	13.67	19.32	5.65	0.41
23.04	20.50	25.78	5.28	0.25
30.72	27.34	32.96	5.62	0.20
38.40	34.17	38.56	4.39	0.12
115.2	102.51	126.63	24.12	

DFBA meets MCR guarantees for small buffer sizes (0.5RTT).

Table 6.8: DFBA: Effect of Buffer Size (3 kcells)

Buffer Size

Tables 6.7 and 6.8 show that DFBA provides MCR guarantees even when the bottleneck backbone switch has small buffers (6 kcells and 3 kcells respectively). The configuration uses 100 TCPs with 85% MCR allocation. Note that in most cases, the excess throughput is divided almost equally between the 5 VCs.

Result 6.7 Buffer Size. A buffer size of half round trip delay-bandwidth product is sufficient for DFBA to provide MCR guarantees.

MCR	Expected Throughput	Achieved Throughput	Excess Throughput	Excess/Expected
7.68	6.83	10.55	3.71	0.54
15.36	13.67	17.06	3.39	0.24
23.04	20.50	24.22	3.72	0.18
30.72	27.34	33.74	6.4	0.23
38.40	34.17	41.10	6.9	0.20
115.2	102.51	126.63	24.12	

DFBA meets MCR guarantees for VCs with different latencies.

Table 6.9: Heterogeneous RTT. VC3 = 60ms RTT

Heterogeneous Round Trip Time

In this configuration, the access hop (x) for VC 3, is a LEO link with a 25 ms one way delay. This results in a round trip delay of 60 ms for VC3. All other VCs still have negligible access delay and the backbone delay is also 5 ms one way. The results of this simulation with buffer size = 6000 cells is shown in table 6.9. The table again shows that DFBA provides the allocated rates to VCs with different MCRs.

Result 6.8 Heterogeneous RTT. DFBA does not have a bias against VCs with large RTTs.

Long Round Trip Time

Finally, we present the case where the backbone hop is a GEO link. The round trip delay in this case is about 550 ms. The GEO hop is the most dominant hop with respect to latency and the access hops have negligible latency. The MSS used in this simulation is 9180 bytes. Figure 6.10 shows the achieved throughputs for three different buffer sizes. Again, the table shows that DFBA provides MCR guarantees to VCs over long delay networks.

MCR (Mbps)	Expected (Mbps)	Buffer=200k Achieved (Mbps)	Buffer=200k Excess (Mbps)	Buffer=150k Achieved (Mbps)	Buffer=150k Excess (Mbps)	Buffer=100k Achieved (Mbps)	Buffer=100k Excess (Mbps)
7.68	6.91	12.4	5.49	12.8	5.89	11.40	4.49
15.36	13.82	14.96	1.14	16.17	2.35	16.99	3.17
23.04	20.74	21.86	1.12	21.63	0.83	24.56	3.82
30.72	27.65	32.10	4.45	30.25	2.60	33.72	6.07
38.40	34.56	40.21	5.65	39.84	5.28	35.52	0.96

DFBA meets MCR guarantees for long latency GFR backbones.

Table 6.10: Minimum rate guarantees with DFBA. GEO backbone

Result 6.9 Long RTT. DFBA can be used to provide MCR guarantees over long delay networks such as satellite-ATM networks based on GEO systems.

Effect of Z_i

Table 6.11 shows the effect of Z_i on the fairness of the scheme in allocating excess bandwidth. We selected 2 values of Z_i based on the weights of the VCs. In the first experiment, Z_i was selected to be $(1 - W_i/W)$ so that VCs with larger MCRs have a lower Z_i. N the second experiment, Z_i was selected to be $(1 - W_i/W)^2$. The table shows that in the second case, sharing of the excess capacity is closely related to the MCRs of the VCs.

Result 6.10 Sharing Excess Capacity. The drop probability function controls the excess capacity sharing policy in DFBA.

When Z_i's are equal, excess capacity is distributed approximately equally among the VCs.

$Z_i = 1 - W_i/W$		$Z_i = (1 - W_i/W)^2$	
Excess	Excess/Expected	Excess	Excess/Expected
4.69	0.68	1.38	0.20
4.59	0.33	4.66	0.34
4.81	0.23	5.31	0.25
5.94	0.21	5.77	0.21
4.25	0.12	7.37	0.21

Z_i can be used to control the allocation of the excess capacity

Table 6.11: DFBA: Effect of Z_i

6.8 TCP Friendly Buffer Management

The buffer management schemes discussed in this research use packet drop to signal the end systems of congestion. *Active buffer management schemes* such as RED and DFBA, provide feedback (in the form of packet loss) *before* the onset of congestion. The feedback given to the TCP source must reflect the appropriate level of congestion so that the TCP can adjust its window accordingly. If the feedback is too aggressive, i.e., if too many packets are dropped, TCP may timeout and drastically reduce the new congestion window resulting in underutilization.

The $1/sqrt(p)$ rule described in section 6.5.2 and given in [75, 67, 79, 78], shows that for random loss, TCP congestion control will adjust its throughput to be inversely proportional to the square root of the loss probability. This rule provides several useful insights in designing both network based schemes that support TCP traffic, as well as end system protocols that must coexist with TCP.

If buffer management schemes drop packets based on this rule, they can guarantee equal bandwidth sharing to TCP connections with equal RTT and segment size. RED is a scheme that attempts to do this, but suffers from the well documented dependency

of TCP throughput on RTT [31]. FRED overcomes the RTT bias by ensuring that a minimum number of packets from each flow are present in the buffer. DFBA takes this a step further by guaranteeing weighted sharing of the buffer among VCs carrying TCP connections. The key difference between RED and DFBA is that the former relies solely on drop probability for equal bandwidth sharing, while the latter relies on buffer occupancy levels for weighted bandwidth sharing. In this regard, DFBA is most similar to FRED. The calculation of load in DFBA is based on the load calculations in FBA and SD. All three schemes use a static threshold to limit the total buffer occupancy. They use current buffer occupancy as a dynamic resource to be shared among all the competing connections – either equally as in the case of FBA and SD, or unequally as in the case of DFBA.

Despite the differences, the principles behind DFBA are not incompatible with RED. In fact, RED or FRED can be used within a single VC to guarantee equal sharing of the VC throughput among the TCPs within the VC. In our research, we chose to ignore this fairness issue simply because the individual TCPs are not visible in an ATM network, and FRED would have to be deployed at the edge of the IP-ATM network to guarantee per-TCP fairness within a VC. A different instance of FRED would exist for each VC. Similarly, RED could be used within the ATM network because it does not require per-TCP flow information. Again, at each hop, a different instance of RED would exist for each GFR VC.

6.9 Chapter Summary

In this chapter, we have used FIFO buffers to control TCP rates by buffer management. An optimal set of thresholds should be selected that is high enough to

provide sufficient network utilization and is low enough to allow stable operation. The achieved TCP throughputs are in proportion to the fraction of the buffer occupied by the VC.

We have presented the Differential Fair Buffer Allocation scheme that uses per-VC accounting and FIFO queuing to provide minimum rate guarantees to VCs carrying TCP traffic. DFBA maintains buffer occupancies in proportion to the desired rates and drops packets probabilistically during congestion.

The following summarizes the conclusions from the simulations in this chapter:

Conclusion 6.1 (TCP throughput and buffer management) TCP throughput can be controlled using proportional buffer allocation on a FIFO buffer.

Conclusion 6.2 (GFR) The GFR service category provides MCR guarantees to VCs and is aware of higher layer frames.

Conclusion 6.3 (DFBA) The DFBA scheme is a per-VC accounting scheme which provides MCR guarantees to TCP/IP traffic by allocating buffers in proportion to MCR and probabilistically dropping packets during congestion.

In this chapter we have not studied the effect of network based tagging in the context of GFR. In the strict sense, GFR only provides a low CLR guarantee to the CLP=0 cell stream i.e., the cells that were not tagged by the source and passed the GCRA conformance test. However, when source based tagging is not performed, it is not clear if the CLP0 stream has any significance over the CLP1 stream. As a result, the benefits of tagging TCP packets in the network are not clear at this time. The CLP bit sensitivity in DFBA is primarily for compatibility with the ATM Forum GFR specification.

CHAPTER 7

ABR Virtual Source / Virtual Destination: Buffer Allocation in Long Delay Networks

The virtual source / virtual destination (VS/VD) option in ATM ABR networks can provide segment by segment congestion control to ABR feedback control loops. Not much work has been reported on design issues for VS/VD switches. We describe issues in designing rate allocation schemes in ATM-ABR networks for VS/VD switches. We propose a rate allocation scheme for VS/VD switches that provides efficient and fair feedback, while controlling per-VC rates. We analyze the performance of this scheme and conclude that VS/VD can help in limiting buffer requirements of switches, based on the length of their VS/VD control loops. *VS/VD is especially useful in isolating terrestrial networks from the effects of long delay satellite networks by limiting the buffer requirements of the terrestrial switches.* We present simulation results to substantiate our analysis and conclusions.

7.1 Chapter Goals

In this chapter we design a VS/VD algorithm for the ABR service and illustrate how VS/VD can be used for buffer allocation in long delay networks. We present several issues in VS/VD switch design. We describe the basic architectural components

of a VS/VD switch. We present a rate allocation scheme for feedback control in a VS/VD switch. This scheme is based on the ABR feedback control scheme ERICA+ (Explicit Rate Indication for Congestion Avoidance +) [60]. The unique feature of this scheme is that it uses the per-VC control available with VS/VD and integrates the ABR source end system policies with the feedback. As such, the scheme uses a switch model with per-VC queues which is available in the VS/VD scenario. Our simulation results show that VS/VD can help in switch buffer sizing and isolate terrestrial networks from the large buffer requirements of long delay-bandwidth product networks such as broadband satellite networks.

The presentation follows the outline below:

- We first present an overview of the VS/VD option in ABR and some potential advantages of VS/VD.
- We then briefly describe the ERICA+ algorithm for ABR congestion control. The proposed VS/VD algorithm uses ERICA+ as the underlying feedback control algorithm. However, the proposed scheme an also be used with any other feedback control algorithm like ERICA+.
- We describe an architecture for a VS/VD switch and outline the design principles for a VS/VD algorithm.
- We present our VS/VD scheme and simulation results on satellite networks.

7.2 The VS/VD Option in ABR

The Available Bit Rate (ABR) service category in ATM has been specifically developed to support data applications. Traffic is controlled intelligently in ABR using

a rate-based closed-loop end-to-end traffic management framework [34]. Resource management (RM) cells (which carry feedback from the switches) travel from the source to the destination and back. An RM cell is sent by the source every Nrm cells. The network switches monitor available capacity and provide feedback to the sources asking them to change their transmission rates. Several switch algorithms have been developed [60, 86, 1] to calculate feedback intelligently.

One of the options of the ABR framework is the Virtual Source/Virtual Destination (VS/VD) option. The VS/VD behavior specified for the ATM Available Bit Rate Service allows ATM switches to split an ABR control loop into multiple control loops. Each loop can be separately controlled by the nodes in the loop. The coupling between adjacent ABR control loops has been left unspecified by the ATM forum standards and is implementation specific. On one loop, the switch behaves as a destination end system, i.e., it receives data and turns around resource management (RM) cells (which carry rate feedback) to the source end system. On the next loop, the switch behaves as a source end system, i.e., it controls the transmission rate of every virtual circuit (VC) and schedules the sending of data and RM cells. Such a switch is called a "VS/VD switch." In effect, the end-to-end control is replaced by segment-by-segment control as shown in Figure 7.1.

VS/VD control can isolate different networks from each other. For example, two ABR networks can be isolated from a non-ATM network that separates them. Also, we show in our simulations that long latency satellite networks can be isolated from terrestrial networks so as to keep the effects of large latency to within the satellite loop.

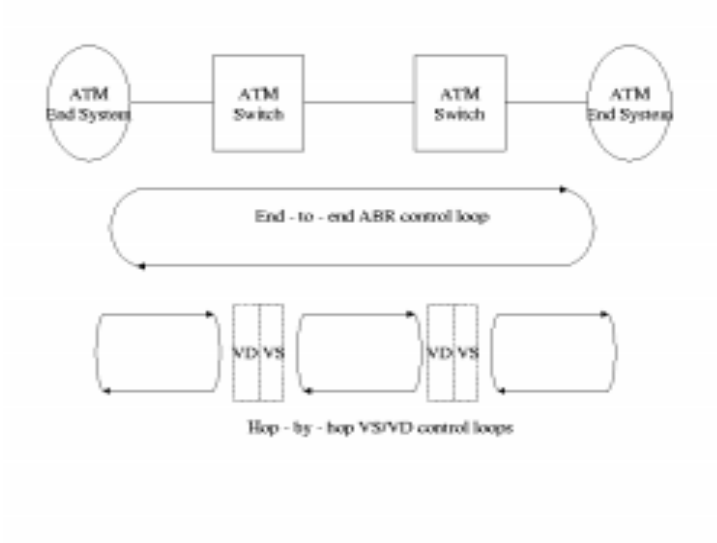


Figure 7.1: End-to-End Control vs VS/VD Control

VS/VD implementation in a switch and the coupling of adjacent control loops present several design options to switch manufacturers. A VS/VD switch is required to enforce the ABR end-system rules for each VC. As a result, the switch must be able to control the rates of its VCs at its output ports. Per-VC queuing and scheduling can be used to enforce the rate allocated to each VC. With the ability to control per-VC rates, switches at the edge of the VS/VD loops can respond to congestion notification from the adjacent loop by controlling their output rates. Switches can also use downstream congestion information, as well as their internal congestion information, to provide feedback to the upstream loop. The ability to perform per-VC queuing adds an extra dimension of control for switch traffic management schemes. Rate allocation mechanisms can utilize the per-VC control at every virtual end system (VS/VD end point) for dimensioning of resources for each VS/VD loop. Not much work has been done in examining the options for VS/VD control in ATM switches.

Kalyanaraman et. al. [62] present an exhaustive set of options available to the VS/VD switch designer for measuring load and providing switch feedback. The paper however, does not present a complete VS/VD scheme nor does it show how VS/VD can help in buffer sizing. A study of incorporating the source end system rules in VS/VD switches is presented in [22]. The work presents simulation results with EFCI based VS/VD switches over terrestrial networks and highlights the effects of varying response times in different segments of the network. The paper also presents guidelines for the design and choice of VS/VD parameters. Our work is unique because it presents an Explicit Rate scheme that uses the varying response times in different segments of the network to move the queue backlogs. Queue buildups at the edges of various VS/VD segments are proportional to the segments' bandwidth-delay products. Although the VS/VD option has been accepted as an important as useful feature of ABR because of its ability to isolate networks [34], no other work has been reported to out knowledge on the design of rate based schemes for VS/VD.

The main intent of this chapter is to present a VS/VD scheme and to show its effectiveness in buffer allocation in various segments of the network. The simulation results presented in this chapter also illustrate the convergence of the scheme in presence of temporary overload. In general, a formal proof of the convergence, fairness and efficiency properties of this scheme is very similar to that of ERICA, and is beyond the scope of this chapter. For a formal proof of convergence and max-min fairness of ERICA, the reader is referred to [60].

7.3 The ERICA Switch Scheme

In this section, we present a brief overview of the ERICA switch algorithm. Explicit Rate Indication for Congestion Avoidance (ERICA) is a switch scheme that allocates bandwidth fairly with a fast response. More details can be found in [60]. The scheme presented in this chapter is based on ERICA.

The ERICA algorithm periodically monitors the load on each link and determines a load factor (z), the ABR capacity and the number of currently active virtual connections (N) during an "averaging interval." Measurements are made on cells traveling in the forward direction, whereas the feedback is given in RM cells going in the reverse direction. At most one new feedback per source is given in any averaging interval. The variables MaxAllocPrevious, MaxAllocCurrent and FairShare are used for achieving max-min fairness while Target ABR Capacity, ABR Input Rate, z and VCShare are used for achieving efficiency (utilization and queuing delay targets). The key steps in ERICA are as follows:

Initialization:

MaxAllocPrevious $\leftarrow$ MaxAllocCurrent $\leftarrow$ FairShare

End of Averaging Interval:

$$\begin{aligned}
\text{Total ABR Capacity} &\leftarrow \text{Link Capacity} - \text{VBR Capacity} \\
\text{Target ABR Capacity} &\leftarrow \textit{Fraction} \times \text{Total ABR Capacity} \\
z &\leftarrow \frac{\text{ABR Input Rate}}{\text{Target ABR Capacity}} \\
\text{FairShare} &\leftarrow \frac{\text{Target ABR Capacity}}{\text{Number of Active VCs}} \\
\text{MaxAllocPrevious} &\leftarrow \text{MaxAllocCurrent} \\
\text{MaxAllocCurrent} &\leftarrow \text{FairShare}
\end{aligned}$$

When an FRM is received:

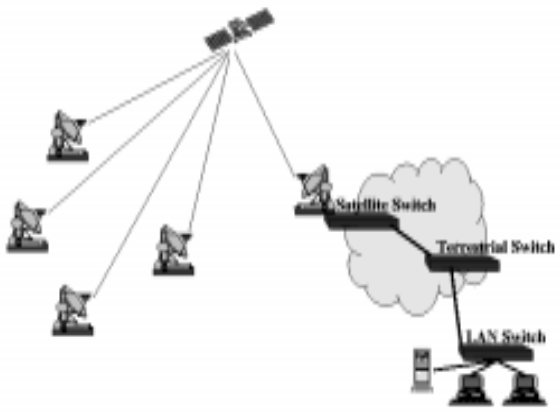
$$CCR[VC] \quad \leftarrow \quad CCR\_in\_RM\_Cell$$

When a BRM is received:

$$\begin{aligned}
\text{VCShare} \quad &\leftarrow \quad \frac{CCR[VC]}{z} \\
\text{IF } (z > 1 + \delta) & \\
\text{THEN ER} \quad &\leftarrow \quad \text{Max (FairShare, VCShare)} \\
\text{ELSE ER} \quad &\leftarrow \quad \text{Max (MaxAllocPrevious, VCShare)} \\
\text{MaxAllocCurrent} \quad &\leftarrow \quad \text{Max (MaxAllocCurrent,ER)} \\
\text{IF (ER} > \text{FairShare} \quad &\text{AND} \quad \text{CCR[VC]} < \text{FairShare)} \\
\text{THEN ER} \quad &\leftarrow \quad \text{FairShare} \\
\text{ER_in_RM_Cell} \quad &\leftarrow \quad \text{Min (ER_in_RM_Cell, ER, Target ABR Capacity)}
\end{aligned}$$

A complete pseudo-code of the scheme including all the features is provided in [60]. This simple algorithm has several desirable properties including fast response time, low queue length and simplicity. *It has been shown that with ERICA and other comparable ABR feedback control schemes, the buffer requirements of ABR switches can be bounded in most cases to a constant multiple of the bandwidth-delay product of the ABR feedback-control loop [60].*

Consider the example shown in figure 7.2. The figure illustrates a satellite ATM network that provides access to multiple remote sites via satellite. Each earth terminal on the left of the figure is connected to the central network on the right via the satellite-ATM network. ATM-ABR is used as the service to transfer data between



The link between the terrestrial and satellite switches is the bottleneck. The terrestrial switch should only buffer a small number of cells. Most of the buffering should be in the satellite switch.

Figure 7.2: A satellite-ATM network

the remote terminals and the LAN switch shown in the figure. When end-to-end ABR is used between the remote earth terminals and the LAN switch, congestion in the network can result in large queues in the terrestrial switches. These queues can be of the order of the bandwidth-delay product of the entire ABR loop which includes the satellite hop. This phenomenon is also illustrated in the simulation results in section 7.6. Unlike switches connected to large delay-bandwidth product links, terrestrial switches are not designed with large buffers. As a result, it is desirable to move the large queue backlogs caused due to congestion, to the satellite segment of the network that is expected to be able to buffer the backlog. In this chapter we will

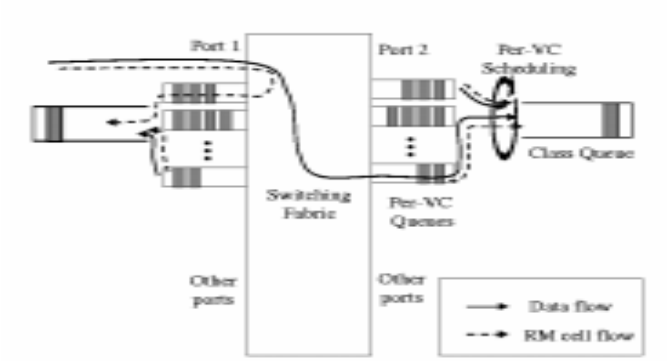
present a VS/VD scheme that will move the buffer requirements to the satellite hop, thus enabling efficient buffer size selection for switches.

7.4 A VS/VD Switch Architecture

Figure 7.3 illustrates the basic architecture of an output buffered VS/VD switch. The figure shows two output ports of the switch and the data and RM cell flow of a VC going through the switch. Data and RM cells arrive at the input side of port 1. Data cells are switched to the appropriate destination port to be forwarded to the next hop. RM cells are turned around and sent back to the previous hop. For the VC shown in the figure, port 1 acts as the VD that accepts the data cells and turns around the RM cells, while port 2 acts as the VS for the next hop. Port 1 provides feedback to the upstream node in the VC's path by inserting congestion and rate information in the appropriate RM cell fields. Port 2 sends the data to the next hop, generates an RM cell every *Nrm* cells and enforces all the source rules specified in the ABR end-system behavior. Port 1 also accepts and processes the turned around BRM cells returned by the downstream end system in the VC's path.

Each port has a single queue for the ABR service category, as well as per-VC queues for each ABR VC.[5] Each per-VC queue contains the data cells for its VC. Each per-VC queue drains into the ABR queue at the ACR allocated to the corresponding VC. RM cells are generated at the per-VC queues as specified by the ABR end system rules and are enqueued on to the ABR queue. The ABR queue is serviced by a service discipline that schedules the CBR, VBR, ABR and UBR queues according to priority

[5]The ABR queue is not essential if per-VC queuing and scheduling are used, but we include it to illustrate a general architecture. The ABR queue can be removed without affecting the scheme presented in this chapter.



Data cells are forwarded to the output port through the fabric. Forward RM cells are turned around. Backward RM cells are destroyed after their value is noted. Forward RM cells are generated by the VS after every Nrm-1 cells.

Figure 7.3: VS/VD switch architecture

and bandwidth guarantees. Details of the scheduling policy are beyond the scope of this research.

In the remainder of this section, ERICA and ERICA+ are used as a basis for presenting the switch models for feedback control. However, the discussion is general and applies to any rate allocation scheme that uses the target utilization and queue length parameters in its rate calculations. The discussion presents some fundamental concepts that should be used in the design of rate allocation algorithms for VS/VD switches.

7.4.1 A Non-VS/VD Switch Model

Figure 7.4 shows a queuing model for a single port of an **output buffered non-VSVD switch** (port i). The port has a single queue for all ABR VCs. Cells from all the ABR VCs destined for the output port are enqueued in the ABR queue in a FIFO manner. We define the following:

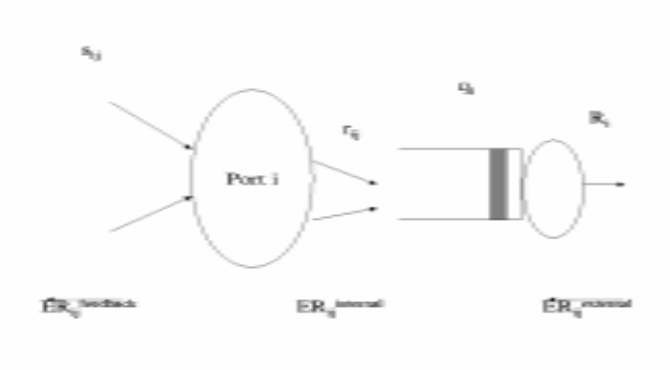


Figure 7.4: Queuing model for non-VS/VD switch

s_{ij}: Input rate of VC_j on port i.

r_{ij}: Input rate into the ABR queue of port i.

R_i: Service rate of the ABR queue at port i.

q_i: Queue length of the ABR queue at port i.

F_i: Target ABR utilization.

$g(q_i)$: Queue control function.

$ER_{ij}^{internal}$: ER calculated at port i for VC j based on internal congestion.

$ER_{ij}^{external}$: ER received on port i for VC j from the downstream node.

$ER_{ij}^{feedback}$: ER calculated for feedback to the upstream hop.

In this case, since the node simply switches cells from the input to the output port, we have $s_{ij} = r_{ij}$. Let R_i be the service rate of the ABR queue at port i. Then R_i corresponds to the total bit rate of the link available to ABR. Let q_i be the queue length of the ABR queue. Let N be the number of ABR VCs sharing the link.

Many rate allocation algorithms use a parameter $F_i, (0 < F_i \leq 1)$ which is the target utilization of the link. The link rate is allocated among the VCs so that

$$\sum_{j=1}^{j=N} r_{ij} \leq F_i R_i$$

i.e., the goal of the switch is to bring the total input rate into the ABR queue to the desired value of $F_i R_i$. Let $ER_{ij}^{internal}$ be the ER calculated at port i based on the internal congestion in the port. This is the rate at which the switch desires VC_j to operate. Port i also receives rate allocation information from the downstream node. This is shown in the figure as $ER_{ij}^{external}$. Then, the switch also provides feedback from port i to the upstream node, as

$$ER_{ij}^{feedback} \leftarrow Min(ER_{ij}^{internal}, ER_{ij}^{external})$$

At the upstream node, the feedback is received (say on port k) as $ER_{kj}^{external}$ and the ABR algorithm performs its rate calculations for VC_j in a similar fashion.

The internal explicit rate calculation is based on the local switch state only. A typical scheme like ERICA [60], uses several factors to calculate the explicit rate. In particular, the ERICA algorithm uses the total input rate to the ABR queue, the target utilization of the link and the number of VCs sharing the link to calculate the desired operating point of each VC in the in the next feedback cycle, i.e.,

$$ER_{ij}^{internal} \leftarrow f(\sum_{j} r_{ij}, F_i R_i, N)$$

In steady state, the ERICA algorithm maintains $\sum_j r_{ij} = F_i R_i$, so that any queue accumulation due to transient overloads can be drained at the rate $(1 - F_i)R_i$. As a result, the ERICA algorithm only allocates a total of $F_i R_i$ to the VCs sharing the link and results in $100F_i\%$ steady state link utilization of the outgoing link.

The ERICA+ algorithm can achieve 100% steady state link utilization by additionally considering the queue length of the ABR queue when it calculates the internal rate for VC_j, i.e., for ERICA+,

$$ER_{ij}^{internal} \leftarrow f(\sum_j r_{ij}, g(q_i)R_i, N)$$

where

$$g(q_i), (0 < g_{min} \leq g(q_i) \leq g_{max})$$

is a function known as the *queue control function*, that scales the total allocated capacity R_i based on the current queue length of the ABR queue. If q_i is large, then $g(q_i) < 1$ so that $\sum_j r_{ij} = g(q_i)R_i$ is the target operating point for the link, and $(1 - g(q_i))R_i$ can be used to drain the queue to a desired value (q_i^{target}). The queue control function is bounded below by $g_{min} > 0$ so that at least some minimal capacity is allocated to the VCs. A typical value for the ERICA+ algorithm of g_{min} is 0.5. When the queue is small, ($q_i < q_i^{target}$), $g(q_i)$ may increase to slightly more than 1 so that sources are encouraged to send at a high rate. As a result, switches try to maintain a pocket of queues of size q_i^{target} at all times so as to ensure 100% link utilization.

7.4.2 A VS/VD Switch Model

Figure 7.5 shows a model for a port of an output queued VS/VD switch. The port consists of per-VC queues that drain into a single ABR queue. The input rates to the per-VC queues are not the same as the output rates from the per-VC queues. Each queue has a separate ABR server that performs the functions of a ABR source end system as well as integrates the end system rules with the switch algorithm (ERICA+). Thus, the servers at the per-VC queues also control the output rates

of their respective queues based on their rate allocations and the ABR source end system rules.

we define the following:

s_{ij}: Input rate of VC_j onto its per-VC queue.

r_{ij}: Output rate from the per-VC queue of VC i into the ABR queue of port i.

q_{ij}: Queue length of VC j's per-VC queue at port i.

F_{ij}: Target per-VC utilization of r_{ij}.

R_i: Service rate of the ABR queue at port i.

q_i: Queue length of the ABR queue at port i.

F_i: Target ABR utilization.

$g(.)$: Queue control function.

$ER_{ij}^{internal}$: ER calculated at port i for VC j based on internal congestion.

$ER_{ij}^{external}$: ER received on port i for VC j from the downstream node.

$ER_{ij}^{feedback}$: ER calculated for feedback to the upstream hop.

Let r_{ij} be the rate allocated to VC_i's per-VC queue. Thus, r_{ij} is the Allowed Cell Rate (ACR) for VC j. In the case of ERICA, the sum total of the input rates to the ABR queue should be limited by F_iR_i. This allows the ABR queue to drain in case of transient overloads from the per-VC queues. The input to the per-VC queues (s_{ij}) should be now limited by $F_{ij}r_{ij}$, allowing the per-VC queues to also use a rate of $(1-F_{ij})r_{ij}$ to recover from transient overloads. Note that $F_{ij}r_{ij}$ is an upper bound to

the rate allocation. In practice the observed VC's drain rate may be further restricted by the ABR source end system rules. Moreover, for an ERICA+ like scheme that uses queue length information to calculate available capacity, additional per-VC queue length information (q_{ij}) is used to control s_{ij} in relation to r_{ij}. Thus, for ERICA+, the desired operating point is decided such that,

$$\sum_j r_{ij} \leq g(q_i)R_i$$

and

$$s_{ij} \leq g(q_{ij})r_{ij}$$

The feedback given to the previous loop is set to the desired per-VC operating point, which is the desired input rate to the per-VC queues. As a result, the per-VC feedback is controlled by the VC's queue length. This can be used to isolate the VCs on the same link from one another. Thus, if VC_j experiences a transient overload, only $ER_{ij}^{feedback}$ is reduced and the feedbacks to the remaining VCs are not necessarily affected by this temporary overload.

The following section presents the complete pseudocode of the rate allocation scheme for VS/VD switches based on the above principles. This scheme is a variation of the ERICA+ algorithm and converges to max-min fairness while achieving high link utilization. The scheme also limits the maximum buffer sizes in a switch to a function of the delay-bandwidth product of its upstream VS/VD control loop. Section 7.6 presents the simulation results of this scheme and illustrates how VS/VD can be used in switches to limit buffer sizes over non-VS/VD switches.

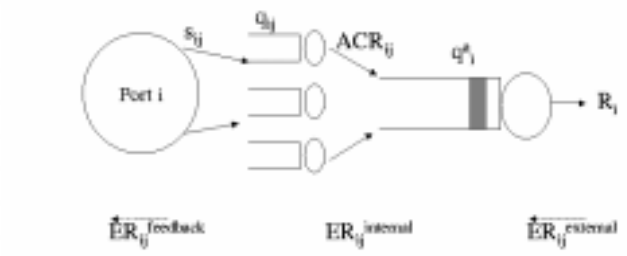


Figure 7.5: Queuing model for per-VC VS/VD switch

7.5 A Per-VC Rate Allocation Algorithm for VS/VD

The scheme presented in this section is based on the ERICA+ scheme for ABR feedback [60]. The basic switch model is shown in figure 7.5. The switch maintains an averaging interval at the end of which it calculates the rate allocations ($ER_{ij}^{internal}$) for each VC to provide feedback to the previous hop. Feedback is calculated for each VC based on the following factors:

- The actual (measured) scheduled rate of the VC queue into the ABR queue or the link.

  The allocated rate (ACR) of the VC queue into the ABR queue or the link (r_{ij}).

- The queue length of the ABR queue (q_i).

- The output rate of the ABR queue (R_i). This is also the total estimated ABR capacity of the link.

- The number of active ABR VC's sharing the ABR queue (N). N is measured during the switch interval as described in [60].

- The external rate allocation received by each VC from the downstream hop ($ER_{ij}^{external}$).

- The queue control function $g(.)$. Various queue control functions are proposed in [98].

The basic design is based on the following principle. A desired input rate is calculated for each queue in the switch and this desired rate is given as feedback to "the previous server" in the network. In the case of the ABR queue, the previous server controls the per-VC queues of the same node. The previous server for the per-VC queue is the ABR queue of the upstream hop in the VS/VD loop.

A portion of the link capacity $g(q_i)R_i$ is allocated in a max-min fair manner among the per-VC queues[6]. The remaining portion is used to drain the ABR queue formed due to transient overloads. Then, the per-VC feedback is calculated for the upstream hop based on the per-VC queue length (q_{ij}) and the allocated rate of the per-VC queue (r_{ij}). This calculation allocates a fraction (that depends on the queue length) of r_{ij} to the previous hop as $ER_{ij}^{feedback}$ so that s_{ij} in the next cycle is less than r_{ij} thus allowing the per-VC queue to drain out any transient overloads.

The algorithm supports the complete ATM Forum specification of the end-system rules and provides explicit rate feedback via the RM cells. The complete algorithm is presented in figures 7.7 through 7.10. The figures describe the actions taken by the switch when certain events are triggered in the switch. These events are caused by

[6]Optionally, the sum of the per-VC queues and the ABR queue can be used. In our simulations, this resulted in lower oscillations. In the absence of a ABR queue, the function $g(q_i)R_i = F_iR_i$ where $F_i \leq 1$ is the target utilization of the link

arrival of cells or by timeouts occurring in the switch. The algorithm is executed on every output port in the switch. In the algorithm presented, values read from a cell are preceded by "cell" and the port number and VC number are subscripted by i and j respectively. The following events can be triggered in the switch:

- *Receipt of a data cell* (figure 7.6): When a data cell arrives on a link, it is received by the input port which acts as the VD for the VC. The VD is also the VS for the reverse direction VC and it must provide feedback to the node in the reverse direction. As a result, the VD makes a note of the EFCI state of the cell and forwards the cell to the switching fabric. After the cell is switched, it is queued onto the outgoing port (the VS side). The VS checks if the VC queue was empty before the cell arrived. If the cell arrives into an empty queue, then the VS schedules an event to send output the cell to the link if such an event has not been scheduled already.

- *Receipt of a Forward RM cell* (figure 7.7): When a Forward RM (FRM) cell arrives on VC j of port i, the VD must turn it around so that VS_{ij} can send a Backward RM (BRM) cell. This BRM cell must provide feedback to the previous hop that sent the FRM. The VD first uses the feedback value $ER_{ij}^{feedback}$ to update the ER field in the cell. It then follows the ABR destination end system rules [34] to turn around the FRM cell and schedules a send event at the VS if needed.

- *Receipt of a Backward RM cell* (figure 7.8): A Backward RM (BRM) cell is received by the VS on port i for VC j. The BRM cell contains feedback from the downstream ABR node (switch/ destination end system or virtual destination).

The VS updates the ACR for the outgoing VC based on the feedback from the BRM cell, as well as the internal congestion of the switch. The VS first updates the ACR based on the CI and NI bits in the cell. The ER field in the cell then is used for updating ACR. The VS also uses the ER allocated by ERICA+ in the CalculateER() function to update the ACR. The CalculateER() function uses the same calculations as standard ERICA+ described in [60]. The ACR is compared to the MCR to make sure that at least a rate of MCR is allocated to the VC. The VS then reschedules the sending time of the next cell based on the updated ACR if necessary. This behavior is consistent with the ABR source end system rules specified in [34].

- *Expiration of the switch averaging interval* (figure 7.9): The switch maintains an averaging interval (similar to the ERICA+) algorithm. During the interval, the switch measures the CBR and VBR usage, the ABR input rate and the number of active sources. At the end of each interval, the switch computes the overload factor and the fair share as in ERICA+ using the ABR queue or the sum of the ABR queue and the per-VC queues. The switch also updates the MaxAllocPrevious and MaxAllocCurrent variables as in ERICA+. In addition, the switch also uses the per-VC queue lengths to compute $ER_{ij}^{feedback}$ for each VC. The other computations performed at this time are the same as in ERICA+ [60].

- *Time to send a cell according the to the source end system policies* (figure 7.10): This event is triggered only for a VS/VD switch because it models the end-system behavior of the switch. The actions taken by the switch are exactly the

same as those taken by an ABR source end system (SES). The VS follows the SES rules and sends either FRM, BRM or data cells from the per-VC queue to the ABR queue.

```
Case 1: Data cell received from link
        CI_VC_ij  ← EFCI state of cell
        Send cell to switching fabric
Case 2: Data cell received from switch fabric
        if (q_ij = 0 ∧ time_to_send not scheduled) then
                Schedule: time_to_send_ij  ← now()
```

Figure 7.6: Data Cell Received

```
cell.ER  ← Min(cell.ER, ER_ij^feedback)
if (turnaround_ij) then
        CI_TA_ij ← CI_TA_ij ∨ CI_VC_ij
        Send BRM cell
        CI_VC_ij  ← 0
CCR_TA_ij  ← cell.CCR
MCR_TA_ij  ← cell.MCR
ER_TA_ij  ← cell.ER
CI_TA_ij  ← cell.CI
NI_TA_ij  ← cell.NI
turnaround_ij  ← TRUE
if (time to send not scheduled) then
        Schedule Event: time_to_send_ij  ← now()
Discard FRM cell
```

Figure 7.7: FRM cell received

```
if (cell.CI=0) then
        ACR_ij ← ACR_ij − ACR_ij × RDF_ij
else if (cell.NI=1) then
        ACR_ij = ACR_ij + RIF_ij × PCR_ij
        ACR_ij = Min(ACR_ij, PCR_ij)
ACR_ij ← min(cell.ER, ACR_ij)
ACR_ij ← Min(ACR_ij, CalculateER(i,j))
ACR_ij ← Max(ACR_ij, MCR_ij)
CCR_ij ← ACR_ij
if (cell.BN = 0) then
        if (time_to_send_ij > now() + 1/(ACR_ij)) then
                Reschedule: time_to_send_ij ← now() + 1/ACR_ij
Discard the BRM cell
```

Figure 7.8: BRM cell received

7.6 Simulation Model

In this section we present the simulation model to highlight the features of the VS/VD rate allocation scheme described in this chapter, and its potential advantages over non-VS/VD switches. In particular, we are interested in comparing the buffer requirements of a VS/VD switch with those of a non-VS/VD switch.

Figure 7.11 shows the basic configuration used in the simulations. The configuration consists of three switches separated by 1000 km links. The one way delay between the switches is 5 ms. Five sources send data as shown in the figure. The first hop from the sources to switch 1 is a long delay satellite hop. We simulated two values of one way delay – 275 ms (GEO satellite delay) and 50 ms (LEO satellite delay). The link capacity of link 2 is 45 Mbps, while all other links are 155 Mbps links. Our simulations use infinite ABR sources. ABR initial cell rates are set to

TotalABRCapacity $\leftarrow$ LinkCapacity - VBRCapacity
TargetABRCapacity $\leftarrow$ g(q_i)$\times$TotalABRCapacity
(Optional) TargetABRCapacity $\leftarrow$ g($\sum_j q_{ij} + q_i$)$\times$TotalABRCapacity
InputRate $\leftarrow$ Measured ABRInputRate
OverLoadFctr $\leftarrow$ InputRate/TargetABRCapacity

For each VC j on port i
    $ER_{ij}^{feedback} \leftarrow g(q_{ij}) \times ACR_{ij}$
    $FirstFRMinInterval_{ij} \leftarrow$ TRUE
    $PrevIntvlMaxER_{ij} \leftarrow CurrIntvlMaxER_{ij}$
    $CurrIntvlMaxER_{ij} \leftarrow 0$

Figure 7.9: End of averaging interval

30 Mbps in all experiments. Thus, only link 2 is the bottleneck link for the entire connection.

7.7 Simulation Results

Figure 7.12 illustrates the difference in the maximum buffer requirements for a VS/VD switch and a non-VS/VD switch with the GEO satellite delay configuration. Switch 2 is the bottleneck switch since link 2 has a capacity of 45 Mbps. Switch 1 is connected to the satellite hop and is expected to have large buffers. Switch 2 is a terrestrial switch whose buffer requirements should be proportional to the bandwidth-delay product of terrestrial links. Without VS/VD, all queues are in the bottleneck switch (switch 2). The delay-bandwidth product from the bottleneck switch to the end system is about 150,000 cells (155 Mbps for 550 ms). This is the maximum number of cells that can be sent to switch 2 before the effect of its feedback is seen by the switch. Figure 7.12(d) shows that without VS/VD, the maximum queue length in

switch 2 is proportional to the feedback delay-bandwidth product of the control loop between the ABR source and the bottleneck switch. However, a terrestrial switch is not expected to have such large buffers and should be isolated from the satellite network. In the VS/VD case, (figure 7.12 (a) and (b)), the queue is contained in switch 1 and not switch 2. The queue in switch 2 is limited to the feedback delay-bandwidth product of the control loop between switch 1 and switch 2. The observed queue is always below the maximum expected queue size of about 3000 cells (155 Mbps for 10 ms).

Figure 7.13 shows the corresponding result for the LEO satellite configuration. Again, with the VS/VD option, queue accumulation during the open loop period is moved from switch 2 to switch 1. The maximum queue buildup in switch 1 during the open loop phase is about 35000 (155 Mbps for 120 ms). The results also show that the corresponding link utilizations for link 1 and link 2 are comparable for VS/VD and non-VS/VD. The ACRs allocated allocated to each source are the same and the resulting scheme is fair in the steady state. Detailed results of the fairness and efficiency can be obtained from [41].

Figures C.1 and C.2 illustrate the corresponding link utilizations for link 1 and link 2 for the GEO and LEO configurations respectively. The figures show that the link utilizations are comparable for VS/VD and non-VS/VD. Figures C.3 and C.4 show the ACRs allocated to each source for the GEO and LEO cases respectively. The ACR graphs show that the resulting scheme is fair in the steady state. The transient differences in the ACRs due to the small transient differences in the per-VC queue length.

We have also conducted experiments with two configurations with persistent TCP sources. The TCP sources result in bursty traffic to the switch because of the window based TCP flow control. Figure C.8 shows the results of the above. The figures show that even with bursty sources, the maximum buffer size requirements for VS/VD switch is still proportional to the feedback delay-bandwidth product of the upstream VS/VD loop.

The following conclusion can be drawn for ABR networks implementing ERICA+ and the VS/VD algorithm presented here:

Result 7.1 Non-VS/VD Switches: Buffer Requirements. ABR switches must have buffers proportional to the round trip delay-bandwidth product of the ABR feedback control loop to which they belong.

Result 7.2 VS/VD Switches: Buffer Requirements. ABR switches at the edges of two VS/VD control segments must have buffers proportional to the round trip delay-bandwidth product of the upstream VS/VD segment to which they are connected.

This demonstrates that VS/VD can be helpful in limiting buffer requirements in various segments of a connection and can isolate network segments from one another.

7.8 Chapter Summary

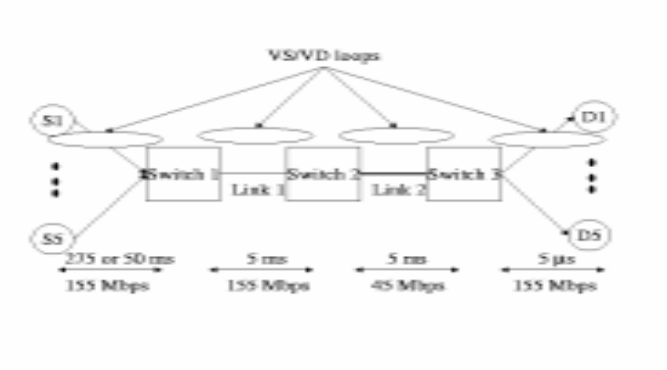
In this chapter, we have presented a per-VC rate allocation mechanism for VS/VD switches based on ERICA+. This scheme retains the basic properties of ERICA+ (max-min fairness, high link utilization and controlled queues), and isolates VS/VD control loops thus limiting the buffer requirements in each loop. We have shown

that VS/VD helps in reducing the buffer requirements of terrestrial switches that are connected to satellite gateways. Without VS/VD, terrestrial switches that are a bottleneck, must buffer cells of upto the feedback delay-bandwidth product of the entire control loop (including the satellite hop).

Conclusion 7.1 (Effect of VS/VD) With a VS/VD loop between the satellite and the terrestrial switch, the queue accumulation due to the satellite feedback delay is confined to the satellite switch. The terrestrial switch only buffers cells that are accumulated due to the feedback delay of the terrestrial link to the satellite switch.

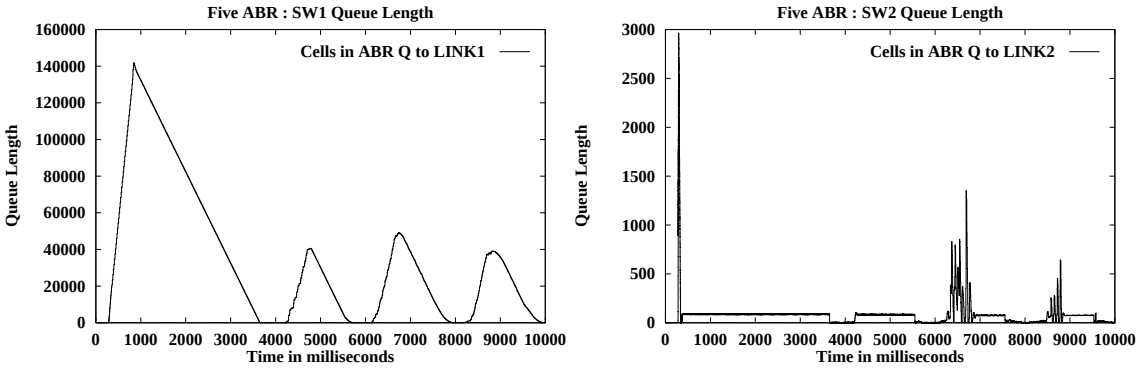
```
time_to_send_ij  ← 0
if (q_ij > 0 ∨ turn_around_ij) then
        if (ACR_ij < TCR_ij) then
                Send out of rate FRM
                Schedule: time_to_send_ij = now() + 1/TCR_ij
        else
                if ((count_ij ≥ Nrm) ∨ ((count_ij > Mrm) ∧ (now() ≥ last_RM_ij + Trm)))
then
                        time  ← now() - last_RM_ij
                        if (time > ADTF ∧ACR_ij > ICR_ij) then
                                ACR_in ← ICR_ij
                        if (unack_ij ≥ CRM_ij) then
                                ACR_ij ← ACR_ij − ACR_ij × CDF_ij
                                ACR_ij ← Max(ACR_ij, MCR_ij)
                        Send in rate FRM()
                        count_ij  ← 0
                        last_RM_ij  ← now()
                        first_turn_ij  ← TRUE
                        unack_ij = unack_ij + 1
                else if (turn_around_ij ∧ (first_turn_ij ∨ q_ij > 0)) then
                        CI_TA_ij ← CI_TA_ij ∨ CI_VC_ij
                        send in rate BRM
                        CI_VC_ij ← 0
                        turn_around_ij ←  FALSE
                        first_turn_ij ←  FALSE
                else
                        Send data cell

        count_ij ← count_ij + 1
        Schedule: time_to_send_ij ←  now() + 1/ACR_ij
```

Figure 7.10: Time to send expires (now() ≥ time_to_send$_{ij}$)



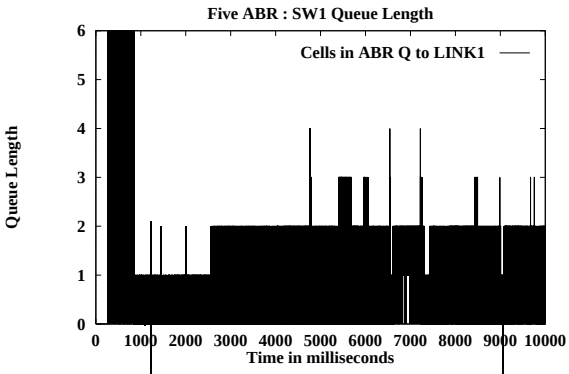
Link 2 is the bottleneck link.

Figure 7.11: Five sources satellite configuration

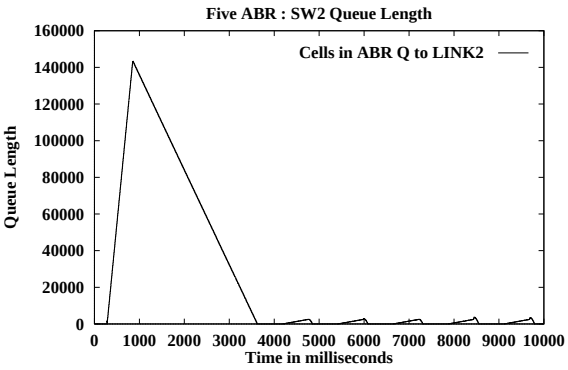


(a) VS/VD: Switch 1 Queue

(b) VS/VD: Switch 2 Queue

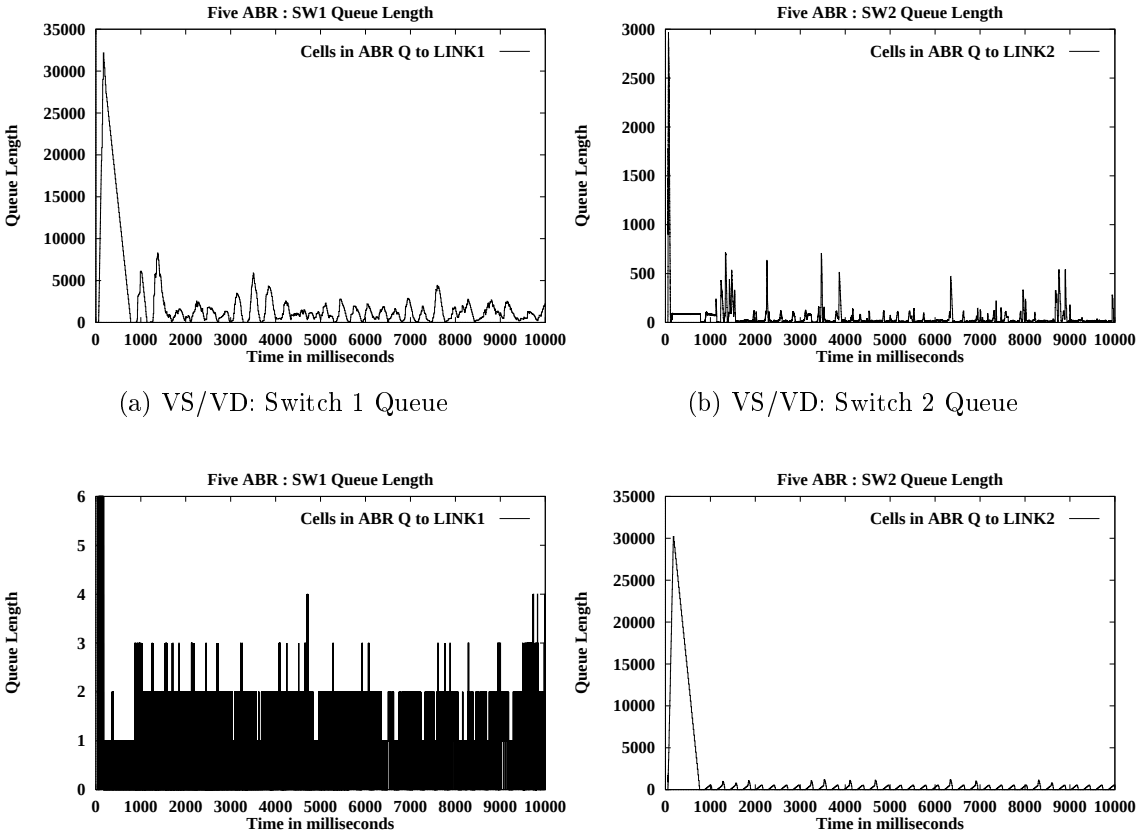


(c) Non-VS/VD: Switch 1 Queue



(d) Non-VS/VD: Switch 2 Queue

Figure 7.12: Switch Queue Length for VS/VD and non-VS/VD:GEO



(a) VS/VD: Switch 1 Queue

(b) VS/VD: Switch 2 Queue

(c) Non-VS/VD: Switch 1 Queue

(d) Non-VS/VD: Switch 2 Queue

Figure 7.13: Switch Queue Length for VS/VD and non-VS/VD Case: LEO

CHAPTER 8

Summary and Future Work

The Internet is playing an ever increasing role in the social fabric of our lives. It is competing with not only the telephony infrastructure, but also with traditional print media and entertainment. Corporations are moving towards intranets and extranets for communication and information retrieval. If the Internet is to blossom into a ubiquitous and transparent medium for communication and information dispersal, it must provide services that can support reliable and timely exchange of information. It is our belief that robust traffic management is the foundation on which these services should be built.

In this research, we have studied possible implementations of two fundamental types of services in the future Internet. The first type consists of the best effort services, such as the Unspecified Bit Rate service in ATM networks. We have shown that naive design of UBR can degrade the service level by delivering poor TCP throughput and low fairness. We have explored the use of buffer management as a simple technique to improve the efficiency and fairness. Intelligent buffer management combined with robust TCP implementations can provide high quality best effort service. In this context, we have proposed the Selective Drop scheme and analyzed Early Packet

Discard, Fair Buffer Allocation and Selective Drop. We have also studied the advantages of providing a minimum rate guarantee to the best effort service queue as a means of preventing starvation of best effort traffic by higher priority traffic.

The second type of service is the best effort service with minimum rate guarantees to its flows or groups of flows. The Guaranteed Frame Rate and the Available Bit Rate in ATM are examples of this service type. Both GFR and ABR provide Minimum Cell Rate guarantees to each VC. ABR has the added feature of rate based feedback control that can effectively limit the amount of congestion in the network. We have designed a buffer management scheme called Differential Fair Buffer Allocation for the GFR service. The scheme is frame aware and CLP aware and provides MCR guarantees to VCs with aggregated TCP flows. We have designed a feedback control scheme for the virtual source / virtual destination option in ABR to control network queues in proportion to the delay-bandwidth products of the segments.

In this chapter, we provide some further thoughts on the applications, limitations and future potential of this research. We first briefly compare and contrast the UBR, GFR and ABR service categories. We then summarize our results and discuss some limitations and ideas for future work.

8.1 Comparison of ATM Service Categories

Existing and proposed ATM standards provide several options for TCP/IP data transport over a satellite-ATM network. The three service categories – ABR, UBR and GFR – and their various implementation options present a cost-performance tradeoff for TCP/IP over ATM. A comparison of the service categories can be based on the following factors:

- Implementation Complexity
- Buffering requirements for switches and ATM end-systems
- Network bandwidth utilization
- Bandwidth allocation (fairness and MCR guarantees)

Higher complexity arises from resource allocation algorithms for Connection Admission Control (CAC) and Usage Parameter Control (UPC), as well as from sophisticated queuing and feedback control mechanisms. While UPC is performed at the entrance of the ATM network to control the rate of packets entering the network, CAC is performed during connection establishment by each network element.

UBR is the least complex service category because it does not require any CAC or UPC. Typical UBR switches are expected to have a single queue for all UBR VCs. Buffer management in switches can vary from a simple tail drop to the more complex per-VC accounting based algorithms such as FBA or SD. An MCR guarantee to the UBR service would require a scheduling algorithm that prevents the starvation of the UBR queue.

The ABR service can be implemented with a single ABR queue in the switch. The VS/VD option requires the use of per-VC queuing and increases the implementation complexity of ABR.

The GFR service can be implemented by either a single queue using a DFBA like mechanism, or per-VC queues and scheduling. The CAC requirements for GFR and ABR are similar. However, the tagging option, CLP conformance and MFS conformance tests in GFR add complexity to the UPC function.

The additional complexity for ABR feedback control presents a tradeoff with ABR buffer requirements. Network buffering is lower for ABR than for UBR or GFR. In addition, ABR has controlled buffer requirements that depend on the bandwidth-delay product of the ABR feedback loop. At the edge of the ATM network, network feedback can provide information for buffer dimensioning. Large buffers in edge routers can be used when the ABR network is temporarily congested. In the case of UBR and GFR, edge devices do not have network congestion information and simply send the data into the ATM network as fast as they can. As a result, extra buffers at the edge of the network do not help for UBR or GFR. This is an important consideration for large delay bandwidth satellite networks. With ABR, satellite gateways (routers at the edges of a satellite-ATM network) can buffer large amounts of data, while the buffer requirements of the on-board ATM switches can be minimized. The buffer requirements with UBR/GFR are reversed for the gateways and on-board switches.

The ABR service can make effective use of available network capacity by providing feedback to the sources. Edge devices with buffered data can fill up the bandwidth within one feedback cycle of the bandwidth becoming available. This feedback cycle is large for satellite networks. With UBR and GFR, available bandwidth can be immediately filled up by edge devices that buffer data. However, the edge devices have no control on the sending rate and data is likely to be dropped during congestion. This data must be retransmitted by TCP and this can result in inefficient use of the satellite capacity.

In addition to efficient network utilization, an ATM network must also fairly allocate network bandwidth to the competing VCs. While vanilla UBR has no mechanism for fair bandwidth allocation, UBR or GFR with buffer management can provide

per-VC fairness. ABR provides fairness by per-VC rate allocation. A typical ATM network will carry multiple TCP connections over a single VC. In ABR, most losses are in the routers at the edges of the network and these routers can perform fair buffer management to ensure IP level fairness. On the other hand, in UBR and GFR, most losses due to congestion are in the satellite-ATM network, where there is no knowledge of the individual IP flows. In this case, fairness can only be provided at the VC level.

8.2 Summary of Results

Several issues arise in optimizing the performance of TCP over ATM networks. This research emphasizes that both TCP mechanisms as well as ATM mechanisms should be used to improve TCP performance and provide rate based guarantees over ATM networks.

ATM technology provides at least 3 service categories for data: UBR, ABR and GFR. Each of these categories can be implemented by a number of mechanisms. In this work, the following implementations are considered:

- Unspecified Bit Rate (UBR) with tail drop,
- UBR with frame based discard (EPD),
- UBR with intelligent buffer management,
- UBR with guaranteed rate,
- Guaranteed Frame Rate (GFR) with buffer management,
- ABR with rate based feedback,

- ABR with virtual source/virtual destination (VS/VD).

In addition, TCP provides several congestion control mechanisms including:

- Vanilla TCP with slow start and congestion avoidance,
- TCP Reno with fast retransmit and recovery,
- TCP New Reno,
- TCP with selective acknowledgments (SACK).

ATM network designers as well as service providers must choose the optimal ATM services for efficient TCP transport. This document describes the design choices and performance analysis results of various options available to TCP over ATM networks.

We have shown that vanilla TCP over the UBR service category achieves low throughput and low fairness. This is because during packet loss, TCP loses time waiting for its coarse granularity retransmission timeout.

In the presence of bursty packet losses, fast retransmit and recovery (FRR) (without SACK) further hurts TCP performance over UBR for long delay-bandwidth product networks. This is because after two fast retransmissions, the congestion window is too small to send out new packets that trigger duplicate acks. In the absence of duplicate acks, the third lost packet is not retransmitted and a timeout occurs at a small window. This results in congestion avoidance with a small window, which is very slow for long delay networks.

There are several ways to significantly improve the TCP throughput over UBR. These include,

- Frame-level discard policies,

- Intelligent buffer management policies,
- TCP New Reno,
- TCP SACK and
- Guaranteed rates.

Frame level discard policies such as early packet discard (EPD) improve the throughput significantly over cell-level discard policies. However, the fairness is not guaranteed unless intelligent buffer management using per-VC accounting is used.

Throughput increases further with more aggressive New Reno and SACK. SACK gives the best performance in terms of throughput. We have shown for long delay paths, the throughput improvement due to SACK is more than that from discard policies and buffer management. When several TCP flows are multiplexed on to a few VCs, fairness among the TCP flows can be provided by the routers at the edges of the ATM network, while VC level fairness must be provided by the ATM network using either buffer management or per-VC queuing.

The fourth method of improving the UBR performance is the so called "guaranteed rate" (GR) in which a small fraction of the bandwidth is reserved in the switches for the UBR service. This bandwidth is shared by all UBR VCs. Using guaranteed rates helps in the presence of a high load of higher priority traffic such as Constant Bit Rate (CBR) or Variable Bit Rate (VBR) traffic. We have shown that reserving just a small fraction, say 10%, of the bandwidth for UBR significantly improves TCP performance. This is because the reserved bandwidth ensures that the flow of TCP packets and acknowledgments is continuous and prevents TCP timeouts due to temporary bandwidth starvation of UBR. Note that this mechanism is different from

the GFR service category where each VC (rather than the entire UBR class) has a minimum rate guarantee.

The Guaranteed Frame Rate (GFR) service category is designed to provide per-VC minimum rate guarantees. We have proposed the Differential Fair Buffer Allocation (DFBA) scheme that uses FIFO buffers and provides MCR guarantees to the VCs. We have presented simulation results for large number of TCP sources, multiple TCPs perVC, a large MCR allocation, small and large RTTs and heterogeneous RTTs with DFBA. In each case DFBA can control the rates to the desired level for SACK TCP traffic.

For TCP over ABR, we have considered the use of virtual source / virtual destination to control buffers in the network. Our studies have indicated that VS/VD can be used to isolate long-delay segments from terrestrial segments. This helps in efficiently sizing buffers in routers and ATM switches. As a result, terrestrial switches only need to have buffers proportional to the bandwidth-delay products of the terrestrial segment of the TCP path. Switches connected to the satellite VS/VD loops must have buffers proportional to the satellite delay-bandwidth products.

8.3 Limitations and Future Work

While the work conducted in this project has made significant contributions to the the development of ATM technology, a portion of the space of potential solutions for efficient traffic management in high speed networks still remains unexplored. The approach taken here is one of working with the ATM Forum and IETF standards and extending them to improve the performance of TCP data transfer over ATM. The existing IETF standards were designed using technology and knowledge available

to the designers of the original ARPANET almost two decades ago. With higher bandwidth, faster processors and a fundamental change in the traffic patterns across the Internet, the legacy IETF standards are quickly becoming inadequate to meet the requirements of a multiservice network.

TCP congestion control algorithms are based on window based flow control with packet loss as the sole indicator of congestion. These algorithms are inherently dependent on the bandwidth-delay product of the network. Longer delay TCP connections need more network resources to achieve the same throughput as shorter delay TCP connections. The current Internet paradigm is to use minimal state within the network. The network elements do not have any knowledge about round trip times of its flows. Floyd [31] describes TCPs inherent limitation with multiple congested gateways. TCPs traversing multiple congested nodes tend to get lower throughput than other TCPs that share some of the same bottlenecks, but traverse fewer congested nodes. The buffer management policies (Selective Drop, Fair Buffer Allocation, Differential Fair Buffer Allocation) presented and analyzed in this research suffer from the same problem. This is a fundamental problem with TCP and not the buffer management schemes. The philosophy of *TCP Friendly Buffer Management* presented in chapter 6 proposes approaches to solve the problem by designing network based solutions. DFBA is in fact a step towards designing TCP friendly solutions in the network.

Recent researchers have focused on the topic of *TCP Friendly Congestion Control* for other Internet applications [96, 99, 95, 85, 13]. These papers propose congestion control schemes and transport protocols for applications such as real time video [95] which have traditionally used UDP. The schemes presented here try to be consistent

with the $1/sqrt(p)$ rule in doing congestion control, so that their flows can be modeled as TCP flows. Traffic from such schemes can coexist with legacy TCP traffic without taking up an unfair share of the network capacity.

However, to make significant progress in traffic management of broadband networks, we must re-examine all the fundamental principles in congestion control for end systems and network elements in the context of a network that supports several services and QoS guarantees. The network elements now have available to them, several new ways of providing feedback to the end systems. The set of feedback mechanisms includes not only drop based feedback, but also bit based and explicit rate feedback. The questions posed in [56] for *end systems* were the following:

- How to reach steady state quickly?
- How to maintain steady state in the presence of changing network conditions?
- How to use the feedback signals from the network?
- How often to change sending rate based based on feedback?
- How much to change the sending rate?

A potential solution space is to design a new end-system protocol that can use some of the feedback capabilities of current technologies such as the ATM ABR service. Although deployment issues in changing the TCP stack appear implausible at first, an incremental solution that performs sophisticated traffic management should be deployable. The TCP Explicit Congestion Notification (ECN) [84] proposal in the IETF suggests the use of 2 bits that can be set by the network to indicate congestion.

Floyd [31] also indicates the use of a rate based TCP algorithm for improved TCP performance for long RTT connections.

The ultimate goal of providing Quality of Service can only be achieved with an integrated solution that combines network control and feedback with cooperative end-system control.

APPENDIX A

ATM QoS: A Description of Current Standards

ATM networks have been designed to carry voice, video and data traffic in an integrated fashion and satisfy the QoS guarantees for the various traffic types. Specification and measurement of Quality of Service is a difficult problem for both the user and the network designer. The user must map application level requirements to network level performance objectives. These objectives must be understandable, implementable and measurable in the variety of networks. Understanding the QoS model is critical for designing a traffic management system that supports the services enabled by the QoS model.

The International Telecommunications Union (ITU-T) has chosen ATM as the technology to implement Broadband Integrated Services Digital Networks (B-ISDN). ATM networks have been designed to provide real time and non-real time services to various kinds of applications. These applications range from those having stringent service requirements like conversational voice, to those with very simple reliability requirements like email transfer. For a majority of the applications, it is difficult to precisely define the requirements for a service that transports the application's packets across a network. The requirements expected by an application from a network service are called Quality of Service (QoS) requirements. The goal of a networking technology

is to transfer application data and satisfy its QoS requirements, at the same time, efficiently utilizing the network resources. The ATM QoS model has been a topic of extensive discussion at the ATM Forum and the ITU-T. Each organization has its own model. The primary issue being addressed by these bodies is that of the interoperability of the two QoS models. The issue of validity of either model is also of some concern.

In this chapter, we discuss the Quality of Service (QoS) model for ATM networks. We first provide a comprehensive overview of the QoS model specified by the ITU-T and the ATM Forum. We discuss the QoS parameters used to specify the requirements of various applications using ATM services. We describe how the user specifies traffic characteristics and quality of service requirements when requesting a connection. We then describe the services offered by ATM, as specified by the ATM Forum and ITU-T. These include the service categories specified by the ATM Forum and the ATM Transfer Capabilities (ATCs) specified by ITU-T. We discuss interoperability of QoS parameters and their appropriateness for providing adequate QoS support.

A.1 The QoS Problem

The QoS problem for real-time as well as non real-time applications can be analyzed from three perspectives – the sender, the receiver and the carrier. Figure A.1 illustrates this visualization in the form of a QoS triangle. Senders want to send traffic with arbitrarily varying data rates and high burstiness. The senders cannot always specify the traffic characteristics of the data originating from them. The receivers may expect data to arrive with low delay and high throughput. In addition, the receivers may also be sensitive to variations in packet delay (jitter). The network operators, on

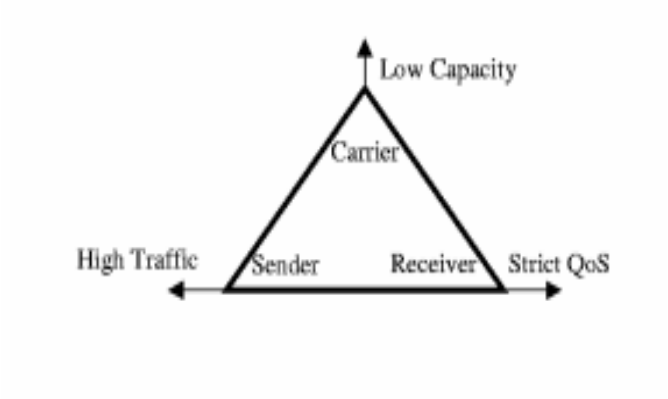


Figure A.1: QoS Triangle

the other hand, want to minimize their infrastructure and cost, by maximizing their multiplexing gain. In fact, if any one of these three entities relax their requirements, traffic management could become an easier problem.

The lack of QoS specifications of the sender, strict QoS requirements by the receiver and high network utilization of the carrier are conflicting parameters that lead to the following issues:

1. *Lack of a precise definition of QoS metrics.* Most real time applications like audio and video serve a human end user. There are very few quantitative metrics that can describe the quality of an audio or a video reception. It is not easy to specify requirements for the network to meet the needs of a particular application. Data applications typically measure performance in terms of throughput to the end user and packet/cell loss ratio. However, high throughput and low cell loss in the network do not necessarily correspond to high throughput and low packet loss in the application. Network throughput could be wasted by excessive retransmissions by higher layer protocols like TCP. A single cell drop

in the network will result in the entire AAL5 frame being dropped at the edge of the ATM network.

2. *Lack of precise definitions of the tolerances for QoS metrics.* Even if QoS metrics can be defined, it is difficult to quantify acceptable tolerances for the metrics. For example, most broadcast video standards refer to standard video quality of 30 frames/sec delivered to the user. However, the point at which the video quality becomes unacceptable, depends on the individual watching the video. Moreover, if part of a frame is lost by a cell-based transmission technology like ATM, should the entire frame be discarded? Also, is it better to lose a few cells from many frames than many cells from a few frames? The answers to these questions are subjective and cannot be easily expressed in a mathematical form.

3. *Complexity in measuring QoS.* When QoS metrics are defined, they are usually defined as statistical entities. For example, the ATM Forum specifies the maximum cell transfer delay as a percentile of the cell delay distribution measured at every ATM node. It is difficult to calculate a percentile in real time, because percentile computations typically require sorting, which is an expensive operation.

4. *Problems with the interoperability of QoS metrics.* The data of a single application session can be transferred over several networks, each based on a different technology. The cumulative quality of service delivered by the networks must correspond to the quality of service requested by the application. This process requires the interoperability of application-network QoS as well as network-network QoS. Although there are several common QoS metrics like throughput

and delay, the precise definitions of these may vary from one network technology to another. In fact, even for the same technology, several standards organizations may define the metrics differently. For example, the ATM Forum and the ITU-T definitions for ATM QoS differ on several issues and an interoperability issue arises.

5. *Problems with QoS accumulation.* Not only must the delivered QoS be measured by each node in the application packet's path, the end-to-end QoS delivered to the application must be calculated as the cumulative effect of individual QoS values. Accumulation of statistical quantities is often complex and may not have a closed form solution. For example, the point-to-point cell delay variation at each node is defined by the ITU-T as the difference between two quantiles of the cell delay distribution. The end-to-end cell delay variation suffered by the cells of a connection must be calculated from the quantiles of the sum of the individual delay distributions. Adding distributions is non-trivial and several approximations must be used to calculate the end-to-end quantiles.

6. *Utilizing the network efficiently.* Circuit switched technology with sufficient bandwidth can meet the most stringent QoS requirements. However, circuit switched networks are not an efficient technology for application streams that have bursty characteristics. For example, packetized voice can be transferred over circuit emulation or constant bit rate services. However, network capacity can be used more efficiently by the multiplexing gains provided by variable bit rate services. The stochastic nature of bursty traffic makes it difficult to provide strict QoS guarantees while achieving multiplexing gains.

7. *Difficulty in specifying traffic characteristics.* To guarantee QoS for a connection, the network must provision resources based on the expected load offered by the connection. Sometimes, the network's resources may be allocated for a connection's lifetime, while at other times, the network may reallocate resources periodically. In most cases, it is impossible for the network to predict the exact amount of traffic received from a bursty application, especially for the short term. For a real time application, it is also difficult for the application to notify the network of the generated traffic within sufficient time to allow the network to reallocate resources. As a result, there may be considerable inaccuracy in network resource allocation. Underallocation of resource results in loss of network efficiency and overallocation results in the degradation of the offered quality of service.

A.2 The ATM Forum Quality of Service Model

The ATM Forum specifies that ATM connections belong to ATM service categories that support certain Quality of Service (QoS) requirements. The ATM-Forum Traffic Management Specification 4.0 defines six service categories for ATM networks. Each service category is defined using a traffic contract and a set of QoS parameters. The *traffic contract* is a set of parameters that specify the characteristics of the source traffic. This defines the requirements for compliant cells of the connection. The *QoS parameters* are negotiated by the source with the network and are used to define the expected quality of service provided by the network. For each service category, the network guarantees the negotiated QoS parameters if the end system complies

with the negotiated traffic contract. For non-compliant traffic, the network need not maintain the QoS objective.

A.2.1 Traffic Parameters

The ATM Forum specifies traffic parameters to describe the characteristics of an ATM traffic source. The traffic parameters were originally designed with the following objectives:

1. The parameters should be able to characterize the properties of traffic generated by the end-system.

2. It should be easy to perform conformance testing and policing (Usage Parameter Control – UPC) of the traffic parameters, at the entry to a network.

3. The network should be able to allocate resources and perform connection admission control (CAC) based on the values of the traffic parameters.

The traffic management specification 4.0 defines four traffic parameters – Peak Cell Rate (PCR), Sustainable Cell Rate (SCR), Maximum Burst Size (MBS) and Minimum Cell Rate (MCR). A tolerance value for PCR, called Cell Delay Variation Tolerance (CDVT), is also specified. This defines the allowable variation from PCR for a connection's traffic to be conforming. The network determines conformance to PCR and SCR by using the Generic Cell Rate Algorithm (GCRA). The GCRA is a version of the token bucket algorithm that determines if an incoming cell is conforming to the traffic specifications. If a cell is non-conforming, the usage parameter control (UPC) function of the network may tag the cell by setting the CLP bit in the cell.

The untagged and the aggregate cell streams are referred to as the CLP0 and CLP0+1 cell streams respectively.

A.2.2 QoS Parameters

In addition to specifying the traffic parameters for a connection, the ATM Forum also defines six QoS parameters that specify the network performance objectives for the connection Cell Loss Ratio (CLR), peak-to-peak Cell Delay Variation (peak-to-peak CDV), maximum Cell Transfer Delay (max CTD), Cell Error Ratio (CER), Severely Errored Cell Block Ratio (SECBR) and Cell Misinsertion Rate (CMR). Of these, the CDV, CTD and the CLR parameters are negotiated during connection setup, while the others are specified by the network.

The CTD and the CDV parameters are defined as quantiles of the measured cell delays at each queuing point in the network. The cell delay at a queuing point consists of a fixed delay and a variable queuing delay. The measured delay can be expressed as a delay distribution as shown in figure. The max CTD is defined as the α quantile of the delay distribution. α is small, but is left unspecified by the ATM Forum. α represents the proportion of cells delivered late by the node. For real time connections requiring delay guarantees, if a cell is delivered late, it is of no use and can be considered lost. As a result, the value of $(1-\alpha)$ can be bounded by the CLR parameter specified for the connection.

The cell loss ratio is defined for the lifetime of a connection as the ratio of the cell lost to the total cell transmitted by the source. The CLR is further classified as being applicable for the CLP0 cell stream or the CLP0+1 cell stream.

A.2.3 ATM Service Categories

Based on the traffic parameters and the QoS parameters, the ATM forum specifies five service categories for the transport of ATM cells. The *Constant Bit Rate (CBR)* service category is defined for traffic that requires a constant amount of bandwidth, specified by a Peak Cell Rate (PCR), to be continuously available. The network guarantees that all cells emitted by the source that conform to this PCR will be transferred by the network with minimal cell loss and within fixed bounds of maxCTD and peak-to-peak CDV.

The *real time Variable Bit Rate (VBR-rt)* class is characterized by PCR (and its associated tolerance), Sustained Cell Rate (SCR) and a Maximum Burst Size (MBS) in cells that controls the bursty nature of VBR traffic. The Burst Tolerance (BT) parameter is used to test the conformance of the traffic to MBS. BT is defined as

$$BT = \lceil (MBS - 1)(1/SCR - 1/PCR) \rceil$$

The network attempts to deliver cells of conforming connections within fixed bounds of maxCTD and peak-to-peak CDV.

Non-real-time VBR sources are also specified by PCR, SCR and MBS, but are less sensitive to delay and delay variation than the real time sources. The network does not specify any delay and delay variation parameters for the VBR-nrt service. Both VBR-rt and VBR-nrt are further divided into three categories based on whether CLR0 or CLR0+1[7] is specified as the CLR performance objective for the network, and whether tagging is allowed by the user or not.

[7]The suffix 0 or 0+1 added to a QoS parameter denotes that the parameter is applicable to the cells with the CLP bit set to 0 (CLP0 cell stream) or to all cells (CLP0+1 cell stream)

The Available Bit Rate (ABR) service category is specified by a PCR and Minimum Cell Rate (MCR) which is guaranteed by the network. The bandwidth allocated by the network to an ABR connection may vary during the life of a connection, but may not be less than MCR. ABR connections use a rate-based closed-loop feedback-control mechanism for congestion control. The network tries to maintain a low Cell Loss Ratio by changing the allowed cell rates (ACR) at which a source can send.

The Unspecified Bit Rate (UBR) service category is intended for best effort applications and this category does not support any service guarantees. UBR has no built in congestion control mechanisms. The UBR service manages congestion by efficient buffer management policies in the switch. A new service called Guaranteed Frame Rate (GFR) is being introduced at the ATM Forum and the ITU-T. GFR is based on UBR, but guarantees a minimum rate to connections. The service also recognizes AAL5 frames and performs frame level dropping as opposed to cell level dropping. Table A.1 shows the traffic and QoS parameters for the various service categories specified by the ATM Forum.

A.3 The ITU-T Quality of Service Model

The ITU-T ATM model is mainly specified in three ITU-T recommendations I.356, I.362, I.371. Of these, I.362 has been discontinued by the ITU-T, but its implementations still exist in current networks. Also, the ATM Forum TM4.0 document specifies support for the I.362 model.

A.3.1 I.362 Service Classes

I.362 defines four ATM service classes (A, B, C and D) based on the following three parameters:

	CBR	Nrt VBR	Nrt VBR	Nrt VBR	Rt VBR	Rt VBR	Rt VBR	ABR	GFR	UBR
Traffic Parameter										
PCR(0+1)	X	X	X	X	X	X	X	X	X	X
CDVT(0+1)	X	X	X	X	X	X	X	X	X	X
SCR(0)			X	X		X	X			
BT(0)			X	X		X	X			
SCR(0+1)		X			X					
BT(0+1)		X			X					
MCR(0+1)								X		X
Tagging				X			X			X
QoS Parameter										
MaxCTD	X				X	X	X			
p-p CDV	X				X	X	X			
CLR0			X X		X	X			X	
CLR0+1	X	X		X				X		

Table A.1: The ATM Forum QoS Model

Timing relation between source and destination	Class A Required	Class B	Class C Not Required	Class D
Bit Rate	Constant	Variable		
Connection mode	Connection oriented			Connectionless

Table A.2: I.362 Service Classes

1. Whether a timing relation required between the source and the destination, i.e., delay requirement.

2. Whether the traffic pattern is constant rate or variable rate.

3. Whether the service is connectionless or connection-oriented.

Table A.2 lists the requirements of the four service classes specified by I.362.

I.362 also provides the following examples of applications that would use each of the service classes.

- **Class A.** Circuit emulation, constant bit rate video
- **Class B.** Variable bit rate video and audio
- **Class C.** Connection-oriented data transfer
- **Class D.** Connectionless data transfer

Thus, I.362 defines the service classes based on common application types that must be supported by an ATM network. Signaling for the individual parameters in the traffic classes is specified in ITU recommendation Q.2931. However, I.362 does

not define default parameter values and tolerances for each of the service classes. This makes the service class definition open to individual interpretation.

A.3.2 ATM Forum support for I.362

The ATM Forum TM4.0, specifies support for the ITU-T service classes. The specification further defines QoS classes for supporting the I.362 service classes. An ATM Forum QoS class consists of a set of QoS parameters with specified values. These parameters are selected from the ATM Forum defined QoS parameters – maxCTD, CLR and point-to-pointCDV. Two values of CLR (CLR0 and CLR0+1) may be specified. All other parameters are specified for the aggregate cell stream. A QoS class with specified values of one or more QoS parameters is called a *specified QoS class*. A specified class may contain objective values for any subset of the ATM Forum QoS parameters. In addition, an unspecified QoS class can also be defined where none of the QoS parameters have any specified values. The network may set internal performance objectives for the unspecified class, but no explicit QoS is signaled by the user. Such a class is typically used for best effort traffic.

An ATM user should be able to request an ATM Forum QoS class during connection establishment. Any number of specified QoS classes may be supported by the network, but at most one unspecified QoS class can be supported. An ATM Forum TM4.0 compliant service should provide at least one specified QoS class for each I.362 service category. Four specified QoS classes, specified QoS Class 1...specified QoS Class 4, are defined by TM4.0 corresponding to service classes A...D respectively. These four QoS classes are mandatory requirements of an ATM forum compliant ATM service. TM4.0 attempts to provide interoperability between ITU and ATM

Service Category (TM 4.0)	QoS Class (TM4.0)	Service Class (I.362)
CBR	1	A
rt-VBR	2	B
nrt-VBR	3	C
ABR	3	C
GFR	3	D
UBR	4	D

Table A.3: A proposed mapping for ATM Forum and I.362

Forum standards. A user can thus initiate a connection based on QoS classes. However, the exact mapping of QoS classes and service categories has not been specified by either standard body. Table 3 lists a possible mapping based on the definitions of the service categories and service classes. This can be used as a guideline by the user to specify QoS classes for connections belonging to particular service categories. TM4.0 does not specify default values of the QoS parameters for QoS Classes 1...4. As a result, a connection passing through different networks may receive different service along its path.

I.362 has since been replaced by I.356 and I.371.

A.3.3 I.356 QoS Classes

ITU specification I.356 also defines six QoS parameters that are called ATM cell transfer performance parameters:

- *Mean Cell Transfer Delay (mean CTD)*: Mean CTD is defined as the arithmetic average of the measured cell transfer delays.

	Mean CTD	2-pt CDV	CLR (0+1)	CLR (0)	CER	CMR	SECBR
Default	None	None	None	None	4×10^{-6}	1 per day	10^{-4}

Table A.4: I.356 cell transfer performance objectives – default values

- *Two point Cell Delay Variation (2-pt CDV):* This is the difference between the upper and lower α quantiles of the CTD distribution.
- *Cell Loss Ratio (CLR)*: Three CLRs are defined for the CLP0, CLP1 and CLP0+1 cell streams respectively.
- *Cell Error Ratio (CER)*
- *Cell Misinsertion Rate (CMR)*
- *Severely Errored Cell Block Ratio (SECBR)*

The definitions for CER, CMR and SECBR are the same as those of the ATM Forum. I.356 also allows the user to signal a maximum acceptable CTD (maxCTD) parameter. However, the use and accumulation of this parameter is open to discussion. Table A.4 shows the default values assigned by I.356 to some of the ATM cell transfer performance parameters.

Based on the six performance objectives, I.356 defines four QoS classes with defined values for some or all of the QoS parameters.

1. **Class 1 (stringent class):** This class is associated with real time connections. It specifies a limit on the mean CTD, 2-pt CDV and the cell loss ratio for the CLP0+1 stream.

Class	Mean CTD	2-pt CDV	CLR (0+1)	CLR (0)	CER	CMR	SECBR
Class 1	400 ms	3 ms	3×10^{-7}	None	Default		
Class 2	U	U	10^{-5}	None	Default		
Class 3	U	U	U	10^{-5}	Default		
Class U	U	U	U	U	U	U	

Table A.5: I.356 QoS Classes

2. **Class 2 (tolerant class):** In this class, only a CLR objective for the aggregate (CLP0+1) cell stream is specified. The network is not required to support performance objectives for delay, delay variation and CLR0.

3. **Class 3 (bi-level class):** In this class, the network distinguishes between tagged and untagged cells. A CLR objective for the CLP0 stream must be specified by the network.

4. **Class U (unspecified class):** The network does not provide any guarantees to connections belonging to this class. This class is typically used by best effort connections.

The I.356 class 1 corresponds to ATM Forums QoS Class 1. Classes 2, 3 and U have no counterpart in the ATM Forum. Also, unlike the ATM Forum, the ITU does not distinguish between real time and non-real time traffic in classes 2 and 3. Table A.5 shows the default values for the I.356 QoS classes.

A.3.4 I.371 ATM Transfer Capabilities

I.371 defines ATM Transfer Capabilities (ATC) that loosely correspond to the ATM Forum service categories. The ATCs are specified in terms of traffic parameters similar to the ATM Forum traffic parameters. In addition to the Peak Cell Rate (PCR), the Sustainable Cell Rate (SCR), the Maximum Burst Size (MBS) and the Minimum Cell Rate (MCR), rate tolerances or delay variation tolerances, CDVT_{PCR} and CDVT_{SCR} are also specified for PCR and SCR respectively. The MBS is also specified in the form of the Inter Burst Time (IBT) as

$$IBT = \lceil (MBS - 1)(1/SCR - 1/PCR) \rceil$$

The PCR and MCR are specified for the aggregate cell stream and separate SCRs are specified for the CLP0 and the CLP0+1 streams. Based on the traffic characteristics, I.371 defines four basic ATM Transfer Capabilities. However, I.371 does not associate ATCs with QoS classes. The network is free to support one or more QoS classes for each ATC. As a result, the user must signal an ATC as well as a QoS class during connection establishment.

Deterministic Bit Rate Transfer Capability (DBR). The DBR transfer capability is analogous to the ATM Forums CBR service category. It is intended for applications with constant bandwidth requirement that can be specified by a PCR. However, the DBR transfer capability is not restricted to CBR type QoS guarantees. In particular, there is no requirement for delay and delay variation guarantees. The network is only required to guarantee a CLR bound on the aggregate cell stream. The network is expected to meet the negotiated QoS for a conforming DBR connection. DBR does not distinguish between CLP0 and CLP0+1 flows, nor does a DBR network

support tagging of the cells. During connection establishment, the user specifies the traffic parameters and negotiates a QoS class supported by the network. The network may provide the choice of more than one QoS class for DBR connections.

Statistical Bit Rate Transfer Capability (SBR) SBR is analogous to the VBR service category. During connection establishment, the SBR user specifies a PCR and SCR/IBT. The PCR is defined for the aggregate CLP0+1 cell stream, while the SBR/IBT may be defined either for the CLP0 or CLP0+1 cell stream. In addition, when SBR/IBT are specified for the CLP0 stream, the tagging option may be specified by the user wherein the network is allowed to tag cells that are non-conforming to the traffic contract. Also, in this case, the network may selectively discard CLP1 cells over CLP0 cells. Again, the network may support multiple QoS classes for the SBR transfer capability. If the SCR/IBT parameters are specified for the CLP0+1 stream, then a QoS commitment on CLR is only made for the CLP0+1 cell stream. If SCR/IBT is specified for the CLP0 stream, then CLR is committed only for CLP0 cells. The QoS for CLP0+1 is left unspecified. In either case, I.371 allows the network to support a QoS commitment on delay that applies to the CLP0+1 stream.

ATM Block Transfer Capabilities (ABT) The ABT service does not have a counterpart in the ATM Forum. This service defines ATM blocks as groups of cells separated by Resource Management (RM) cells. The traffic parameters of a connection are negotiated on an ATM block basis and the network must treat an ATM block as a DBR connection with a peak cell rate called the Block Cell Rate (BCR). As a result, the QoS received by an ABT block is the same as that received by a DBR connection. During connection establishment, the user must negotiate the following static parameters for the lifetime of the connection:

- The PCR and associated tolerance for the CLP0+1 cell flow.
- The maximum frequency of BCR re-negotiation. This done by specifying the PCR and tolerance of the RM cell flow.
- An SCR/IBT for the CLP0+1 cell flow. The SCR may be set to zero. The SCR acts like a long term minimum cell rate parameter for the connection. The network must attempt to provide a committed bandwidth of SCR to the connection. In addition, a BCR greater than SCR may be negotiated.

Two types of ABT capabilities are specified:

1. **ABT with delayed transmission (ABT/DT).** Here, the BCR is re-negotiated by the user or the network by the sending of an RM cell that specifies the BCR requested for the next block. The network sends these RM cells back to the user with a positive or a negative acknowledgment for the requested BCR. The source can send the next ATM block at BCR once it has received a positive acknowledgment by the network. The ABT/DT transfer capability provides cell level as well as block level guarantees to the connection. The cell level guarantee specifies that the cells within an ATM block will be transferred by the network as a DBR connection with PCR equal to BCR. The block level guarantee specifies that when SCR is greater than 0 and cells across blocks are conforming to SCR, then the network must accept new BCR renegotiations within a finite time interval. This time interval is established during connection establishment.
2. **ABT with immediate transmission (ABT/IT).** Here the user does not wait for the return of the RM cell from the network, but sends the data at

BCR immediately following the RM cell that contains the value for BCR. If the network has sufficient resources to transmit the block at BCR, it does so, otherwise, it discards the entire block. The network guarantees that if the cell rate is less than SCR, then the blocks are transferred with a high probability. The cell level guarantee similar to ABT/DT also applies to ABT/IT. In addition, ABT/IT users can set an *elastic* bit in RM cells that indicates if the network is allowed it shape the block. If the elastic bit is set to 1 (no shaping allowed), then the network must also guarantee a cell delay variation bound for the cells within a block. The block level guarantee provided by the network specifies that when SCR is greater than 0 and cells in block are conforming to SCR, then a new BCR should be accepted with a certain probability. This probability is also specified as QoS parameter during connection establishment.

Available Bit Rate Transfer Capability (ABR). There is no significant difference between the ATM Forums ABR service category and ITU-Ts ABR transfer capability.

Table A.6 shows the respective traffic parameters for I.371 ATCs. Table A.7 shows the ITU recommended QoS classes for each ATC.

Table A.8 shows a possible mapping of ATM Forum service categories to QoS classes. Note that since ITU does not provide a QoS class that differentiates between CLP0 and CLP0+1 traffic and support real time guarantees. As a result, there is no clear support for a user requesting rt-VBR.2 and rt-VBR.3 connection in an ITU-T network.

Interworking solutions between QoS classes and QoS parameters must be developed for seamless interoperability between private and public networks. The ATM

	DBR	SBR1	SBR2	SBR3	ABT/DT	ABR
PCR(0+1)	X	X	X	X	X	X
$\text{CDVT}_{\text{PCR}}(0+1)$	X	X	X	X	X	
SCR(0)			X	X		
IBT(0)			X	X		
$\text{CDVT}_{\text{SCR}}(0)$			X	X		
SCR(0+1)		X			X	
IBT(0+1)		X			X	
$\text{CDVT}_{\text{SCR}}(0+1)$		X			X	
MCR(0+1)						X
Tagging				X		X

Table A.6: I.371 ATCs

ATC	QoS Class
DBR, SBR1, ABT/DT, ABT/IT	Class 1
DBR , SBR1, ABT/DT, ABT/IT	Class 2
SBR2, SBR3, ABR	Class 3
Any ATC	U Class

Table A.7: QoS classes for carrying ATCs

Service Category	QoS Class
CBR	1
Rt VBR.1	1
Rt VBR.2	?
Rt VBR.3	?
Nrt VBR.1	2
Nrt VBR.2	3
Nrt VBR.3	3
ABR	3, U
UBR	U
GFR	3

Table A.8: QoS classes for service categories

Forums traffic management group is currently discussing proposals for the alignment and interoperability of the ITU and ATM Forum's QoS models.

The ATM QoS parameters described above can be used to measure the conformance of an ATM service to the requirements of the ITU-T and the ATM Forum. These parameters can be used by the network provider to measure the capabilities of the network in meeting the service specifications. However, a given level of performance at the ATM layer does not necessarily correspond to the same level of performance at higher layers.

APPENDIX B

Pseudocode

B.1 Early Packet Discard

List of variables:

X = Total buffer occupancy at any given time

R = EPD threshold

i = index of VC

Middle$_i$ = TRUE if at least one cell of VC_i has been received

Drop$_i$ = TRUE if the remaining cells of this frame of VC_i are being dropped

```
        X ← 0
For each VC i
        Middle_i ← FALSE
        Drop_i ← FALSE
```

Figure B.1: EPD: Initialization

```
X ← X − 1
```

Figure B.2: EPD: Cell dequeued

```
IF (cell is NOT the last cell of the pkt) THEN
        IF (NOT (Middle_i)) THEN

                IF ((X ≤ R) THEN

                        Middle_i ← TRUE
                        X ← + 1
                        Enqueue cell
                ELSE
                        Middle_i ← TRUE
                        Drop_i ← TRUE
                        Drop cell
        ELSE
                IF (Drop_i) THEN
                        Drop cell
                ELSE
                        Enqueue cell if possible
                        IF (enqueued) THEN
                                X ← X + 1
                        ELSE
                                Drop_i ← 1
ELSE
        Enqueue cell if possible
        Middle_i ← FALSE
        Drop_i ← FALSE
        IF (Enqueued) THEN
                X ← X + 1
```

Figure B.3: EPD: Cell Received

B.2 Selective Drop and Fair Buffer Allocation

List of variables:

X = Total buffer occupancy at any given time

R = Threshold

K = Buffer size

N = Count of active VCs

Z = Parameter

i = index of VC

X_i = Per-VC buffer occupancy ($X = \Sigma X_i$)

Middle$_i$ = TRUE if at least one cell of VC_i has been received

Drop$_i$ = TRUE if the remaining cells of this frame of VC_i are being dropped

```
        X ← 0
        N ← 0
For each VC i
        Middle_i ← FALSE
        Drop_i ← FALSE
        X_i ← 0
```

Figure B.4: SD and FBA: Initialization

$X_i \leftarrow X_i - 1$
$X \leftarrow X - 1$
IF $(X_i = 0)$ THEN $N \leftarrow N - 1$

Figure B.5: SD: Cell dequeued

```
IF (cell is NOT the last cell of the pkt) THEN
        IF (NOT (Middle_i)) THEN
                IF ((X ≤ R)∨
                  (X_i × N/X ≤ Z) THEN        (SD)
                  (X_i × N/X ≤ Z × (K − R)/(X − R)) THEN      (FBA)
                        Middle_i ← TRUE
                        IF (X_i = 0) THEN
                                N ← N + 1
                        X_i ← X_i + 1
                        X ← + 1
                        Enqueue cell
                ELSE
                        Middle_i ← TRUE
                        Drop_i ← TRUE
                        Drop cell
        ELSE
                IF (Drop_i) THEN
                        Drop cell
                ELSE
                        Enqueue cell if possible
                        IF (enqueued) THEN
                                IF (X_i = 0) THEN
                                        N ← N + 1
                                X_i ← X_i + 1
                                X ← X + 1
                        ELSE
                                Drop_i ← 1
ELSE
        Enqueue cell if possible
        Middle_i ← FALSE
        Drop_i ← FALSE
        IF (Enqueued) THEN
                IF (X_i = 0) THEN
                        N ← N + 1
                X_i ← X_i + 1
                X ← X + 1
```

Figure B.6: SD and FBA: Cell Received

B.3 Differential Fair Buffer Allocation

List of variables:

N = Count of active VCs

X = Total buffer occupancy at any given time

L = Low buffer threshold

H = High buffer threshold

i = index of VC

MCR_i = MCR guaranteed to VC_i

W_i = Weight of VC_i = MCR_i/(GFR capacity)

$W = \Sigma W_i$

X_i = Per-VC buffer occupancy ($X = \Sigma X_i$)

Z_i = Parameter ($0 \leq Z_i \leq 1$)

Middle_i = TRUE if at least one cell of VC_i has been received

Drop_i = TRUE if the remaining cells of this frame of VC_i are being dropped

$X \leftarrow 0$
$N \leftarrow 0$
For each VC i
    $\text{Middle}_i \leftarrow FALSE$
    $\text{Drop}_i \leftarrow FALSE$
    $X_i \leftarrow 0$

Figure B.7: DFBA: Initialization

$X_i \leftarrow X_i - 1$
$X \leftarrow X - 1$
IF $(X_i = 0)$ THEN $N \leftarrow N - 1$
    $W \leftarrow W - W_i$

Figure B.8: DFBA: Cell dequeued

```
IF (cell is NOT the last cell of the pkt) THEN
        IF (NOT (Middle_i)) THEN
                IF ((X ≤ L)∨
                  (X_i > X × W_i/W ∧ X < H ∧ rand_var > P{drop} ∧ CELL.CLP = 0)∨
                  (X_i < X × W_i/W ∧ X < H ∧ CELL.CLP = 0)) THEN
                        Middle_i ← TRUE
                        IF (X_i = 0) THEN
                                N ← N + 1
                                W ← W + W_i
                        X_i ← X_i + 1
                        X ← + 1
                        Enqueue cell
                ELSE
                        Middle_i ← TRUE
                        Drop_i ← TRUE
                        Drop cell
        ELSE
                IF (Drop_i) THEN
                        Drop cell
                ELSE
                        Enqueue cell if possible
                        IF (enqueued) THEN
                                IF (X_i = 0) THEN
                                        N ← N + 1
                                        W ← W + W_i
                                X_i ← X_i + 1
                                X ← X + 1
                        ELSE
                                Drop_i ← 1
ELSE
        Enqueue cell if possible
        Middle_i ← FALSE
        Drop_i ← FALSE
        IF (Enqueued) THEN
                IF (X_i = 0) THEN
                        N ← N + 1
                        W ← W + W_i
                X_i ← X_i + 1
                X ← X + 1
```

Figure B.9: DFBA: Cell Received

B.4 Virtual Source / Virtual Destination

```
Case 1: Data cell received from link
        CI_VC_ij  ← EFCI state of cell
        Send cell to switching fabric
Case 2: Data cell received from switch fabric
        if (q_ij = 0 ∧ time_to_send not scheduled) then
                Schedule: time_to_send_ij  ← now()
```

Figure B.10: VS/VD: Data Cell Received

```
cell.ER  ← Min(cell.ER, ER_ij^feedback)
if (turnaround_ij) then
        CI_TA_ij ← CI_TA_ij ∨ CI_VC_ij
        Send BRM cell
        CI_VC_ij  ← 0
CCR_TA_ij  ← cell.CCR
MCR_TA_ij  ← cell.MCR
ER_TA_ij  ← cell.ER
CI_TA_ij  ← cell.CI
NI_TA_ij  ← cell.NI
turnaround_ij  ← TRUE
if (time to send not scheduled) then
        Schedule Event: time_to_send_ij  ← now()
Discard FRM cell
```

Figure B.11: VS/VD: FRM cell received

```
if (cell.CI=0) then
        ACR_ij ← ACR_ij − ACR_ij × RDF_ij
else if (cell.NI=1) then
        ACR_ij = ACR_ij + RIF_ij × PCR_ij
        ACR_ij = Min(ACR_ij, PCR_ij)
ACR_ij ← min(cell.ER, ACR_ij)
ACR_ij ← Min(ACR_ij, CalculateER(i,j))
ACR_ij ← Max(ACR_ij, MCR_ij)
CCR_ij ← ACR_ij
if (cell.BN = 0) then
        if (time_to_send_ij > now() + 1/(ACR_ij)) then
                Reschedule: time_to_send_ij ← now() + 1/ACR_ij
Discard the BRM cell
```

Figure B.12: VS/VD: BRM cell received

```
TotalABRCapacity ← LinkCapacity - VBRCapacity
TargetABRCapacity ← g(q_i)×TotalABRCapacity
(Optional) TargetABRCapacity ← g(∑_j q_ij + q_i)×TotalABRCapacity
InputRate ← Measured ABRInputRate
OverLoadFctr ← InputRate/TargetABRCapacity

For each VC j on port i
        ER_ij^feedback ← g(q_ij) × ACR_ij
        FirstFRMinInterval_ij ← TRUE
        PrevIntvlMaxER_ij ← CurrIntvlMaxER_ij
        CurrIntvlMaxER_ij ← 0
```

Figure B.13: VS/VD: End of averaging interval

```
time_to_send_ij  ← 0
if (q_ij > 0 ∨ turn_around_ij) then
        if (ACR_ij < TCR_ij) then
                Send out of rate FRM
                Schedule: time_to_send_ij = now() + 1/TCR_ij
        else
                if ((count_ij ≥ Nrm) ∨ ((count_ij > Mrm) ∧ (now() ≥ last_RM_ij + Trm)))
then
                        time  ← now() - last_RM_ij
                        if (time > ADTF ∧ACR_ij > ICR_ij) then
                                ACR_in ← ICR_ij
                        if (unack_ij ≥ CRM_ij) then
                                ACR_ij ← ACR_ij − ACR_ij × CDF_ij
                                ACR_ij ← Max(ACR_ij, MCR_ij)
                        Send in rate FRM()
                        count_ij  ← 0
                        last_RM_ij  ← now()
                        first_turn_ij  ← TRUE
                        unack_ij = unack_ij + 1
                else if (turn_around_ij ∧ (first_turn_ij ∨ q_ij > 0)) then
                        CI_TA_ij ← CI_TA_ij ∨ CI_VC_ij
                        send in rate BRM
                        CI_VC_ij ← 0
                        turn_around_ij ←  FALSE
                        first_turn_ij ←  FALSE
                else
                        Send data cell

        count_ij ← count_ij + 1
        Schedule: time_to_send_ij ←  now() + 1/ACR_ij
```

Figure B.14: VS/VD: Time to send expires (now() ≥ time_to_send$_{ij}$)

APPENDIX C

Miscellaneous Tables and Results

Number of Sources	Z	R	K (Cells)	Efficiency	Fairness
5	0.8	0.9	1000	0.66	0.93
5	0.5	0.9	1000	0.80	0.99
5	0.2	0.9	1000	0.71	0.92
5	0.8	0.5	1000	0.21	0.45
5	0.5	0.5	1000	0.05	1.00
5	0.2	0.5	1000	0.33	0.78
5	0.8	0.1	1000	0.06	1.00
5	0.5	0.1	1000	0.04	1.00
5	0.2	0.1	1000	0.01	1.00
5	0.8	0.9	2000	0.84	0.98
5	0.5	0.9	2000	0.83	0.97
5	0.2	0.9	2000	0.89	0.97
5	0.8	0.5	2000	0.47	0.77
5	0.5	0.5	2000	0.58	0.97
5	0.2	0.5	2000	0.93	0.99
5	0.8	0.1	2000	0.20	1.00
5	0.5	0.1	2000	0.10	1.00
5	0.2	0.1	2000	0.04	1.00
5	0.8	0.9	3000	0.91	0.97
5	0.5	0.9	3000	0.88	0.96
5	0.2	0.9	3000	0.88	0.98
5	0.8	0.5	3000	0.92	0.99
5	0.5	0.5	3000	0.94	0.96
5	0.2	0.5	3000	0.94	0.90
5	0.8	0.1	3000	0.87	0.93
5	0.5	0.1	3000	0.20	1.00
5	0.2	0.1	3000	0.39	0.82

Table C.1: TCP over UBR+: Parameter analysis of Fair Buffer Allocation: 5 sources, LAN

Number of Sources	Z	R	K (Cells)	Efficiency	Fairness
15	0.8	0.9	1000	0.60	0.71
15	0.5	0.9	1000	0.68	0.77
15	0.2	0.9	1000	0.68	0.62
15	0.8	0.5	1000	0.28	0.34
15	0.5	0.5	1000	0.21	0.45
15	0.2	0.5	1000	0.40	0.61
15	0.8	0.1	1000	0.04	1.00
15	0.5	0.1	1000	0.06	0.20
15	0.2	0.1	1000	0.01	0.99
15	0.8	0.9	2000	0.85	0.96
15	0.5	0.9	2000	0.92	0.96
15	0.2	0.9	2000	0.87	0.96
15	0.8	0.5	2000	0.74	0.72
15	0.5	0.5	2000	0.73	0.63
15	0.2	0.5	2000	0.80	0.88
15	0.8	0.1	2000	0.11	1.00
15	0.5	0.1	2000	0.14	0.33
15	0.2	0.1	2000	0.20	0.29
15	0.8	0.9	3000	0.95	0.93
15	0.5	0.9	3000	0.94	0.96
15	0.2	0.9	3000	0.92	0.97
15	0.8	0.5	3000	0.43	0.74
15	0.5	0.5	3000	0.80	0.85
15	0.2	0.5	3000	0.85	0.90
15	0.8	0.1	3000	0.18	1.00
15	0.5	0.1	3000	0.11	1.00
15	0.2	0.1	3000	0.04	1.00

Table C.2: TCP over UBR+: Parameter analysis of Fair Buffer Allocation: 15 sources, LAN

Number of Sources	Z	R	K (Cells)	Efficiency	Fairness
5	0.8	0.9	12000	0.95	0.94
5	0.5	0.9	12000	0.95	0.98
5	0.2	0.9	12000	0.91	0.97
5	0.8	0.5	12000	0.84	0.99
5	0.5	0.5	12000	0.92	0.96
5	0.2	0.5	12000	0.89	0.96
5	0.8	0.1	12000	0.88	0.98
5	0.5	0.1	12000	0.88	0.95
5	0.2	0.1	12000	0.78	0.97
5	0.8	0.9	24000	0.92	1.00
5	0.5	0.9	24000	0.93	0.95
5	0.2	0.9	24000	0.93	1.00
5	0.8	0.5	24000	0.93	1.00
5	0.5	0.5	24000	0.93	1.00
5	0.2	0.5	24000	0.86	0.96
5	0.8	0.1	24000	0.93	0.98
5	0.5	0.1	24000	0.93	0.97
5	0.2	0.1	24000	0.85	0.99
5	0.8	0.9	36000	0.81	1.00
5	0.5	0.9	36000	0.81	1.00
5	0.2	0.9	36000	0.81	1.00
5	0.8	0.5	36000	0.81	1.00
5	0.5	0.5	36000	0.86	0.99
5	0.2	0.5	36000	0.93	1.00
5	0.8	0.1	36000	0.93	1.00
5	0.5	0.1	36000	0.89	0.98
5	0.2	0.1	36000	0.87	0.99

Table C.3: TCP over UBR+: Parameter analysis of Fair Buffer Allocation: 5 sources, WAN

Number of Sources	Z	R	K (Cells)	Efficiency	Fairness
15	0.8	0.9	12000	0.95	0.97
15	0.5	0.9	12000	0.94	0.99
15	0.2	0.9	12000	0.96	0.98
15	0.8	0.5	12000	0.95	0.99
15	0.5	0.5	12000	0.96	0.98
15	0.2	0.5	12000	0.96	0.98
15	0.8	0.1	12000	0.94	0.98
15	0.5	0.1	12000	0.91	0.99
15	0.2	0.1	12000	0.86	0.98
15	0.8	0.9	24000	0.96	0.98
15	0.5	0.9	24000	0.96	0.98
15	0.2	0.9	24000	0.96	0.98
15	0.8	0.5	24000	0.94	0.98
15	0.5	0.5	24000	0.94	0.97
15	0.2	0.5	24000	0.95	0.98
15	0.8	0.1	24000	0.93	0.99
15	0.5	0.1	24000	0.94	0.97
15	0.2	0.1	24000	0.96	0.99
15	0.8	0.9	36000	0.95	0.97
15	0.5	0.9	36000	0.96	0.97
15	0.2	0.9	36000	0.96	0.97
15	0.8	0.5	36000	0.96	0.99
15	0.5	0.5	36000	0.95	0.98
15	0.2	0.5	36000	0.96	0.97
15	0.8	0.1	36000	0.94	1.00
15	0.5	0.1	36000	0.94	0.95
15	0.2	0.1	36000	0.96	0.98

Table C.4: TCP over UBR+: Parameter analysis of Fair Buffer Allocation: 15 sources, WAN

Number of Sources	Configuration	Buffer Size (Cells)	Z	Efficiency	Fairness
5	LAN	1000	2	0.36	0.78
5	LAN	1000	1	0.13	0.81
5	LAN	1000	0.95	0.72	0.93
5	LAN	1000	0.9	0.65	0.96
5	LAN	1000	0.85	0.68	0.89
5	LAN	1000	0.8	0.75	0.98
5	LAN	1000	0.75	0.63	0.95
5	LAN	1000	0.5	0.57	0.95
5	LAN	1000	0.2	0.50	0.58
5	LAN	2000	1	0.47	0.92
5	LAN	2000	0.9	0.72	0.98
5	LAN	2000	0.8	0.84	0.95
5	LAN	3000	1	0.88	0.99
5	LAN	3000	0.9	0.89	0.98
5	LAN	3000	0.8	0.90	0.98
15	LAN	1000	1	0.38	0.48
15	LAN	1000	0.9	0.73	0.77
15	LAN	1000	0.8	0.75	0.76
15	LAN	2000	1	0.38	0.13
15	LAN	2000	0.9	0.91	0.95
15	LAN	2000	0.8	0.81	0.97
15	LAN	3000	1	0.93	0.94
15	LAN	3000	0.9	0.95	0.95
15	LAN	3000	0.8	0.94	0.94

Table C.5: TCP over UBR+: Parameter analysis of Selective Drop: LAN

Number of Sources	Configuration	Buffer Size (Cells)	Z	Efficiency	Fairness
5	WAN	12000	2	0.86	0.93
5	WAN	12000	1	0.91	0.96
5	WAN	12000	0.9	0.86	0.93
5	WAN	12000	0.8	0.90	0.95
5	WAN	12000	0.5	0.89	0.94
5	WAN	24000	1	0.92	0.97
5	WAN	24000	0.9	0.92	0.97
5	WAN	24000	0.8	0.91	0.98
5	WAN	36000	1	0.85	0.99
5	WAN	36000	0.9	0.80	0.99
5	WAN	36000	0.8	0.80	0.99
15	WAN	12000	1	0.93	0.97
15	WAN	12000	0.9	0.92	0.97
15	WAN	12000	0.8	0.93	0.90
15	WAN	24000	1	0.95	0.89
15	WAN	24000	0.9	0.94	0.92
15	WAN	24000	0.8	0.94	0.96
15	WAN	36000	1	0.94	0.97
15	WAN	36000	0.9	0.96	0.92
15	WAN	36000	0.8	0.96	0.88

Table C.6: TCP over UBR+: Parameter analysis of Selective Drop: WAN

Number of Sources	Buffer (cells)	TCP	GR	UBR	EPD	Selective Drop
5	1000	SACK	0.5	0.26	0.85	0.96
5	1000	SACK	0.1	0.98	0.57	0.75
5	1000	SACK	0.0	0.71	0.88	0.98
5	3000	SACK	0.5	0.96	0.97	0.95
5	3000	SACK	0.1	0.93	0.89	0.99
5	3000	SACK	0.0	0.83	0.91	0.92
5	1000	Reno	0.5	0.22	0.30	0.61
5	1000	Reno	0.1	0.37	0.41	0.66
5	1000	Reno	0.0	0.14	0.92	0.39
5	3000	Reno	0.5	0.60	0.69	0.76
5	3000	Reno	0.1	0.55	0.79	0.93
5	3000	Reno	0.0	0.59	0.72	0.92
5	1000	Vanilla	0.5	0.46	0.47	0.58
5	1000	Vanilla	0.1	0.40	0.58	0.70
5	1000	Vanilla	0.0	0.27	0.73	0.80
5	3000	Vanilla	0.5	0.88	0.72	0.87
5	3000	Vanilla	0.1	0.61	0.63	0.90
5	3000	Vanilla	0.0	0.61	0.88	0.85
15	1000	SACK	0.5	0.38	0.74	0.92
15	1000	SACK	0.1	0.49	0.76	0.91
15	1000	SACK	0.0	0.57	0.98	0.90
15	3000	SACK	0.5	0.90	0.96	0.92
15	3000	SACK	0.1	0.61	0.94	0.96
15	3000	SACK	0.0	0.43	0.86	0.95
15	1000	Reno	0.5	0.43	0.52	0.70
15	1000	Reno	0.1	0.35	0.48	0.68
15	1000	Reno	0.0	0.29	0.40	0.70
15	3000	Reno	0.5	0.68	0.88	0.95
15	3000	Reno	0.1	0.63	0.81	0.97
15	3000	Reno	0.0	0.54	0.69	0.89
15	1000	Vanilla	0.5	0.59	0.42	0.80
15	1000	Vanilla	0.1	0.38	0.52	0.70
15	1000	Vanilla	0.0	0.36	0.39	0.75
15	3000	Vanilla	0.5	0.68	0.90	0.97
15	3000	Vanilla	0.1	0.54	0.96	0.98
15	3000	Vanilla	0.0	0.37	0.85	0.89

Table C.7: Guaranteed Rate: TCP with VBR (300ms on/off): Efficiency for LAN

Number of Sources	Buffer (cells)	TCP	GR	UBR	EPD	Selective Drop
5	12000	SACK	0.5	0.95	0.93	0.94
5	12000	SACK	0.1	0.87	0.66	0.69
5	12000	SACK	0.0	0.42	0.43	0.61
5	36000	SACK	0.5	0.97	0.99	0.99
5	36000	SACK	0.1	0.96	0.98	0.96
5	36000	SACK	0.0	0.55	0.52	0.96
5	12000	Reno	0.5	0.93	0.96	0.94
5	12000	Reno	0.1	0.61	0.79	0.71
5	12000	Reno	0.0	0.34	0.45	0.33
5	36000	Reno	0.5	0.97	0.97	0.93
5	36000	Reno	0.1	0.90	0.96	0.75
5	36000	Reno	0.0	0.33	0.92	0.33
5	12000	Vanilla	0.5	0.94	0.97	0.96
5	12000	Vanilla	0.1	0.82	0.70	0.69
5	12000	Vanilla	0.0	0.49	0.36	0.42
5	36000	Vanilla	0.5	0.97	0.97	0.97
5	36000	Vanilla	0.1	0.96	0.90	0.94
5	36000	Vanilla	0.0	0.92	0.33	0.92
15	12000	SACK	0.5	0.88	0.85	0.90
15	12000	SACK	0.1	0.72	0.61	0.76
15	12000	SACK	0.0	0.64	0.48	0.58
15	36000	SACK	0.5	0.96	0.95	0.97
15	36000	SACK	0.1	0.95	0.94	0.97
15	36000	SACK	0.0	0.93	0.72	0.95
15	12000	Reno	0.5	0.97	0.94	0.97
15	12000	Reno	0.1	0.84	0.66	0.79
15	12000	Reno	0.0	0.67	0.53	0.51
15	36000	Reno	0.5	0.97	0.97	0.98
15	36000	Reno	0.1	0.96	0.96	0.97
15	36000	Reno	0.0	0.67	0.66	0.59
15	12000	Vanilla	0.5	0.90	0.92	0.96
15	12000	Vanilla	0.1	0.77	0.66	0.74
15	12000	Vanilla	0.0	0.67	0.61	0.67
15	36000	Vanilla	0.5	0.98	0.97	0.97
15	36000	Vanilla	0.1	0.96	0.96	0.97
15	36000	Vanilla	0.0	0.94	0.93	0.93

Table C.8: Guaranteed Rate: TCP with VBR (300ms on/off): Efficiency for WAN

Number of Sources	Buffer (cells)	TCP	GR	UBR	EPD	Selective Drop
5	1000	SACK	0.5	0.69	0.90	0.97
5	1000	SACK	0.1	0.21	0.81	0.91
5	1000	SACK	0.0	0.21	0.20	0.20
5	3000	SACK	0.5	0.79	0.97	0.94
5	3000	SACK	0.1	0.90	0.96	0.95
5	3000	SACK	0.0	0.95	0.99	0.99
5	1000	Reno	0.5	0.83	0.89	0.99
5	1000	Reno	0.1	0.60	0.87	0.88
5	1000	Reno	0.0	0.99	0.20	0.97
5	3000	Reno	0.5	0.98	0.81	1.00
5	3000	Reno	0.1	0.90	0.90	0.91
5	3000	Reno	0.0	0.92	0.89	0.98
5	1000	Vanilla	0.5	0.90	0.83	0.95
5	1000	Vanilla	0.1	0.74	0.36	0.93
5	1000	Vanilla	0.0	0.44	0.21	0.27
5	3000	Vanilla	0.5	0.48	0.88	0.96
5	3000	Vanilla	0.1	0.92	0.98	0.98
5	3000	Vanilla	0.0	0.98	0.96	0.98
15	1000	SACK	0.5	0.43	0.79	0.83
15	1000	SACK	0.1	0.49	0.57	0.84
15	1000	SACK	0.0	0.23	0.07	0.69
15	3000	SACK	0.5	0.83	0.91	0.98
15	3000	SACK	0.1	0.50	0.93	0.91
15	3000	SACK	0.0	0.65	0.70	0.96
15	1000	Reno	0.5	0.60	0.86	0.93
15	1000	Reno	0.1	0.55	0.78	0.69
15	1000	Reno	0.0	0.61	0.67	0.37
15	3000	Reno	0.5	0.87	0.96	0.98
15	3000	Reno	0.1	0.63	0.78	0.95
15	3000	Reno	0.0	0.72	0.77	0.94
15	1000	Vanilla	0.5	0.78	0.71	0.87
15	1000	Vanilla	0.1	0.26	0.34	0.71
15	1000	Vanilla	0.0	0.10	0.64	0.48
15	3000	Vanilla	0.5	0.87	0.91	0.96
15	3000	Vanilla	0.1	0.62	0.68	0.95
15	3000	Vanilla	0.0	0.82	0.72	0.88

Table C.9: Guaranteed Rate: TCP with VBR (300ms on/off): Fairness for LAN

Number of Sources	Buffer (cells)	TCP	GR	UBR	EPD	Selective Drop
5	12000	SACK	0.5	0.95	1.00	0.99
5	12000	SACK	0.1	0.75	0.92	0.99
5	12000	SACK	0.0	0.99	0.97	0.82
5	36000	SACK	0.5	0.95	0.86	0.89
5	36000	SACK	0.1	0.96	0.87	0.77
5	36000	SACK	0.0	0.88	0.97	0.63
5	12000	Reno	0.5	0.77	0.93	0.96
5	12000	Reno	0.1	0.84	0.94	0.79
5	12000	Reno	0.0	0.99	0.99	1.00
5	36000	Reno	0.5	0.87	1.00	0.97
5	36000	Reno	0.1	0.46	0.82	0.97
5	36000	Reno	0.0	1.00	0.71	1.00
5	12000	Vanilla	0.5	0.99	0.78	0.89
5	12000	Vanilla	0.1	0.78	0.87	0.76
5	12000	Vanilla	0.0	0.98	0.99	0.99
5	36000	Vanilla	0.5	1.00	0.78	0.98
5	36000	Vanilla	0.1	0.93	0.46	0.83
5	36000	Vanilla	0.0	0.75	1.00	0.73
15	12000	SACK	0.5	1.00	0.98	0.99
15	12000	SACK	0.1	0.96	0.97	0.96
15	12000	SACK	0.0	0.91	0.93	0.90
15	36000	SACK	0.5	0.92	0.98	0.96
15	36000	SACK	0.1	0.73	0.96	0.83
15	36000	SACK	0.0	0.74	0.95	0.84
15	12000	Reno	0.5	0.53	0.90	0.91
15	12000	Reno	0.1	0.91	0.95	0.83
15	12000	Reno	0.0	0.91	0.90	0.90
15	36000	Reno	0.5	0.90	0.79	0.96
15	36000	Reno	0.1	0.65	0.73	0.51
15	36000	Reno	0.0	0.89	0.92	0.92
15	12000	Vanilla	0.5	0.97	0.92	0.95
15	12000	Vanilla	0.1	0.89	0.94	0.94
15	12000	Vanilla	0.0	0.93	0.85	0.92
15	36000	Vanilla	0.5	0.89	0.88	0.92
15	36000	Vanilla	0.1	0.97	0.85	0.72
15	36000	Vanilla	0.0	0.83	0.77	0.88

Table C.10: Guaranteed Rate: TCP with VBR (300ms on/off): Fairness for WAN

Drop Policy	TCP	Buffer	GR	Efficiency	Fairness
Selective Drop	SACK	200000	0.5	0.87	0.91
Selective Drop	SACK	200000	0.1	0.78	0.82
Selective Drop	SACK	200000	0.0	0.74	0.87
Selective Drop	SACK	600000	0.5	0.99	1.00
Selective Drop	SACK	600000	0.1	0.99	0.99
Selective Drop	SACK	600000	0.0	0.99	1.00
Selective Drop	Reno	200000	0.5	0.33	0.71
Selective Drop	Reno	200000	0.1	0.24	0.93
Selective Drop	Reno	200000	0.0	0.16	1.00
Selective Drop	Reno	600000	0.5	0.35	0.99
Selective Drop	Reno	600000	0.1	0.39	0.99
Selective Drop	Reno	600000	0.0	0.30	0.98
Selective Drop	Vanilla	200000	0.5	0.83	0.90
Selective Drop	Vanilla	200000	0.1	0.71	0.99
Selective Drop	Vanilla	200000	0.0	0.81	0.87
Selective Drop	Vanilla	600000	0.5	0.79	1.00
Selective Drop	Vanilla	600000	0.1	0.80	0.99
Selective Drop	Vanilla	600000	0.0	0.76	1.00

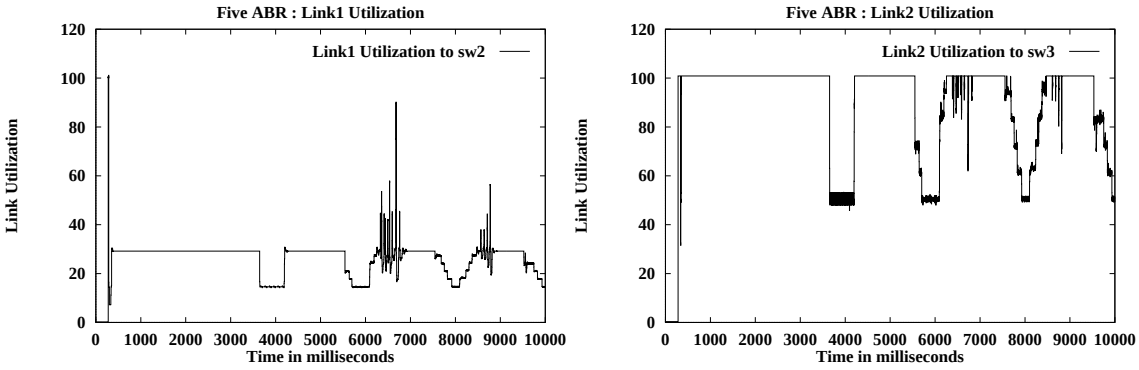
Table C.11: Guaranteed Rate: TCP with VBR (300ms on/off): Satellite

Drop Policy	TCP	Buffer	GR	Efficiency	Fairness
Early Packet Discard	SACK	200000	0.5	0.84	1.00
Early Packet Discard	SACK	200000	0.1	0.88	0.87
Early Packet Discard	SACK	200000	0.0	0.82	0.99
Early Packet Discard	SACK	600000	0.5	0.99	0.95
Early Packet Discard	SACK	600000	0.1	0.99	0.88
Early Packet Discard	SACK	600000	0.0	0.99	1.00
Early Packet Discard	Reno	200000	0.5	0.46	0.51
Early Packet Discard	Reno	200000	0.1	0.26	0.89
Early Packet Discard	Reno	200000	0.0	0.17	0.99
Early Packet Discard	Reno	600000	0.5	0.36	0.96
Early Packet Discard	Reno	600000	0.1	0.34	0.98
Early Packet Discard	Reno	600000	0.0	0.28	0.98
Early Packet Discard	Vanilla	200000	0.5	0.71	1.00
Early Packet Discard	Vanilla	200000	0.1	0.76	0.85
Early Packet Discard	Vanilla	200000	0.0	0.68	1.00
Early Packet Discard	Vanilla	600000	0.5	0.78	0.99
Early Packet Discard	Vanilla	600000	0.1	0.80	0.99
Early Packet Discard	Vanilla	600000	0.0	0.77	0.98

Table C.12: TCP with VBR (300ms on/off) over UBR+ with GR : Satellite

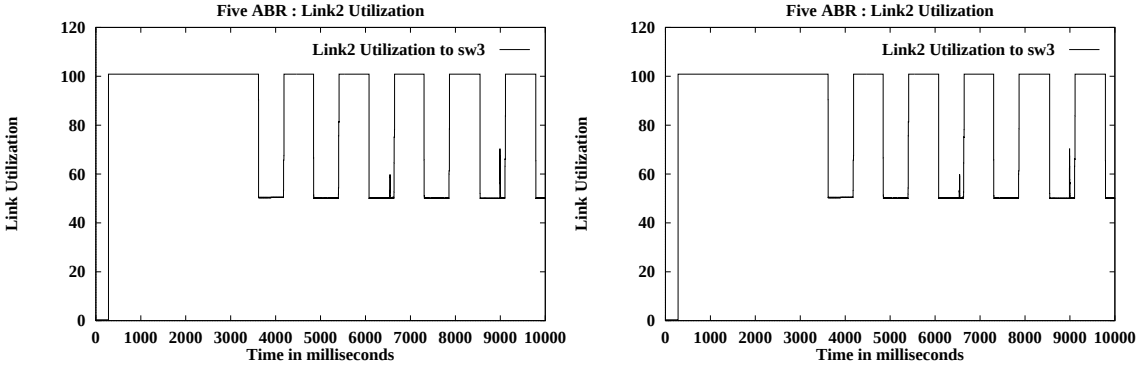
Drop Policy	TCP	Buffer	GR	Efficiency	Fairness
UBR	SACK	200000	0.5	0.87	0.91
UBR	SACK	200000	0.1	0.87	1.00
UBR	SACK	200000	0.0	0.85	1.00
UBR	SACK	600000	0.5	0.93	0.85
UBR	SACK	600000	0.1	0.96	0.87
UBR	SACK	600000	0.0	0.90	0.96
UBR	Reno	200000	0.5	0.87	0.88
UBR	Reno	200000	0.1	0.36	0.92
UBR	Reno	200000	0.0	0.38	0.9
UBR	Reno	600000	0.5	0.84	0.84
UBR	Reno	600000	0.1	0.69	0.77
UBR	Reno	600000	0.0	0.47	0.98
UBR	Vanilla	200000	0.5	0.87	0.84
UBR	Vanilla	200000	0.1	0.73	1.00
UBR	Vanilla	200000	0.0	0.84	0.86
UBR	Vanilla	600000	0.5	0.83	0.99
UBR	Vanilla	600000	0.1	0.83	0.99
UBR	Vanilla	600000	0.0	0.81	1.00

Table C.13: Guaranteed Rate: TCP with VBR (300ms on/off): Satellite



(a) VS/VD: Link 1 Utilization

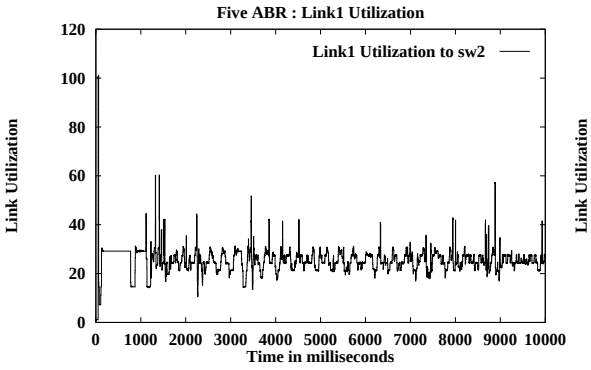
(b) VS/VD: Link 2 Utilization



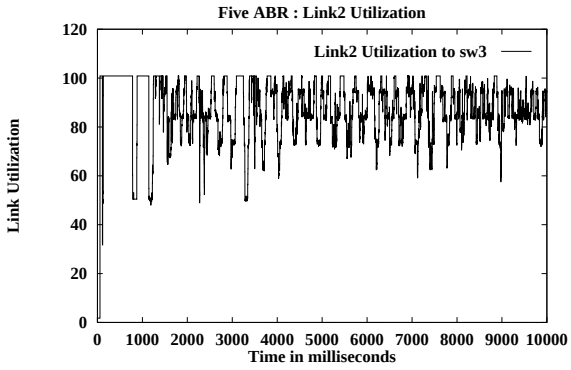
(c) Non-VS/VD: Link 1 Utilization

(d) Non-VS/VD: Link 2 Utilization

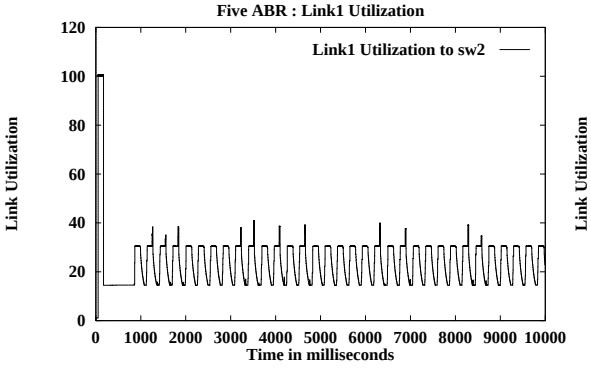
Figure C.1: Link Utilizations for VS/VD and non-VS/VD:GEO



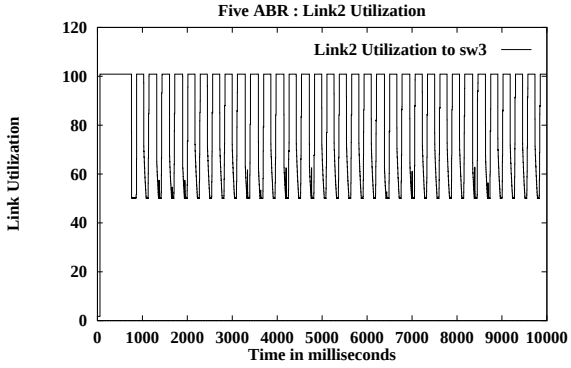
(a) VS/VD: Link 1 Utilization



(b) VS/VD: Link 2 Utilization

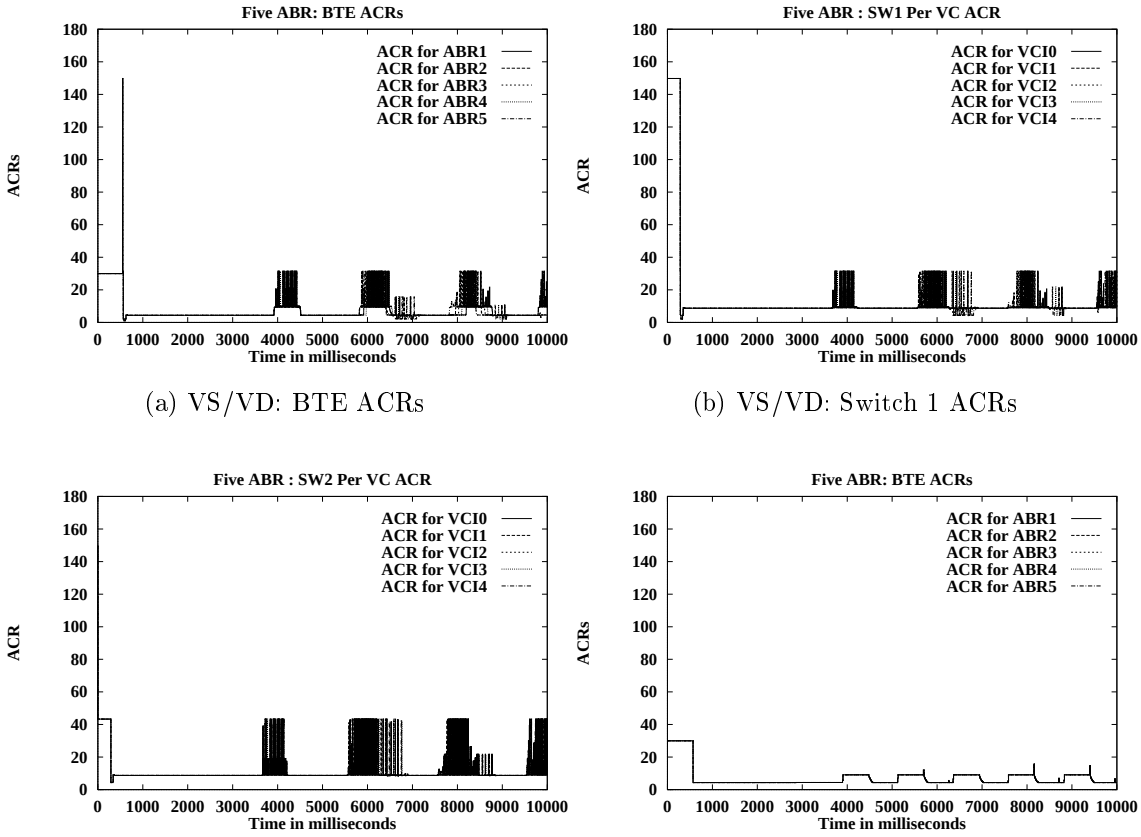


(c) Non-VS/VD: Link 1 Utilization



(d) Non-VS/VD: Link 2 Utilization

Figure C.2: Link Utilizations for VS/VD and non-VS/VD: LEO



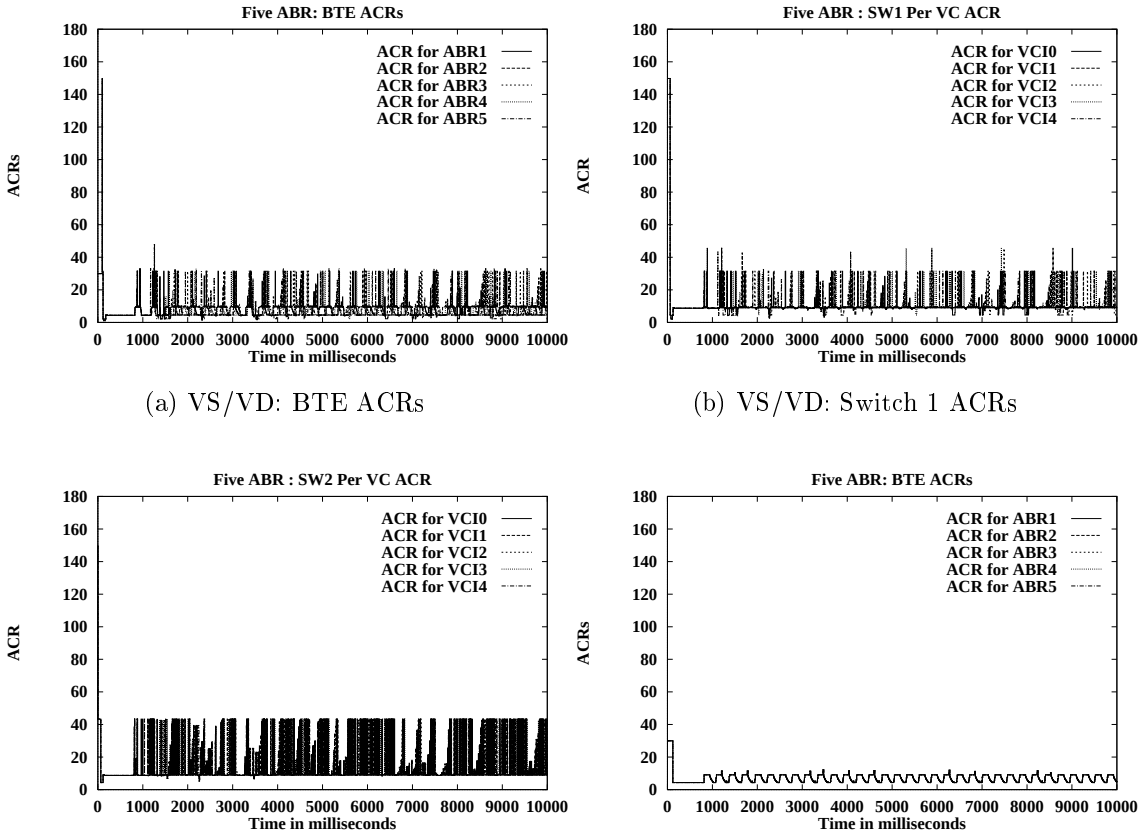
(a) VS/VD: BTE ACRs

(b) VS/VD: Switch 1 ACRs

(c) VS/VD: Switch 2 ACRs

(d) Non-VS/VD: BTE ACRs

Figure C.3: ACRs for VS/VD and non-VS/VD:GEO



(a) VS/VD: BTE ACRs

(b) VS/VD: Switch 1 ACRs

(c) VS/VD: Switch 2 ACRs

(d) Non-VS/VD: BTE ACRs

Figure C.4: ACRs for VS/VD and non-VS/VD Case:LEO

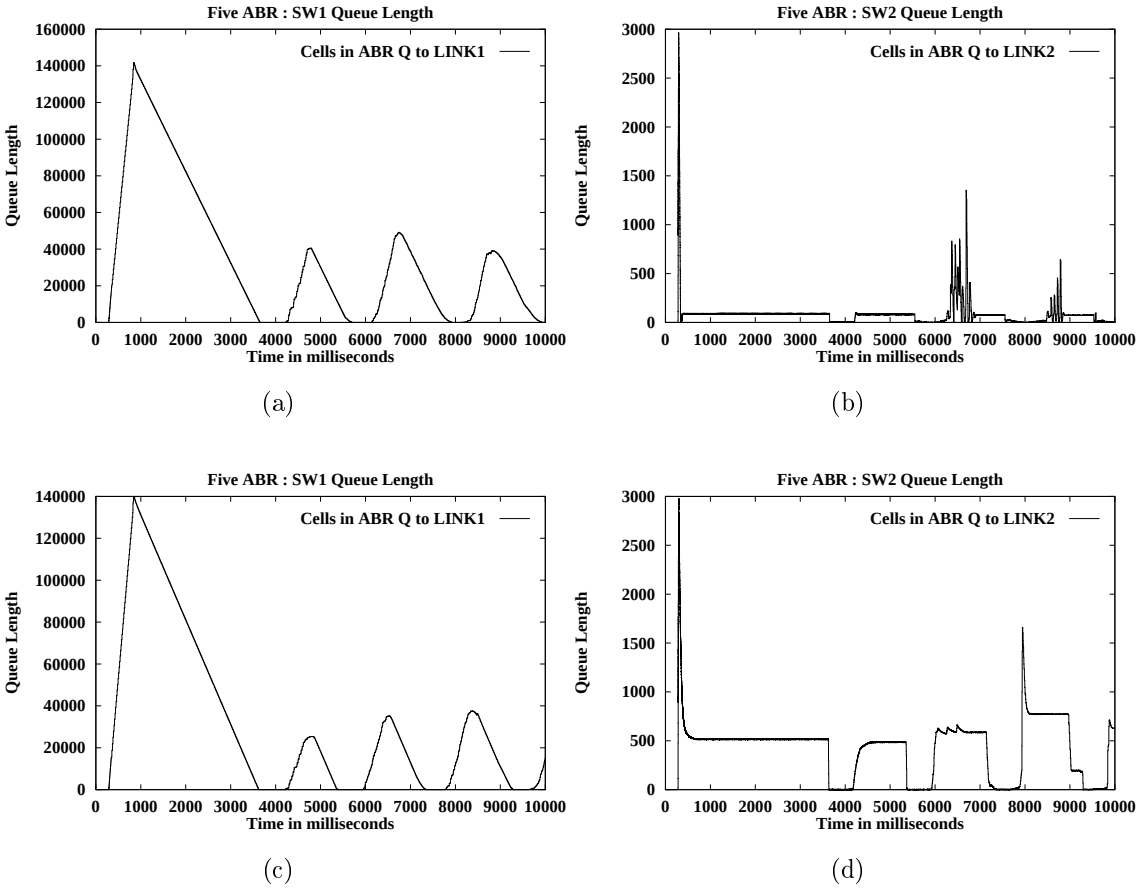
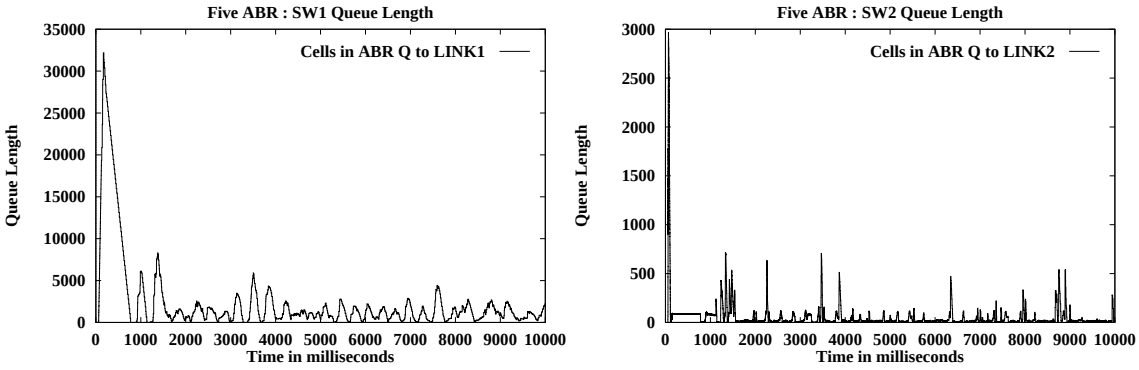
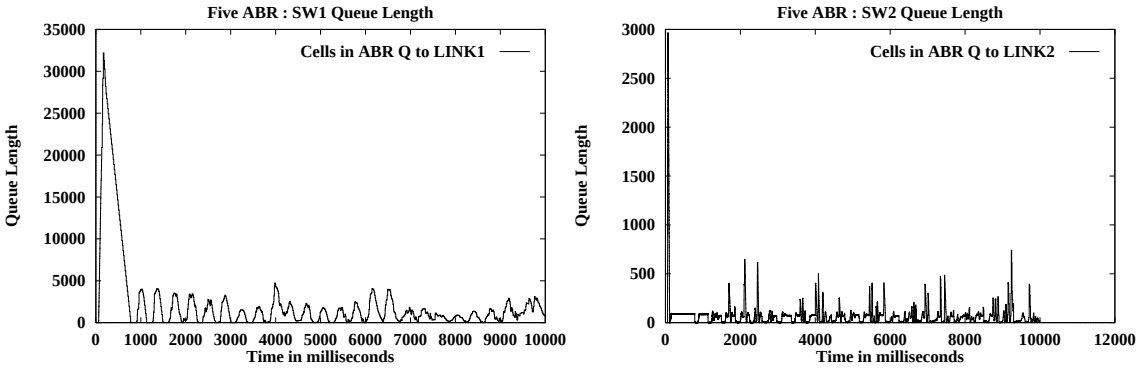


Figure C.5: Switch Queue Comparsion for Different t0v (config55)



(a) (b)



(c) (d)

Figure C.6: Switch Queue Comparsion for Different t0v (config10)

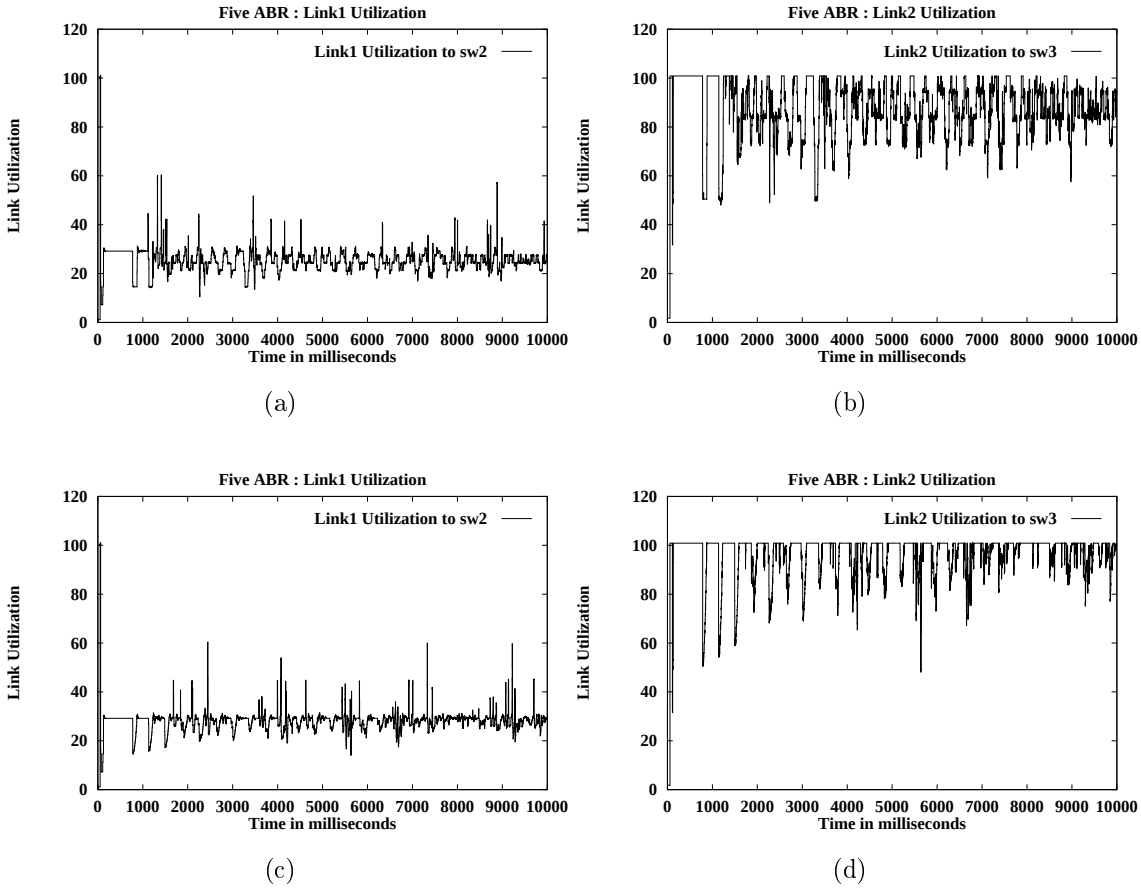
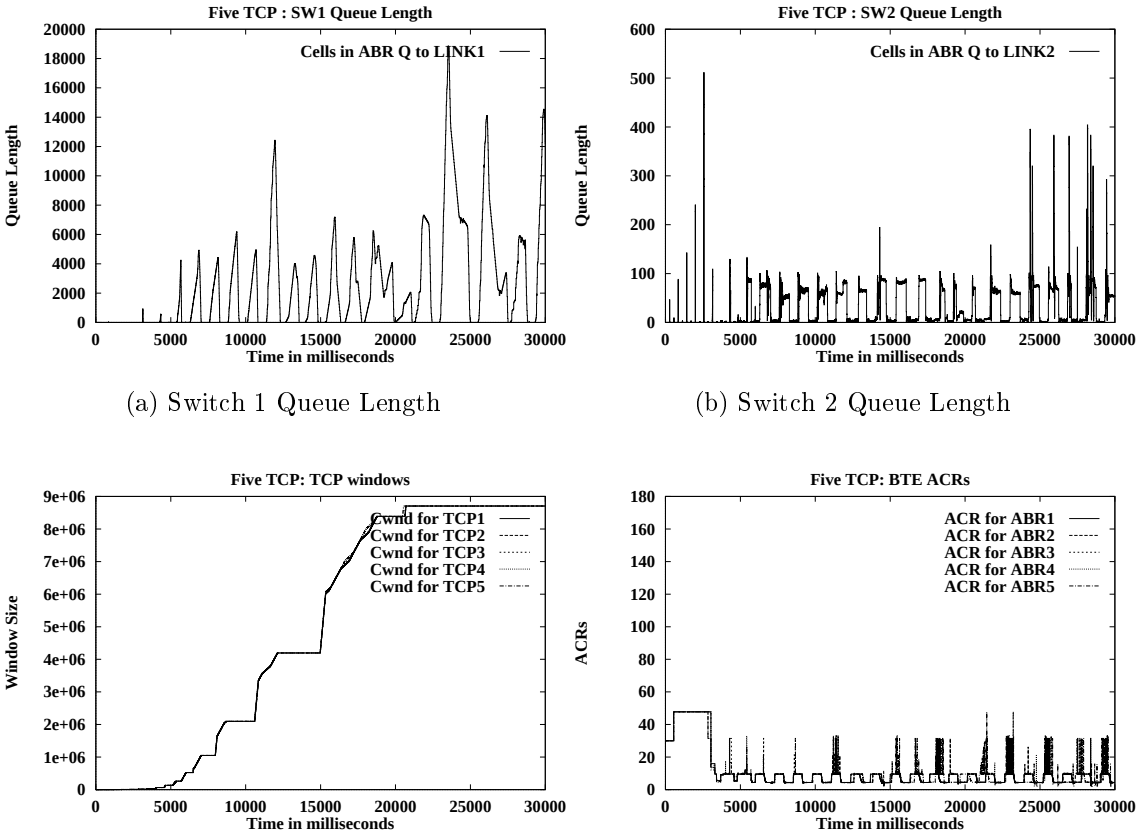


Figure C.7: Link Utilization Comparsion for Different t0v (config10)



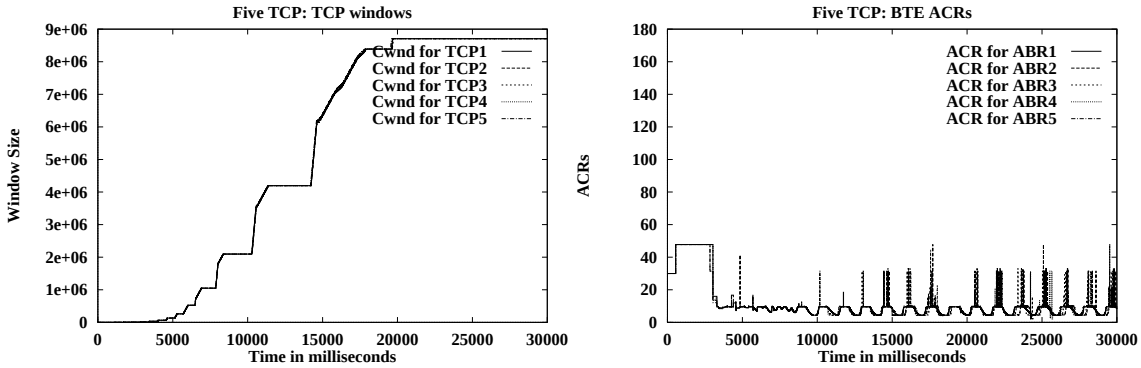
(a) Switch 1 Queue Length

(b) Switch 2 Queue Length

(c) TCP Cwnd

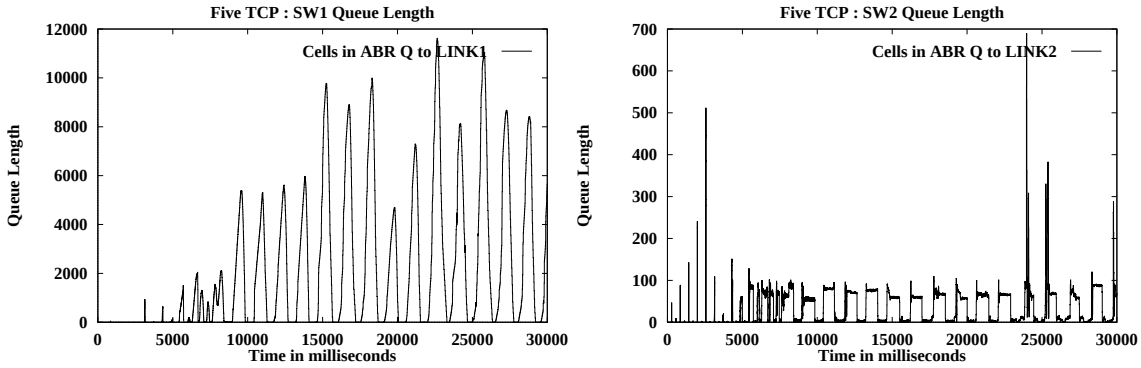
(d) BTE ACRs

Figure C.8: TCP Configuration with VS/VD



(a) Switch 1 Queue Length

(b) Switch 2 Queue Length



(c) TCP Cwnd

(d) BTE ACRs

Figure C.9: TCP Configuration for t0v = 5000

BIBLIOGRAPHY

[1] Yehuda Afek, Yishay Mansour, and Zvi Ostfeld. Phantom: A Simple and Effective Flow Control Scheme. In *Proceedings of the ACM SIGCOMM*, pages 169–182, August 1996.

[2] Ian F. Akyildiz and Seong-Ho Jeong. Satellite ATM Networks: A Survey. *IEEE Communications Magazine*, 5.35(7), July 1997.

[3] Mark Allman. On the generation and use of TCP acknowledgment. Submitted to ACM Computer Communications Review, July 1998.

[4] Mark Allman and editor. Ongoing TCP research related to satellites. Internet draft, work in progress, 1998.

[5] Mark Allman, Sally Floyd, and Craig Partridge. Increasing TCP's initial window. RFC 2414, September 1998.

[6] Mohit Aron and Peter Druschel. End-to-end TCP congestion control over ATM networks. TR97-273, Department of Computer Science, Rice University.

[7] Hari Balakrishnan. Challenges to reliable data transfer over heterogenous wireless links. PhD Dissertation, University of California Berkeley, 1998.

[8] Hari Balakrishnan, Venkata N. Padmanabhan, and Randy H. Katz. The effects of asymmetry on TCP performance. In *Proceedings of the ACM/IEEE Mobicom*, September 1997.

[9] Debashis Basak and Surya Pappu. ATM Forum 97-0528, 1997.

[10] Bellcore. Broadband switching system (BSS) generic requirements. GR-1110-CORE, September 1994.

[11] Dimitri Bertsekas and Robert Gallager. *Data Networks: Second Edition.* Prentice Hall, 1987.

[12] Azer Bestavros and Gitae Kim. TCP boston: A fragmentation-tolerant TCP protocol for ATM networks. Technical Report, Computer Science Department, Boston University.

[13] IETF BOF. Adaptive applications support BOF (adapts). 40th IETF, December 9, 1997, in Washington D.C.

[14] Olivier Bonaventure. A simulation study of TCP with the proposed GFR service category. In *DAGSTUHL Seminar 9725, High Performance Networks for Multimedia Applications*, June 1997.

[15] Olivier Bonaventure. Providing bandwidth guarantees to internetwork traffic in ATM networks. In *Proceedings of IEEE ATM'98*, 1998.

[16] George Box, William Hunter, and Stuart Hunter. *Statistics for Experimenters: An Introduction to Design Analysis, and Model Building.* John Wiley & Sons, 1978.

[17] Robert Braden. T/TCP – TCP extensions for transactions: Functional specification. RFC 1644, July 1994.

[18] R.T. Braden. Requirements for internet hosts - communication layers. RFC 1122, 1989.

[19] Lawrence S. Brakmo, Sean W. O'Malley, and Larry Peterson. TCP vegas: New techniques for congestion detection and avoidance. In *Proceedings of the ACM SIGCOMM*, 1994.

[20] Wu chang Feng, Dilip Kandlur, Debanjan Saha, and Kang G. Shin. Techniques for eliminating packet loss in congested TCP/IP networks. University of Michigan Technical Report.

[21] D. Chiu and R. Jain. Analysis of the Increase/Decrease Algorithms for Congestion Avoidance in Computer Networks. *Journal of Computer Networks and ISDN Systems*, 17(1):1–14, June 1989. http://www.cis.ohio-state.edu/~jain/papers/cong_av.htm.

[22] C. Cseh, R. Karabek, and P. Reichl. Segmentation of the Traffic Control Loop for Available Bit Rate-Service. In *Proceedings of the Fifth IFIP Workshop on Performance Modelling and Evaluation of ATM Networks*, February 1998.

[23] A. Demers, S. Keshav, and S. Shenker. Analysis and simulation of a fair queueing algorithm. *Internetworking: Research and Experience*, pages 3–26, 1990.

[24] Tim Dwight. Guidelines for the simulation of TCP/IP over ATM. ATM FORUM 95-0077r1, March 1995.

[25] Y.S. Lin et. al. Quasi-Pushout Cell Discarding. *IEEE Communications Letters*, September 1997.

[26] Kevin Fall and Sally Floyd. Simulation-based Comparisons of Tahoe, Reno, and SACK TCP. *Computer Communications Review*, July 1996.

[27] Chien Fang and Arthur Lin. On TCP Performance of UBR with EPD and UBR-EPD with a Fair Buffer Allocation Scheme. ATM Forum 95-1645, December 1995.

[28] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, August 1993.

[29] S. Floyd and V. Jacobson. Link-sharing and Resource Management Models for Packet Networks. *ACM/IEEE Transactions on Networking*, 3(4):365–386, August 1995.

[30] Sally Floyd. Sally floyd's home page. http://www-nrg.ee.lbl.gov/floyd/epd.html.

[31] Sally Floyd. Connections with Multiple Congested Gateways in Packet-Switched Networks, Part 1. *Computer Communications Review*, 21(5), October 1991.

[32] Sally Floyd. Issues of TCP with SACK. Lawrence Berkeley Labs Technical Report, December 1995.

[33] Sally Floyd and Van Jacobson. On Traffic Phase Effects in Packet-Switched Gateways. *Computer Communications Review*, 21(2), 1991.

[34] ATM Forum. The ATM Forum Traffic Management Specification Version 4.0. ftp://ftp.atmforum.com/pub/approved-specs/af-tm-0056.000.ps, April 1996.

[35] ATM Forum. The ATM Forum Traffic Management Specification Version 4.1. Straw Vote Document, February 1999.

[36] Rich Gobbi, Eduardo Elizondo, Anthony Modelfino, and Frank Gargione. Evolution of the Astrolink^{tm} system. In *Proceedings of the 3rd KaBand Utilization Conference*, September 1997.

[37] Mukul Goyal, Rohit Goyal, Raj Jain, Bobby Vandalore, Sonia Fahmy, Tom Von-Deak, Kul Bhasin, Norm Butts, , and Sastri Kota. Performance analysis of TCP enhancements for WWW traffic using UBR+ with limited buffers over satellite links. ATM Forum 98-0876, 1998. http://www.cis.ohio-state.edu/~jain.

[38] R. Goyal, R. Jain, Sonia Fahmy, and Bobby Vandalore. Simulation experiments with guaranteed frame rate for TCP/IP traffic. ATM Forum 97-0607, 1997. http://www.cis.ohio-state.edu/~jain/atmf/a97-0607.htm.

[39] R. Goyal, R. Jain, S. Kalyanaraman, and S. Fahmy. Further results on UBR+: Effect of fast retransmit and recovery. ATM Forum 96-176, December 1996. http://www.cis.ohio-state.edu/~jain/atmf/a96-1761.htm.

[40] R. Goyal, R. Jain, S. Kalyanaraman, and S. Fahmy. UBR+:improving performance of TCP over ATM-ubr service by fair buffer management. In *Proceedings of the IEEE International Communications Conference (ICC)*, 1997. http://www.cis.ohio-state.edu/~jain/papers/icc97.htm.

[41] Rohit Goyal, Xiangrong Cai, Raj Jain, Sonia Fahmy, and Bobby Vandalore. Per-vc rate allocation techniques for ABR feedback in VS/VD networks. ATM Forum 97/1086r1, February 1998. http://www.cis.ohio-state.edu/~jain/atmf/a97-1086r1.htm.

[42] Roch Guerin and Juha Heinanen. UBR+ enhancements. ATM Forum 0015, February 1997.

[43] Roch Guerin and Juha Heinanen. UBR+ service category definition. ATM Forum 96-1598, December 1997.

[44] E. Hashem. Analysis of random drop for gateway congestion control. Report LCS TR-465, Laboratory of Computer Science, MIT, 1989.

[45] Juha Heinanen and Kalevi Kilkki. A fair buffer allocation scheme. Unpublished manuscript.

[46] Tom Henderson. On improving the fairness of TCP congestion avoidance. In *Proceedings of the IEEE GLOBECOM*, 1998.

[47] Andrew Heybey and Niel Robertson. The network simulator, version 3.1. MIT CS Tech Report, 1994.

[48] Janey C. Hoe. Improving the start-up behavior of a congestion control scheme for TCP. In *Proceedings of the ACM SIGCOMM*, 1996.

[49] American National Standards Institute. Broadband ISDN - ATM layer functionality and specification. ANSI T1.627-1993, 1993.

[50] V. Jacobson. Congestion avoidance and control. In *Proceedings of the SIGCOMM*, pages 314–332, AUG 1988.

[51] V. Jacobson, R. Braden, and D. Borman. TCP extensions for high performance. RFC 1323, 1992.

[52] J. Jaffe. Bottleneck Flow Control. *IEEE Transactions on Communications*, COM-29(7):954–962, 1980.

[53] R. Jain. A timeout-based congestion control scheme for window flow-controlled networks. *IEEE Journal on Selected Areas in Communications*, SAC-4(7):1162–1167, October 1986. http://www.cis.ohio-state.edu/~jain/papers/control.htm.

[54] R. Jain. A delay-based approach for congestion avoidance in interconnected heterogeneous computer networks. *Computer Communications Review*, pages 56–71, 1989. http://www.cis.ohio-state.edu/~jain/papers/delay.htm.

[55] R. Jain. Congestion control and traffic management in ATM networks: Recent advances and a survey. *Computer Networks and ISDN Systems*, October 1996. http://www.cis.ohio-state.edu/~jain/papers/cnis.htm.

[56] R. Jain, K. K. Ramakrishnan, and D. M. Chiu. Congestion Avoidance in Computer Networks with a Connectionless Network Layer. Technical Report DEC-TR-506, Digital Equipment Corporation, August 1987. http://www.cis.ohio-state.edu/~jain/papers/cr5.htm.

[57] Raj Jain. *The Art of Computer Systems Performance Analysis*. John Wiley & Sons, 1991.

[58] Raj Jain. Myths about Congestion Management in High-speed Networks. *Internetworking: Research and Experience*, 3:101–113, 1992. http://www.cis.ohio-state.edu/~jain/papers/cong_myt.htm.

[59] Raj Jain, Shiv Kalyanaraman, R. Goyal, and S. Fahmy. Buffer requirements for TCP over abr. ATM Forum 96-0517, March 1996. http://www.cis.ohio-state.edu/~jain/atmf/af_abr22.htm.

[60] S. Kalyanaraman, R. Jain, S. Fahmy, and B. Vandalore. The ERICA switch algorithm for ABR traffic management in ATM networks. Under revision for Transaction on Networking.

[61] Shivkumar Kalyanaraman. Congestion control for the available bit rate (ABR) service in asynchronous transfer mode (ATM) networks. PhD Dissertation, The Ohio State University, 1997.

[62] Shivkumar Kalyanaraman, Raj Jain, Jianping Jiang, Rohit Goyal, Sonia Fahmy, and Seong-Cheol Kim. . *Computer Networks and ISDN Systems Journal*, 30(19,14):1811–1824, 1998. http://www.cis.ohio-state.edu/~jain/papers/vsvd.htm.

[63] F. Kamoun and L. Kleinrock. Analysis of Shared Finite Storage in a Computer Network Node Environment Under General Traffic Conditions. *IEEE Transactions on Communications*, COM-28(7), 80.

[64] Sastri Kota. Standardized profile for asynchronous transfer mode (ATM). MIL-STD-188-176 Coordination Draft, 1996.

[65] Sastri Kota, Rohit Goyal, and Raj Jain. Satellite-ATM network architectural considerations and TCP/IP performance. In *Proceedings of the 3rd Ka-Band Utilization Conference*, 1997. http://www.cis.ohio-state.edu/~jain/papers/kaband.htm.

[66] T.V. Lakshman. The drop from front strategy in TCP and TCP over ATM. In *Proceedings of the IEEE INFOCOM*, 1996.

[67] T.V. Lakshman and Upamanyu Madhow. The Performance of Networks with High Bandwidth-delay Products and Random Loss. *ACM/IEEE Transactions on Networking*, June 1997.

[68] M. Laubach and J. Halpern. Classical IP and ARP over ATM. RFC 2225, April 1998.

[69] Hongqing Li, Kai-Yeung Siu, and Hong-Ti Tzeng. TCP over ATM with abr service versus ubr+epd service. ATM FORUM 95-0718, June 1995.

[70] Hongqing Li, Kai-Yeung Siu, Hong-Ti Tzeng, Chinatsu Ikeda, and Hiroshi Suzuki. TCP over abr and ubr services in ATM. In *Proceedings of IPCCC'96*, March 1996.

[71] D. Lin and R. Morris. TCP fast recovery strategies. In *Proceedings of the IEEE INFOCOM*, 1998.

[72] Dong Lin and Robert Morris. Dynamics of random early detection. In *Proceedings of the ACM SIGCOMM*, 1997.

[73] J. Mahdavi and S. Floyd. TCP-friendly unicast rate-based flow control. Technical note sent to the end2end-interest mailing list, January 1997.

[74] M. Mathis, J. Madhavi, S. Floyd, and A. Romanow. TCP selective acknowledgment options. RFC 2018, 1996.

[75] M. Mathis, J. Semke, J. Mahdavi, and T. Ott. The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm. *Computer Communications Review*, 27(3), July 1997.

[76] Matthew Mathis and Jamshid Madhavi. Forward acknowledgment: Refining TCP congestion control. In *Proceedings of the ACM SIGCOMM*, 1996.

[77] NIST. NIST ATM/HFC network simulator. http://www.hsnt.hist.gov/misc/hsnt/prd_atm-sim.html.

[78] T. Ott, J.H.B. Kemperman, and M. Mathis. The stationary behavior of ideal TCP congestion avoidance. http://www.psc.edu/networking.

[79] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP throughput: a simple model and its empirical validation. UMASS CMPSCI Tech Report TR98-008, February 1998.

[80] Craig Partridge. *Gigabit Networking*. Addison-Wesley, Reading, MA, 1993.

[81] Craig Partridge. ACK spacing for high delay-bandwidth paths with insufficient buffering. Internet draft, work in progress, July 1997.

[82] J. Postel. Transmission control protocol. RFC 793, 1991.

[83] K. K. Ramakrishnan and Sally Floyd. A proposal to add explicit congestion notification (ECN) to IPv6 and to TCP. Internet draft, work in progress, July 1998.

[84] K.K. Ramakrishnan and Sally Floyd. A proposal to add explicit congestion notification (ECN) to IP. RFC 2481, January 1999.

[85] Reza Rejaie, Mark Handley, and Deborah Estrin. RAP: An end-to-end rate-based congestion control mechanism for realtime streams in the internet. In *Proceedings of the IEEE INFOCOM*, 1998.

[86] L. Roberts. Enhanced PRCA (proportional rate-control algorithm). ATM Forum 94-0735R1, August 1994.

[87] Allyn Romanow and Sally Floyd. Dynamics of TCP Traffic over ATM Networks. *IEEE Journal on Selected Areas in Communications*, 13(4), May 1995.

[88] Ramakrishna Satyavolu, Ketan Duvedi, and Shivkumar Kalyanaraman. Explicit rate control of TCP applications. ATM Forum 98/0152R1, February 1998.

[89] Kai-Yeung Siu and Hong-Yi Tzeng. Performance of TCP over ATM with time-varying available bandwidth. *Computer Communications*, 19:927–936, 1996.

[90] Kai-Yeung Siu, Yuan Wu, and Wenge Ren. Virtual queuing techniques for UBR+ service in ATM with fair access and minimum bandwidth guarantee. In *Proceedings of the IEEE GLOBECOM*, 1997.

[91] William Stallings. *Data and Computer Communications: Fourth Edition*. Macmillan Publishing Company, New Jersey, 1994.

[92] W. Stevens. TCP slow start, congestion avoidance, fast retransmit, and fast recovery algorithms. RFC 2001, 1997.

[93] W. Richard Stevens. *TCP/IP Illustrated, Volume 1.* Addison-Wesley, Reading, MA, 1995.

[94] Cisco Systems. Distributed weighted random early detection. http://www.cisco.com.

[95] W. Tan and A. Zakhor. Error resilient packet video for the internet. Submitted to ICIP 98.

[96] T. Turletti, Parisis, and J. Bolot. 'experiments with a layered transmission scheme over the internet. INRIA, Technical Report, 1998.

[97] International Telecommunications Union. B-ISDN asynchronous transfer mode functional characteristics. ITU-T Recommendation I.150, November 1995.

[98] Bobby Vandalore, Raj Jain, Rohit Goyal, and Sonia Fahmy. Design and analysis of queue control functions for explicit rate switch schemes. In *Proceedings of the IEEE International Conference on Computers and Communication Networks (ICCCN)*, pages 780–786, October 1998. http://www.cis.ohio-state.edu/~jain/papers/qctrl_bv.htm.

[99] Lorenzo Vicisano, Luigi Rizzo (Pisa), and Jon Crowcroft. TCP-like congestion control for layered multicast data transfer. In *Proceedings of the IEEE INFOCOM*, 1998.

[100] L. Wood. Satellite altitudes taken from lloyd's satellite constellation. http://www.ee.surrey.ac.uk/Personal/L.Wood/constellations/overview.html.

[101] Yuan Wu, Kai-Yeung Siu, and Wenge Ren. Improved virtual queuing and dynamic EPD techniques for TCP over ATM. In *Proceedings of the IEEE International Conference of Network Protocols (ICNP)*, 1997.

[102] Hui Zhang and Srinivasan Keshav. Comparison of rate-based service disciplines. In *Proceedings of the ACM SIGCOMM*, 1991.

ACRONYMS

AAL	ATM Adaptation Layer
ABR	Available Bit Rate
ACK	Acknowledgment
ACR	Allowed Cell Rate
ADTF	ACR Decrease Time Factor
ARP	Address Resolution Protocol
ARQ	Adaptive Repeat Request
ATM	Asynchronous Transfer Mode
B-ISDN	Broadband Integrated Services Digital Network
BECN	Backward Explicit Congestion Notification
BRM	Backward Resource Management
BT	Burst Tolerance
CAC	Connection Admission Control
CBR	Constant Bit Rate
CCR	Current Cell Rate
CDVT	Cell Delay Variation Tolerance
CDV	Cell Delay Variation
CI	Congestion Indication
CLP	Cell Loss Priority
CLR	Cell Loss Ratio
CRM	Missing RM Cell Count
CTD	Cell Transfer Delay
CWND	Congestion Window
DAMA	Demand Assignment Multiple Access
DELAY_ACK	Delayed Acknowledgment
DFBA	Differential Fair Buffer Allocation
DSCP	Differentiated Services Code Point
ECN	Explicit Congestion Notification
EFCI	Explicit Forward Congestion Indication
EOM	End of Message
EPD	Early Packet Discard
ERED	Enhanced Random Early Detection
ERICA+	Explicit Rate Incidation Congestion Avoidance Plus
ERICA	Explicit Rate Incidation Congestion Avoidance

ER	Explicit Rate
FACK	Forward Acknowledgment
FBA	Fair Buffer Allocation
FCFS	First Come First Serve
FIFO	First In First Out
FRED	Flow Random Early Detection
FRM	Forward Resource Management
FRR	Fast Retransmit and Recovery
FTP	File Transfer Protocol
GCRA	Generic Cell Rate Algorithm
GFR	Guaranteed Frame Rate
GR	Guaranteed Rate
HBO	High Buffer Occupancy
ICMP	Internet Control Message Protocol
IETF	Internet Engineering Task Force
IP	Internet Protocol
ISP	Internet Service Provider
ITU	International Telecommunications Union
LAN	Local Area Network
LBO	Low Buffer Occupancy
LLC	Logical Link Control
LQD	Longest Queue Drop
MA	Multiple Accounting
MBS	Maximum Burst Size
MCR	Minimum Cell Rate
MEO	Medium Earth Orbit
MSS	Maximum Segment Size
MTU	Maximum Transmission Unit
MT	Multiple Threshold
NCC	Network Control Center
NI	No Increase
PAWS	Protection Against Wrapped Sequences
PCR	Peak Cell Rate
PDU	Protocol Data Unit
PEP	Performance Enhancing Proxy
PLCP	Physical Layer Convergence Protocol
PME	Packet Marking Enging
PPD	Partial Packet Discard
PVC	Permanent Virtual Circuit
QoS	Quality of Service
RCVWND	Receiver Window
RDF	Rate Decrease Factor
RED	Random Early Detection

RFC	Request For Comments
RIF	Rate Increase Factor
RIO	RED with In and OUT
RM	Resource Management
RTTM	Round Trip Time Measurement
RTT	Round Trip Time
SACK	Selective Acknowledgments
SA	Single Accounting
SCR	Sustainable Cell Rate
SD	Selective Drop
SES	Source End System
SONET	Synchronous Optical Network
SSTHRESH	Slow Start Threshold
SSX	Sum of Squares of X
ST	Single Threshold
SVC	Switched Virtual Circuit
SYN	TCP Connection Establishment Message
TA	Turnaround
TCP	Transmission Control Protocol
TCR	Tagged Cell Rate
TOS	Type Of Service
UBR+	Unspecified Bit Rate Plus
UBR	Unspecified Bit Rate
UPC	Usage Parameter Control
VBR-nrt	Non Real-Time Variable Bit Rate
VBR-rt	Real-Time Variable Bit Rate
VC	Virtual Circuit
VQ	Virtual Queuing
VS/VD	Virtual Source/ Virtual Destination
Vegas-AFR	Vegas with Adaptive Fast Retransmit
WAN	Wide Area Network
WFBA	Weighted Fair Buffer Allocation
WFQ	Weighted Fair Queuing
WRED	Weighted Random Early Detection
WRR	Weighted Round Robin
WWW	World Wide Web