ON SUPPORTING QUALITY-OF-SERVICE OVER THE INTERNET

by

Hungkei (Keith) Chow

A thesis submitted in conformity with the requirements for the Degree of Doctor of Philosophy, Graduate Department of Electrical and Computer Engineering, University of Toronto

© Copyright by Hungkei (Keith) Chow 1999
The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.
On Supporting Quality-of-Service over the Internet

Doctor of Philosophy, 1999

Hungkei (Keith) Chow
Department of Electrical & Computer Engineering
University of Toronto

Abstract

The emergence of real-time applications has underscored the need to provide some forms of service quality in the Internet. Current discussions on quality of services (QoS), however, embrace different approaches and lack a unified definition. Until now, it is still questionable whether ubiquitous end-to-end QoS is possible in the Internet as its decentralised nature does not lend itself easily to implement homogeneous mechanisms for traffic differentiation.

This thesis presents methods of delivering QoS in the Internet. It focuses on two main rudiments of QoS recently proposed by the IETF community: Integrated Services (Int-Serv) and Differentiated Services (Diff-Serv). In addition, it addresses several important issues which arise from supporting these QoS models over the Internet.

For the Int-Serv model, the thesis proposes a fusion model that efficiently integrates the Int-Serv and RSVP into ATM. The architectural design, prototype implementation and experimental evaluation on the system performance are discussed. The results demonstrate an operational environment that efficiently facilitate the delivery of video over RTP/UDP/Int-Serv-IP/ATM. Furthermore, the thesis discusses a number of problems in supporting QoS given two unique characteristics of IP multicast: heterogeneity and any-point-to-any-point connectivity. Novel mechanisms, such as token forwarding and VC-merge capable scheduling, are developed. These mechanisms
complement the design of a common ATM switch in the support of both multicast and scalability. Performance analysis of these mechanisms is also presented.

For Diff-Serv, the thesis proposes a general feedback control extension to the current Diff-Serv framework. Based on this extension, an intra-domain flow control algorithm tailored for the network layer is developed. Our simulation analysis shows that the overall feedback controlled Diff-Serv can offer a significant improvement in resource sharing and utilisation. To justify the importance of this mechanism, the thesis also reviews the performance results and the trade-offs in terms of protocol overhead and complexity of the mechanism.

While this thesis has addressed some of the important issues and provided several solutions, delivering QoS in the Internet involves many aspects of network design. The results presented in this thesis will serve as an important step towards better understanding of issues related to the support QoS over the Internet.
To my love

Stephanie (麗雯) and Nathan (淳哲)
Acknowledgement

I would like to thank Prof. Alberto Leon-Garcia for his support during my study at UofT. He provided constant guidance, inspiration and valuable comments throughout this work. Alg is a superb teacher and an excellent project manager. I was fortunate to have him as my research advisor.

I also thank the member of my thesis committee, Prof. Angelos Bilas, Irene Katzela and Frank Kschischang, and the external examiner, Dr. Osama Aboul-Magd, for reviewing my thesis and for their feedback. Thanks are also due to Dr. Anindo Banerjea and Prof. Elvino Sousa for serving on my proposal committee and their valuable comments at the early stage of this work.

I wish to acknowledge in particular my alggrp-mates during my time at UofT, Kevin Gior, Massoud Hashemi, Wing-chung Hung, Andrew Jun, Sang Lee, Hassan Naser, Kazem Najafi, Walfrey Ng, Yasser Rasheed, Sanjeev Verma and Dr. Rouf Boutaba for their fruitful discussions and the fun time we had together. I especially acknowledge Richard Lee and Michael Yu for the delicious snacks and the great times that they shared with me during many sleepless nights at the lab.

Finally, my deepest thanks to my parents for their support and encouragement; and to the best of all, my wife Stephanie and my boy Nathan. Without their love, support and amazing patience, this work would not have been completed.

This thesis work was funded in part by research grant from CITR, Nortel Networks and National Sciences & Engineering Research Council of Canada (NSERC). I wish to acknowledge GN Nettest (Canada) Inc. for offering us access to their interWATCH 95000 ATM Analyzer; and Lawrence Berkeley Laboratory, Information Sciences Institute and Sun Microsystems for releasing their source code of ns-2, vic and rsvp.
Table of Contents

ABSTRACT .................................................................................................................... I

ACKNOWLEDGEMENT ............................................................................................... IV

TABLE OF CONTENTS ............................................................................................... V

LIST OF PUBLICATIONS ............................................................................................ XI

LIST OF FIGURES ..................................................................................................... XII

LIST OF TABLES ........................................................................................................ XV

LIST OF ACRONYMS .................................................................................................. XVI

CHAPTER 1 .................................................................................................................. 1

INTRODUCTION ......................................................................................................... 1

1.1 Motivations ............................................................................................................. 2
1.2 Aims and Scope ..................................................................................................... 4
1.3 Brief Summary of Contributions ...................................................................... 6
1.4 Thesis Organisation ............................................................................................. 7

CHAPTER 2 ................................................................................................................ 10

QoS Issues .................................................................................................................. 10

2.1 Specifying QoS Requirements ......................................................................... 11

2.1.1 Performance measures ................................................................................. 11
2.1.2 Policy measures .......................................................................................... 12
2.1.3 Service commitment .................................................................................... 13

2.2 Rudiments of QoS in the Internet ..................................................................... 15

2.2.1 Integrated Services ..................................................................................... 15
2.2.2 Differentiated Services .............................................................................. 17
3.8.1 Functional testing .......................................................... 50
3.8.2 Performance of signalling and control ......................... 51
3.8.3 Performance of data delivery ........................................ 53
3.9 SUMMARY ........................................................................ 56

CHAPTER 4 ........................................................................... 58

VC-MERGING SUPPORT ....................................................... 58

4.1 MOTIVATION ................................................................... 59
4.2 RELATED WORK ............................................................. 61
  4.2.1 Frame-level interleaving .............................................. 61
  4.2.2 Header extension ........................................................ 62
  4.2.3 Explicit notification ..................................................... 63
4.3 VC-MERGE CAPABLE SCHEDULING ............................... 65
  4.3.1 Concept of implicit notification ................................. 65
  4.3.2 Merge-capable scheduler ........................................... 65
  4.3.3 Operational details .................................................... 68
4.4 PERFORMANCE EVALUATION ......................................... 72
  4.4.1 Analytical analysis ....................................................... 72
    4.4.1.1 Worst case fairness ........................................... 72
    4.4.1.2 Worst case delay bound ..................................... 74
  4.4.2 Simulation analysis ..................................................... 75
    4.4.2.1 Fairness ............................................................. 77
    4.4.2.2 Average delay .................................................... 81
4.5 PRACTICAL ISSUES ......................................................... 85
  4.5.1 By-pass modes ........................................................... 85
  4.5.2 Signalling requirements .............................................. 88
  4.5.3 Pathological conditions ............................................. 90
  4.5.4 Implementation complexity ....................................... 92
4.6 SUMMARY ....................................................................... 93

CHAPTER 5 ........................................................................... 94
FEEDBACK CONTROLLED DIFFERENTIATED SERVICES

5.1 INTRODUCTION

5.2 MOTIVATION

5.2.1 Generalised forwarding model

5.2.2 Problems and suggested solution

5.3 PROPOSED FEEDBACK CONTROL EXTENSION TO DIFF-SERV

5.3.1 Architectural extension

5.3.1.1 FC-DS Domain

5.3.1.2 FC-DS Ingress node

5.3.1.3 FC-DS Interior node

5.3.1.4 FC-DS Egress node

5.3.2 Framework extension

5.3.2.1 Probing granularity

5.3.2.2 Handling control packet

5.3.2.3 Control Information

5.3.2.4 Adaptive traffic conditioning

5.3.2.5 Control algorithm

5.3.3 Other related considerations

5.3.3.1 Standardisation

5.3.3.2 Inter-domain control

5.3.3.3 Interoperability with non-FC-extended DS components

5.3.3.4 Multicast

5.3.3.5 Security

5.4 POSSIBLE INSTANCES AND APPLICATIONS

5.4.1 QoS monitoring for network management

5.4.2 Receiver control

5.5 SUMMARY

CHAPTER 6

AN INSTANCE OF FEEDBACK CONTROLLED DIFFERENTIATED SERVICES

6.1 RELATED WORK
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.1   SUMMARY OF CONTRIBUTIONS</td>
<td>162</td>
</tr>
<tr>
<td>7.2   FUTURE RESEARCH DIRECTIONS</td>
<td>164</td>
</tr>
<tr>
<td>7.3   CONCLUDING REMARK</td>
<td>165</td>
</tr>
<tr>
<td>APPENDICES</td>
<td>166</td>
</tr>
<tr>
<td>APPENDIX A</td>
<td>166</td>
</tr>
<tr>
<td>Proof of Theorem 4.1</td>
<td>166</td>
</tr>
<tr>
<td>Proof of Theorem 4.2</td>
<td>167</td>
</tr>
<tr>
<td>APPENDIX B</td>
<td>168</td>
</tr>
<tr>
<td>Proof of Lemma 6.1</td>
<td>168</td>
</tr>
<tr>
<td>Proof of Lemma 6.2</td>
<td>169</td>
</tr>
<tr>
<td>Proof of Lemma 6.3</td>
<td>170</td>
</tr>
<tr>
<td>Proof of Theorem 6.1</td>
<td>173</td>
</tr>
<tr>
<td>GLOSSARY</td>
<td>174</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>180</td>
</tr>
</tbody>
</table>
List of Publications


List of Figures

Figure 1.1.1 Generalized End-to-End control mechanisms ........................................... 2
Figure 1.4.1 Thesis organization ................................................................................. 9
Figure 2.1.1 Classification of Service Commitment ..................................................... 14
Figure 3.4.1 Overview of an ISAC environment ......................................................... 32
Figure 3.4.2 ISAC-C software architecture ................................................................. 34
Figure 3.4.3 ISAC-R controller software architecture ................................................... 35
Figure 3.5.1 A Typical RSVP reservation scenario ..................................................... 36
Figure 3.5.2 VCAP flow diagram ............................................................................... 37
Figure 3.5.3 An example of VCI allocation mechanism .............................................. 38
Figure 3.5.4 VC Assignment for heterogeneous reservation ......................................... 41
Figure 3.5.5 An example of heterogeneous multicast using piping concept ............... 42
Figure 3.5.6 Token forwarding design ........................................................................ 47
Figure 3.7.1 An experimental setup for performance evaluation ................................. 49
Figure 3.8.1 r-vic Graphical User Interface (GUI) ..................................................... 50
Figure 3.8.2 Event flow of various signalling sequences ............................................. 52
Figure 3.8.3 Experiment Set 1 (service type and level of utilization) ............................ 54
Figure 3.8.4 Experiment Set 2 (service type and number of hops) ............................... 55
Figure 4.2.1 Frame-level interleaving approach to VC-merging ................................. 62
Figure 4.2.2 Header extension approach to VC-merging ............................................. 63
Figure 4.2.3 Explicit notification approach to VC-merging ........................................ 64
Figure 4.3.1 Merge-capable scheduler ....................................................................... 66
Figure 4.3.2 An example of merging sequence ............................................................ 67
Figure 4.3.3 Operation of the output module of a merging node ................................. 69
Figure 4.3.4 Operation of the input module of a de-merging node ............................. 70
Figure 4.4.1 Simulation scenario .............................................................................. 76
Figure 4.4.2 Average goodput of different reference flows under two loading conditions ......................................................................................................................... 78
Figure 4.4.3 Effect on average goodput under different loading conditions ............... 79
Figure 4.4.4 Effect on average goodput under light loading conditions .................... 80
Figure 4.4.5  Average delay for different reference flows under two loading conditions 82
Figure 4.4.6  Effect on average delay under different loading conditions .................. 83
Figure 4.4.7  Effect of snooping ............................................................................... 84
Figure 4.5.1  Modified operation of input module of a de-merging node .............. 86
Figure 4.5.2  Modified operation of output module of a merging node .............. 87
Figure 4.5.3  Signalling requirement for receiver join/leave scenario .............. 89
Figure 4.5.4  Signalling requirement for sender join/leave scenario .................. 90
Figure 4.5.5  A simplified architecture of a common switch .................. 92
Figure 5.1.1  Diff-Serv Architecture model .......................................................... 96
Figure 5.1.2  Architecture of a Boundary node [68] ............................................ 97
Figure 5.1.3  Logical view of Traffic Conditioner [68] ........................................ 97
Figure 5.2.1  Generalised forwarding model ......................................................... 100
Figure 6.2.1  Probe generation operation ............................................................... 118
Figure 6.2.2  Control Packet Format (Data area only) ......................................... 120
Figure 6.2.3  Adaptive traffic conditioner with soft random discard ............... 122
Figure 6.3.1  NS2 implementation model of a FC-DS Interior node ................ 128
Figure 6.3.2  Topology 1 - Single bottleneck topology ..................................... 130
Figure 6.3.3  Topology 2 - Multiple bottleneck topology .................................. 132
Figure 6.3.4  Topology 3 - Multiple-tier topology ............................................. 133
Figure 6.3.5  Average achieved rate for Set # 1 .............................................. 136
Figure 6.3.6  Average achieved rate for Set # 2 .............................................. 138
Figure 6.3.7  Time response of average achieved rate for Set # 3 .................. 139
Figure 6.3.8  Average achieved rate for Set # 4 ............................................... 142
Figure 6.3.9  Delay distribution of microflow-2 within aggregate-0 & -1 for Set # 4 .............................................. 142
Figure 6.3.10 Delay distribution of microflow-2 within aggregate-2 to -8 for Set # 4 143
Figure 6.3.11 Average achieved rate for Set # 5 .............................................. 144
Figure 6.3.12 Delay distribution of microflow-2 within aggregate-0 & -1 for Set # 5 .............................................. 144
Figure 6.3.13 Delay distribution of microflow-2 within aggregate-2 to -8 for Set # 5 145
Figure 6.3.14 Average achieved rate for Set # 6 .............................................. 146
Figure 6.3.15 Delay distribution of microflow-2 within aggregate-0 & -1 for Set # 6 146
Figure 6.3.16 Average achieved rate for Set # 7 ............................................... 148
Figure 6.3.17 Average achieved rate for Set # 8 ......................................................... 149
Figure 6.3.18 Remarking rate of aggregated flows for Set # 7 ................................. 151
Figure 6.3.19 Remarking rate of aggregated flows for Set # 8 ................................. 151
Figure 6.3.20 Dynamics of the control algorithm ....................................................... 152
List of Tables

Table 2.1.1 Common performance measures............................................................. 12
Table 2.1.2 Examples of policy based requirements and features.......................... 13
Table 2.2.1 Summary of Int-Serv service proposals................................................... 16
Table 2.2.2 Service examples of Diff-Serv.................................................................. 18
Table 2.2.3 Characterisation of various service proposals .......................................... 18
Table 2.3.1 Summary of ATM Service Specification................................................ 22
Table 3.4.1 Functions for major software building blocks........................................... 33
Table 3.5.1 Parameter mapping matrix........................................................................ 39
Table 3.5.2 Summary of VC assignment models supporting multicast heterogeneity ... 41
Table 3.6.1 List of software modules......................................................................... 48
Table 3.8.1 Average signalling and control latency.................................................... 52
Table 4.3.1 Event decoding logic.................................................................................. 70
Table 4.4.1 Flow arrangement..................................................................................... 76
Table 4.6.1 Summary of various merging proposals.................................................. 93
Table 5.2.1 Summary of problems with Diff-Serv mechanism.................................... 101
Table 6.2.1 Summary of system configurations.......................................................... 127
Table 6.3.1 Common parameters for all simulation sets............................................. 129
Table 6.3.2 Simulation parameters for topology 1...................................................... 131
Table 6.3.3 Simulation parameters for topology 2..................................................... 132
Table 6.3.4 Simulation parameters for topology 3..................................................... 134
## List of acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(rt,nrt)VBR</td>
<td>(real-time, non-realtime) Variable Bit Rate</td>
</tr>
<tr>
<td>AAL</td>
<td>ATM Adaptation Layer</td>
</tr>
<tr>
<td>ABR</td>
<td>Available Bit Rate</td>
</tr>
<tr>
<td>ADSL</td>
<td>Asymmetric Digital Subscriber Line</td>
</tr>
<tr>
<td>AF</td>
<td>Assured Forwarding</td>
</tr>
<tr>
<td>ap2ap</td>
<td>anypoint-to-anypoint</td>
</tr>
<tr>
<td>APB</td>
<td>Available Path Bandwidth (Int-Serv GCP)</td>
</tr>
<tr>
<td>API</td>
<td>Application Interface</td>
</tr>
<tr>
<td>ARP</td>
<td>Address Resolution Protocol</td>
</tr>
<tr>
<td>AS</td>
<td>Assured Services</td>
</tr>
<tr>
<td>ATC</td>
<td>Adaptive Traffic Conditioning</td>
</tr>
<tr>
<td>ATM</td>
<td>Asynchronous Transfer Mode</td>
</tr>
<tr>
<td>ATMARP</td>
<td>Address Resolution Protocol for ATM</td>
</tr>
<tr>
<td>ATMF</td>
<td>ATM Forum</td>
</tr>
<tr>
<td>BA</td>
<td>Behavioral Aggregate</td>
</tr>
<tr>
<td>BAC</td>
<td>BW Aggregate Classification</td>
</tr>
<tr>
<td>BBE</td>
<td>Better than Best Effort service</td>
</tr>
<tr>
<td>BCOB</td>
<td>Broadband Connection-Oriented Bearer Capability</td>
</tr>
<tr>
<td>BCOB-{A,C,X}</td>
<td>Bearer Class A, C, or X</td>
</tr>
<tr>
<td>BE</td>
<td>Best Effort service</td>
</tr>
<tr>
<td>B-LLI</td>
<td>Broadband Low Layer Information</td>
</tr>
<tr>
<td>BT</td>
<td>Burst Tolerance</td>
</tr>
<tr>
<td>BUS</td>
<td>Broadcast and Unknown Server</td>
</tr>
<tr>
<td>B-WFI</td>
<td>Bit Worst case Fair Index</td>
</tr>
<tr>
<td>CAC</td>
<td>Call Admission Control</td>
</tr>
<tr>
<td>CBQ</td>
<td>Class-Based Queuing</td>
</tr>
<tr>
<td>CBR</td>
<td>Constant Bit-Rate</td>
</tr>
<tr>
<td>CBT</td>
<td>Core-Based Tree</td>
</tr>
<tr>
<td>CDV</td>
<td>Cell Delay Variation</td>
</tr>
<tr>
<td>CDVT</td>
<td>Cell Delay Variation Tolerance</td>
</tr>
<tr>
<td>CF-PHB</td>
<td>Control Forwarding PHB</td>
</tr>
<tr>
<td>CLIP</td>
<td>Classical IP over ATM</td>
</tr>
<tr>
<td>CLP</td>
<td>Cell Loss Priority</td>
</tr>
<tr>
<td>CLR</td>
<td>Cell Loss Ratio</td>
</tr>
<tr>
<td>CLS</td>
<td>Controlled-Load Service</td>
</tr>
<tr>
<td>CPCS</td>
<td>Common Part Convergence Sublayer</td>
</tr>
<tr>
<td>CRS</td>
<td>Committed-Rate Service</td>
</tr>
<tr>
<td>CS-SDU</td>
<td>Convergence Sublayer-Service Data Unit</td>
</tr>
<tr>
<td>CTD</td>
<td>Cell Transfer Delay</td>
</tr>
<tr>
<td>DCA</td>
<td>Domain Control Agreement</td>
</tr>
<tr>
<td>de-SEQ</td>
<td>de-sequencer</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Diff-Serv</td>
<td>Differentiated Services</td>
</tr>
<tr>
<td>DS</td>
<td>Differentiated Services</td>
</tr>
<tr>
<td>DSCP</td>
<td>DS Code-Point</td>
</tr>
<tr>
<td>EBS</td>
<td>Explicit Bucket Size</td>
</tr>
<tr>
<td>EF</td>
<td>Expedited Forwarding</td>
</tr>
<tr>
<td>EID</td>
<td>Egress node Identifier</td>
</tr>
<tr>
<td>ELAN</td>
<td>Emulated Local Area Network</td>
</tr>
<tr>
<td>EN</td>
<td>Explicit Notification</td>
</tr>
<tr>
<td>EOF</td>
<td>End-Of-Frame</td>
</tr>
<tr>
<td>EOM</td>
<td>End of Message</td>
</tr>
<tr>
<td>ER</td>
<td>Explicit Rate (profile)</td>
</tr>
<tr>
<td>FC-</td>
<td>Feedback Controlled-</td>
</tr>
<tr>
<td>FC-DS</td>
<td>Feedback Controlled Differentiated Services</td>
</tr>
<tr>
<td>FEC</td>
<td>Forward Error Correction</td>
</tr>
<tr>
<td>FI</td>
<td>Frame-Level interleaving</td>
</tr>
<tr>
<td>FIFO</td>
<td>First-In-First-Out</td>
</tr>
<tr>
<td>GCP</td>
<td>Generic Characterization Parameter</td>
</tr>
<tr>
<td>GCRA</td>
<td>Generic Cell Rate Algorithm</td>
</tr>
<tr>
<td>GS</td>
<td>Guaranteed Service</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>HE</td>
<td>Header Extension</td>
</tr>
<tr>
<td>ICMP</td>
<td>Internet Control Message Protocol</td>
</tr>
<tr>
<td>IE</td>
<td>Information Element</td>
</tr>
<tr>
<td>IETF</td>
<td>Internet Engineering Task Force</td>
</tr>
<tr>
<td>IGMP</td>
<td>Internet Group Management Protocol</td>
</tr>
<tr>
<td>IID</td>
<td>Ingress node Identifier</td>
</tr>
<tr>
<td>IN</td>
<td>Implicit Notification</td>
</tr>
<tr>
<td>Int-Serv</td>
<td>Integrated Services</td>
</tr>
<tr>
<td>ION</td>
<td>Internetworking Over NBMA</td>
</tr>
<tr>
<td>IP</td>
<td>Internet Protocol</td>
</tr>
<tr>
<td>IPI</td>
<td>Initial Protocol Identifier</td>
</tr>
<tr>
<td>IPng</td>
<td>IP next generation</td>
</tr>
<tr>
<td>IPv4</td>
<td>Internet Protocol version 4</td>
</tr>
<tr>
<td>IR</td>
<td>Ingress Rate (profile)</td>
</tr>
<tr>
<td>IS</td>
<td>Integrated Services</td>
</tr>
<tr>
<td>ISAC</td>
<td>Integrated Service and RSVP over ATM shortCuts</td>
</tr>
<tr>
<td>ISAC-C</td>
<td>ISAC Client device</td>
</tr>
<tr>
<td>ISAC-R</td>
<td>ISAC Router device</td>
</tr>
<tr>
<td>ISIP</td>
<td>Integrated Service Internet Protocol</td>
</tr>
<tr>
<td>ISSLL</td>
<td>Integrated Services over Specific Link Layers</td>
</tr>
<tr>
<td>ITU</td>
<td>International Telecommunication Union</td>
</tr>
<tr>
<td>IWF</td>
<td>Interworking Function</td>
</tr>
<tr>
<td>L-2</td>
<td>OSI Layer-2 (Link-layer)</td>
</tr>
<tr>
<td>L-3</td>
<td>OSI Layer-3 (Network-layer)</td>
</tr>
<tr>
<td>L-4</td>
<td>OSI Layer-4 (Transport-layer)</td>
</tr>
<tr>
<td>LANE</td>
<td>LAN Emulation</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>LIJ</td>
<td>Leaf-Initiated Join</td>
</tr>
<tr>
<td>LIS</td>
<td>Logical IP Subnetwork</td>
</tr>
<tr>
<td>LLC</td>
<td>Logical Link Control</td>
</tr>
<tr>
<td>LOF</td>
<td>Line Of Fairness</td>
</tr>
<tr>
<td>MAC</td>
<td>Microflow Classification</td>
</tr>
<tr>
<td>MARS</td>
<td>Multicast Address Resolution Server</td>
</tr>
<tr>
<td>MBS</td>
<td>Maximum Burst Size</td>
</tr>
<tr>
<td>MCR</td>
<td>Minimum Cell Rate</td>
</tr>
<tr>
<td>MF</td>
<td>Multi-Field</td>
</tr>
<tr>
<td>MI</td>
<td>Measurement Interval</td>
</tr>
<tr>
<td>MIB</td>
<td>Management Information Base</td>
</tr>
<tr>
<td>MPEG2 TS</td>
<td>MPEG2 Transport Stream</td>
</tr>
<tr>
<td>MPL</td>
<td>Minimum Path Latency</td>
</tr>
<tr>
<td>MPLS</td>
<td>Multi-Protocol Label Switching</td>
</tr>
<tr>
<td>MPOA</td>
<td>Multi-Protocol transport Over ATM</td>
</tr>
<tr>
<td>MSB</td>
<td>Most Significant Bit</td>
</tr>
<tr>
<td>MTU</td>
<td>Maximum Transmission Unit</td>
</tr>
<tr>
<td>NA</td>
<td>Not Available</td>
</tr>
<tr>
<td>NAL</td>
<td>Network Architecture Lab, University of Toronto</td>
</tr>
<tr>
<td>NBMA</td>
<td>Non-Broadcast Multi-Access</td>
</tr>
<tr>
<td>NC</td>
<td>Not Considered</td>
</tr>
<tr>
<td>NHRP</td>
<td>Next-Hop Resolution Protocol</td>
</tr>
<tr>
<td>NIC</td>
<td>Network Interface Card</td>
</tr>
<tr>
<td>N-ISDN</td>
<td>Narrow band-ISDN</td>
</tr>
<tr>
<td>OAM</td>
<td>Operation And Maintenance</td>
</tr>
<tr>
<td>OS</td>
<td>Olympic Services</td>
</tr>
<tr>
<td>P/R</td>
<td>Probe / Report Indication</td>
</tr>
<tr>
<td>PCR</td>
<td>Peak Cell Rate</td>
</tr>
<tr>
<td>PDS</td>
<td>Predictive Services</td>
</tr>
<tr>
<td>PDU</td>
<td>Protocol Data Unit</td>
</tr>
<tr>
<td>PHB</td>
<td>Per-Hop Behaviour</td>
</tr>
<tr>
<td>PIM-SM</td>
<td>Protocol Independent Multicast-Sparse Mode</td>
</tr>
<tr>
<td>PMM</td>
<td>Proportional Max-Min</td>
</tr>
<tr>
<td>PNNI</td>
<td>Private Network-Network Interface</td>
</tr>
<tr>
<td>POC</td>
<td>Point Of Convergence</td>
</tr>
<tr>
<td>POE</td>
<td>Plane Of Efficiency</td>
</tr>
<tr>
<td>POTTS</td>
<td>Plain Old Telephone Service</td>
</tr>
<tr>
<td>PQ</td>
<td>Priority Queuing</td>
</tr>
<tr>
<td>PS</td>
<td>Premium Services</td>
</tr>
<tr>
<td>PSB</td>
<td>PATH State Block</td>
</tr>
<tr>
<td>PVC</td>
<td>Permanent Virtual Connection</td>
</tr>
<tr>
<td>QoS</td>
<td>Quality of Service</td>
</tr>
<tr>
<td>RAPI</td>
<td>RSVP API</td>
</tr>
<tr>
<td>RBS</td>
<td>Request Bucket Size</td>
</tr>
<tr>
<td>RC</td>
<td>Referenced PHB-Class identifier</td>
</tr>
<tr>
<td>RED</td>
<td>Random Early Detection</td>
</tr>
</tbody>
</table>

xviii
RESV  Reservation Message (of RSVP protocol)
RFC  Request for Comments
RIO  RED with IN-OUT
RM cell  Resource Management cell
ROA  Region of Adaptation
RR  Requested Rate (profile)
RSB  RESV State Block
RSpec  Reservation Specification
RSRR  Routing Support for Resource Reservation
RSVP  Resource reSerVation setup Protocol
RSVPD  RSVP daemon
RTP  Real-time Transport Protocol
RTT  Round-Trip-Time
SBI  Service Burstiness Index
SBM  Subnet Bandwidth Manager
SCH  Scheduler
SCI  Switch Control Interface
SCR  Sustainable Cell Rate
SDU  Service Data Unit
SEQ  Sequencer
SID  Source ID
SLA  Service Level Agreement
SLS  Service Level Specification
SNAP  Subnetwork Attachment Point
SSCS  Service-Specific Convergence Sub-layer
SVC  Switched Virtual Connection
TC  Traffic Conditioner
TCA  Traffic Conditioning Agreement
TCI  Traffic Control Interface
TCP  Transmission Control Protocol
TM  Traffic Management
TOS  Type Of Service
TS  Time-stamp(sequence no)
TSpec  Traffic Specification
UBR  Unspecified Bit Rate
UDP  User Datagram Protocol
UNI  User Network Interface
UPC  Usage Parameter Control (ATM traffic policing function)
USD  User-Share Differentiated service
VC  Virtual Connection (ATM)
VCAP  VCI Allocation Protocol
VCM  VC Manager daemon
VLL  Virtual Lease Line service
VMSCH  VC-merge capable scheduler
VMWFQ  WFQ based VC-merge capable scheduler
VPN  Virtual Private Network

xix
<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VSB</td>
<td>VC State Block</td>
</tr>
<tr>
<td>WDM</td>
<td>Wavelength Division Multiplexing</td>
</tr>
<tr>
<td>WF</td>
<td>Wildcard Filter</td>
</tr>
<tr>
<td>WFQ</td>
<td>Weighted Fair Queuing</td>
</tr>
<tr>
<td>WG</td>
<td>Working Group (IETF)</td>
</tr>
<tr>
<td>WRR</td>
<td>Weighted Round Robin</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

"The fear of the LORD is the beginning of wisdom, and knowledge of the Holy One is understanding."

Proverbs 9:10 (NIV)

Computing and networking technologies have been advancing rapidly in recent years. Powerful workstations capable of generating, processing and presenting digital video and audio streams are available in the marketplace at a range of price/performance categories. These workstations, along with high capacity storage devices and real-time audiovisual codecs, have transformed our traditional text-based information into new hypermedia representation.

In the area of networking, new high-speed technologies such as ATM and WDM are being deployed in various commercial and research institutions. These networks connected by the long-haul high-speed optical links have constructed an infrastructure of information "superhighways". In the very near future, access technologies, such as cable modem, ADSL and wireless PCS, will provided convenient and cost-effective means to bring users to these high-speed networks. These global and high-speed networks have redefined our concept of geographical distance. Today, communications via the network can be instantaneous even though the participants might be far apart geographically.
These technological developments are well complemented by changing user perspectives. Even though today users are more readily to adopt new technologies, they also demand better services. As the cost of subscription continues to drop and new services are launched, network user population will continue to grow exponentially. This has created a global business environment in which many new and innovative products or services, such as distance learning, video-conferencing, virtual workshop/office, video-on-demand, tele-medicine, etc., will be developed and unrolled over the network. From the user perspective, in order to differentiate one product or service from the other, users need to know what functionality, features and reliability (or in a general term quality) the product or service can deliver.

1.1 Motivations

To deliver quality assurance to the end user is fundamentally an end-to-end issue - from service provider's application to consumer's application. For example, in considering a typical video on demand application, quality assurances should apply to the complete flow of media information - from the remote video server, across the network to the client set-top-box. As illustrated in Figure 1.1.1, this generally requires the provision of a sequence of control mechanisms. These include (1) a system level admission checking and resource reservation at first instance; (2) a careful co-ordination of memory, I/O devices and OS thread scheduling at the server end-system; (3) a network that is capable of delivering information according to a certain specification; and finally, (4) a similar set of control mechanisms at the client device(s).

![Figure 1.1.1 Generalized End-to-End control mechanisms](image-url)
This thesis focuses the discussion on the major component of this end-to-end control path - network control. The candidate network to study is the Internet mainly because of its widespread acceptability and accessibility. The Internet today is no longer a research project. For many organisations connected to the global Internet, it is a full-fledged business interest. The success of the Internet, as a result of the World Wide Web (WWW) application, has demonstrated the power of combining a user-friendly interface, simple networking protocols, and simple service creation environment. The main engine of the Internet is its network-layer protocol, namely Internet Protocol or IP. The salient features of IP are not only of its simplicity, but also of its inter-operability with various kinds of networking technologies. IP has been proven to work in relatively large-scale networks with a heterogeneous mix of protocols, physical media and devices.

Nevertheless, it has been noticed that the current version of IP (IPv4) has some limitations. For example, since the IP addressing scheme was proposed over two decades ago, it is definitely a concern that this addressing scheme cannot cope with the explosive growth of user population. Recently, the Internet Engineering Task Force (IETF) has started several working groups on defining a new version of IP (IP next generation or IPv6) [1]. IPv6 is intended to support Internet traffic and services for the foreseeable future by enhancing the capabilities of the existing IPv4. Various working groups are currently drafting specifications for the core functionality. The recommended IPv6 will cover a wide range of basic network functions such as addressing architecture, naming service, management information, routing support, security and mobility management.

Among these limitations, an important one is the service quality that the Internet offers. Historically, the Internet has offered a single form of service quality, (so called, best effort) where all data packets are treated with equity in the network. However, we notice that the Internet today does not offer such service quality. Some areas of the network exhibit high levels of congestion and consequently poor quality, while other parts show better quality consistently. Users are now voicing a requirement to define a consistent form of service quality and network service providers are also seeking ways to implement
such a requirement. Furthermore, the emerging real-time applications have underscored the need to provide some forms of service quality guaranteed over the Internet. As a matter of fact, the rapid growth of user population and increasing volume of traffic have made the Internet difficult for users to enjoy consistent and predictable end-to-end service quality. All of these efforts are happening within the umbrella of supporting end-to-end quality of service (QoS).

QoS is a phrase that has become overly used and often non-definitively referenced. Current discussions on QoS embrace abstract concepts and varying ideologies. It is still questionable whether ubiquitous end-to-end QoS is realistic in the Internet, given that the decentralised nature of the Internet does not lend itself easily to homogeneous mechanisms.

In summary, there are several motivations for studying this thesis topic of "Supporting QoS over the Internet". First, delivering end-to-end quality assurance to the end user under a distributed environment will unavoidably involve network-wise QoS. The control mechanism for the provision of network QoS plays an important role and needs to be studied. Second, the motivation for studying the Internet in particular is primarily due to its widespread acceptability and accessibility. To cope with the Internet development, the IETF has started several working groups on defining new service models. Currently, there are two main rudiments to bring QoS to the Internet: Integrated Services (Int-Serv) [2] and Differentiated Services (Diff-Serv) [3]. These issues are currently under active investigation in the industry and academia. These activities are also motives for studying this thesis work.

1.2 Aims and Scope

The aim of this thesis is to contribute towards the development of delivering QoS over the Internet. To achieve this aim, this thesis starts by discussing the notion of network-centric QoS. It reviews various network service specifications, and attempts to characterise existing service proposals for the Internet. In addition, it examines the available control mechanisms of delivering QoS requirements, and attempts to identify
their limitations. The focus of this thesis is to answer three fundamental questions related to the support of QoS over the Internet:

- How can the IETF service models be supported effectively?
- What additional mechanisms are required to facilitate such QoS provisioning?
- What extensions on the current proposed standard are required to achieve better QoS support?

More specifically, this thesis studies how the Int-Serv model can be effectively supported over ATM. Concerning the compatibility between Int-Serv IP and ATM, the thesis will look into three fundamental problems: (1) mapping IP flows onto ATM VCs; (2) integrating RSVP into ATM; and (3) integrating routing protocol into ATM. Through investigating these problems, the thesis aims at designing an efficient and effective Int-Serv-IP-over-ATM architecture.

Furthermore, the unique characteristics of IP multicast make QoS difficult to be supported. In RSVP, since each receiver can make reservation independently, a heterogeneous multicast session is resulted when receivers request different QoS's within a session. This contradicts the native ATM signalling in which a multicast VC has to be homogeneous. Besides its heterogeneity, IP differs from ATM multicast in its connectivity in a multicast session. Although ATM currently supports both point-to-point and point-to-multipoint VCs, an IP multicast session may accommodate any number of senders and receivers. In order to deliver QoS over this general IP multicast using ATM, a new set of control mechanisms are required in the ATM switch design. This thesis will study these mechanisms as required.

Apart from the Int-Serv model, the current development of Diff-Serv model is still in its infancy. It is yet unclear whether the proposed Diff-Serv architecture can eventually deliver the desirable QoS. With regard to this uncertainty, this thesis will investigate the performance of the existing proposal, and make some recommendations towards the development of a better Diff-Serv model.
1.3 Brief summary of contributions

The major contributions of this thesis are twofold: First, the architectural design that illustrates how different types of QoS models can be built using the available control mechanisms; and second, the proposals of new control mechanisms that complement the available ones for better support of the QoS models.

Two architectural frameworks are being proposed: the Integrated Services and RSVP over ATM shortcuts (ISAC) and the Feedback Controlled Differentiated Services (FC-DS). Based on the proposed fusion model, the ISAC architecture is developed to effectively integrate the Int-Serv IP and ATM technology. As part of this architecture, a simple VCI allocation protocol (VCAP) is developed. This protocol interfaces RSVP with ATM VC switching mechanism and effectively eliminates unnecessary protocol duplication. The major development in ISAC is its router controller, which converts an off-the-shelf ATM switch into an Int-Serv and RSVP enabled switch/router. Along with the ISAC architecture, mechanisms such as piping multicast, token forwarding, VC-merge capable scheduling are developed to support the unique characteristics of IP multicast, i.e., heterogeneity and anypoint-to-anypoint connectivity.

For the Diff-Serv, a general feedback control extension to the existing IETF proposal is developed. This Feedback Controlled Diff-Serv architecture can offer better controllability and fairness of resource sharing. To justify this recommendation, a flow control mechanism, which is tailored for the Diff-Serv network and derived from the general FC-DS framework, is developed.

All of these designs and proposals have been subjected to comprehensive verification and evaluation. Depending on feasibility, the assessment includes analytical analysis, system modelling and simulation-based analysis, as well as experimental evaluation that involves prototype implementation, practical application development and experimental measurement.
1.4 Thesis Organisation

Figure 1.4.1 illustrates the structure of the thesis. It is organised in a way that readers may follow any chapter route of their interest without losing contextual continuity. Outlines of each chapter are as follows:

Chapter 2 sets out the notion of network-centric QoS. It discusses the major components in specifying the QoS. It also classifies the existing service proposals for the Internet. Three major categories of QoS mechanisms are reviewed: *provisional, control* and *extrinsic mechanisms*. These mechanisms can be used to build these QoS models.

In Chapter 3, the framework of Int-Serv with RSVP over ATM is described. It primarily focuses on how Int-Serv can be supported effectively over the ATM network. This chapter first reviews different IP-over-ATM models as a comparison with the proposed one. Then it outlines the functional building blocks of the proposed framework. Next, the major functional operations and mechanisms are detailed. These include the mapping of parameters and signalling mechanism. Section 3.5.3, in particular, explores the issues of heterogeneous multicast. It presents a *piping multicast* concept together with a *Token Forwarding* mechanism to support the unique heterogeneous IP multicast. Most of the proposed framework and mechanisms have been implemented and detailed in Section 3.6. These include all system modules and application programs used for testing purposes. It is followed by a comprehensive experimental evaluation of the proposed system. Finally, the chapter is concluded by some discussions of the results and other related issues.

Chapter 4 addresses an issue, namely VC-merging, related to the development of ISAC. However, without loss of contextual continuity, it can also be considered independently because the concept derived there is generally applicable to other situations. VC-merging is known to be a powerful mechanism that allows networks to scale and support any-point-to-any-point connectivity. This chapter presents a VC-merge capable scheduling
concept and details the design of a VC-merge capable scheduler. Analytical and simulation analysis of its performance is discussed.

Chapter 5 switches the topic to the context of Diff-Serv. It begins by examining performance issues of the current IETF proposed standard. Then, it discusses the limitations and their causes of the Diff-Serv framework. To remedy these limitations, this chapter presents a general extension to the framework, namely Feedback Controlled Differentiated Services. Detailed specifications, operational requirements and other concerns of this extension are described.

Following this general specification of FC-DS is an instance of the framework presented in Chapter 6. It is a flow control mechanism that is used to resolve the fairness and efficiency of resource sharing. This chapter presents the detailed design of its components, operations and control algorithm. These system modules are implemented using the ns-2 simulator. The overall FC-DS network is subjected to comprehensive evaluation that focuses mainly on the performance, complexity and stability of the overall network.

Finally, Chapter 7 concludes this thesis with a summary of contributions, suggestions for future work and some general concluding remarks.
Figure 1.4.1 Thesis organization
Chapter 2

QoS Issues

Undoubtedly, in order to support QoS, we first need to understand what it is about. The objective of this chapter is therefore, to provide a background understanding of issues about QoS. In particular, it discusses a general overview on how QoS can be specified and what methods and mechanisms are currently available to deliver various forms of QoS over the Internet.

A network service specification generally includes a traffic specification and a QoS specification. The traffic specification describes either characteristics of user submitted traffic or service provider's expected traffic pattern. It is usually specified in terms of a set of token bucket parameters. Traffic specification forms part of the service contract between user and service provider. Once a service request is accepted, the service provider has agreed to provide a specific QoS for the user's traffic that conforms to the specification.

QoS specification generally describes either the user requested QoS requirements or the available service qualities that the service provider can offer. This chapter focuses on discussing two aspects of the QoS specification. First, it discusses how these QoS requirements or QoS offers are specified. As a summary, it reviews various QoS specifications embedded in the current IETF service models. In addition, the second part of this chapter discusses what tools/mechanisms are currently available for the service provider to deliver these QoS requirements or offers. The discussions are organised according to the nature of these mechanisms. Three primitive classes of mechanisms, provisional, control and extrinsic, are described in this section.
2.1 Specifying QoS requirements

QoS can be specified in terms of performance and management policy. It is usually declarative in nature: users specify what is required or service providers specify what is available rather than how this is to be achieved by underlying mechanisms. QoS requirements can generally be expressed by a combination of three primary components: performance, policy and commitment.

2.1.1 Performance measures

Performance measures are usually quantitative. They capture the network-imposed traffic characteristics that the service incurred. These measures are usually (but not always) specified in a per-flow basis. Table 2.1 lists some common measures and highlights their effects on the Internet traffic.

<table>
<thead>
<tr>
<th>Measures</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delay</td>
<td><em>The elapsed time for a packet to be passed from the sender, through the network, to the receiver:</em> It is usually specified in a statistical average and a maximum value. In general, the longer the delay, the greater the stress that is placed on the transport protocol to operate efficiently. For the TCP protocol, large network delay causes the feedback loop of TCP becomes insensitive to dynamic changes in the network load and therefore the protocol cannot function effectively. For interactive applications, delay also causes the system to appear unresponsive.</td>
</tr>
<tr>
<td>Jitter</td>
<td><em>The variation of end-to-end transit delay:</em> High level of jitter causes TCP to make conservative estimate of the round trip time (RTT) and results in poor efficiency. For UDP-based applications, large jitter is also unacceptable in situations where applications are real-time in nature, such as audio and video delivery. In those cases, large jitter causes the playout buffer at the receiver to suffer from overflow or underflow occasionally.</td>
</tr>
</tbody>
</table>
The maximum number of bytes that may be successfully transferred in unit time: It is the maximum data transfer rate that can be sustained between two end-points. It is not only limited by the physical transmission rate of the path, but it is also limited by the number of flows sharing the common segments of the end-to-end path.

The properties of the data transfer: It is usually specified in terms of the probability that a packet is transferred with error, or that a packet is lost, or that a duplicate copy of the packet is transferred. Unreliable transfer can cause retransmission of packets under TCP. It also invokes the TCP congestion avoidance mechanism and causes the sources to reduce their output rates even if no congestion occurs. For UDP-based applications, unreliable data transfer induces distortion of the original information or signal at the receiving end. This distortion is usually unrecoverable.

The packet arrival order at the receiving end: In most applications, it is desirable that packets are received in the same order as that of originally transmitted at the sender. Severe out-of-sequence packets incur the requirement of a large re-ordering buffer at the end system. Re-ordering also incurs additional processing delay. In most cases (both TCP and UDP traffic), severe out-of-sequence packets are regarded as transmission errors.

| Bandwidth/Rate | The maximum number of bytes that may be successfully transferred in unit time: It is the maximum data transfer rate that can be sustained between two end-points. It is not only limited by the physical transmission rate of the path, but it is also limited by the number of flows sharing the common segments of the end-to-end path. |
| Reliability | The properties of the data transfer: It is usually specified in terms of the probability that a packet is transferred with error, or that a packet is lost, or that a duplicate copy of the packet is transferred. Unreliable transfer can cause retransmission of packets under TCP. It also invokes the TCP congestion avoidance mechanism and causes the sources to reduce their output rates even if no congestion occurs. For UDP-based applications, unreliable data transfer induces distortion of the original information or signal at the receiving end. This distortion is usually unrecoverable. |
| Sequencing | The packet arrival order at the receiving end: In most applications, it is desirable that packets are received in the same order as that of originally transmitted at the sender. Severe out-of-sequence packets incur the requirement of a large re-ordering buffer at the end system. Re-ordering also incurs additional processing delay. In most cases (both TCP and UDP traffic), severe out-of-sequence packets are regarded as transmission errors. |

Table 2.1.1 Common performance measures

2.1.2 Policy measures

Policy measures describe features and high-level requirements that the service can offer. These descriptions can be quantitative (or absolute) and qualitative (or relative). These measures are usually in a coarser granularity than the performance measures, e.g., per-service basis and per-aggregate basis rather than per-flow basis. Several examples of policy measures are listed in Table 2.1.2.
<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aggregated bandwidth</td>
<td>It is a generalised notion of bandwidth for a group of traffic.</td>
</tr>
<tr>
<td>Resources share</td>
<td>It may specify a proportional fair sharing of resources. Resources can be in the form of bandwidth, buffer space or scheduling priority, etc.</td>
</tr>
<tr>
<td>Authorised flow characteristics</td>
<td>It may be specified as denial, admittance or priority of a certain set of flows with certain characteristics.</td>
</tr>
<tr>
<td>Resources usage/accounting</td>
<td>It may specify the total amount of resources consumption in a certain period of time.</td>
</tr>
<tr>
<td>Availability</td>
<td>Availability of the service may be specified with respects to the time-of-day.</td>
</tr>
<tr>
<td>Fault tolerance property</td>
<td>It may specify a guaranteed resumption of the service within a certain limit of time. It may also promise upper bounds on the total connectivity downtime over a period of time. Or it may even specify the amount of traffic which may be dropped due to congestion over a given period of time.</td>
</tr>
</tbody>
</table>

Table 2.1.2 Examples of policy based requirements and features

2.1.3 Service commitment

Service commitment refers to the "degree of promise" that the network will deliver according to the agreement specified using the above-mentioned measures. Service commitment can be specified quantitatively and qualitatively. A typical example of quantitative commitment is deterministic guarantee of a certain performance measure, e.g. a delay bound. The qualitative commitment is merely a certain form of relative assurance about how one set of packets or flows will be treated relative to the others.

From another perspective, service commitment can also be classified into inelastic and elastic type, depending on the way it is specified. Inelastic service commitment is the class of services that are stated in rigid terms, e.g., guarantee or certainty. In contrast,
elastic class is the group of services that have flexible commitments. This class of services usually covers a wider range of services.

Figure 2.1.1 classifies some common service commitments based on the attributes mentioned above. Definitions of these service commitments can be found in Refs. [6-11]
2.2 Rudiments of QoS in the Internet

Having described the three primary components of specifying the QoS, this section discusses various service proposals for the Internet. Currently, there are two main rudiments of service model proposed by the IETF community: Integrated Services (Int-Serv) and Differentiated Services (Diff-Serv).

2.2.1 Integrated Services

Int-Serv covers the definitions of services and the fundamental requirements on the network elements to support end-to-end QoS. In Int-Serv, each network element is QoS-aware and supports interfaces to a set of service definitions. It is required to work with a new reservation protocol called Resource ReSerVation Protocol (RSVP) [4, 5]. The concatenation of these individual services, which are supported at the network elements along an end-to-end path, forms an end-to-end QoS. In Int-Serv, a service is characterised in a per-flow basis. A flow in Int-Serv is a sequence of packets traversing a set of network elements, all of which are covered by the same request for control of QoS [6].

A service definition in Int-Serv is required to specify the invocation information. This describes the set of parameters required by the control module to invoke the service, and a description on how the parameters are used by the control module. These parameters include the traffic specification (TSpec) and the desired service specification (RSpec). Moreover, some services may also specify the export information, which is collected and exported by the control module. Exported information may be available to other modules of the network element, such as setup protocols, routing protocols, network management tools. Last but not least, a service definition must also include an end-to-end behaviour: This is a description of the behaviour that results if all network elements along the path offer the same type of service which is invoked using the same set of parameters.

Several service proposals with different end-to-end behaviours had been submitted to the IETF Int-Serv WG. They are summarised in Table 2.2.1. Among these proposals, the
IETF WG selects only Controlled-Load Service (CLS) and Guaranteed Service (GS) for standard track procedure.

CLS provides the client data flow with a quality of service closely approximating the QoS that same flow would receive from an unloaded network element, but uses capacity (admission) control to assure that this service is received even when the network element is overloaded [7]. On the other hand, GS provides firm bounds on end-to-end datagram queuing delays. It offers a network service that guarantees both delay and bandwidth [9].

<table>
<thead>
<tr>
<th>Services</th>
<th>Invocation</th>
<th>Export Info.</th>
<th>End-to-End Behaviour</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Controlled-Load (CLS) [7]</td>
<td>TSpec: ((m; M; p; r; b))</td>
<td>Not required</td>
<td>• Provide best effort service under &quot;unloaded&quot; condition&lt;br&gt;• Should prevent excess traffic from unfair resource share&lt;br&gt;• Must attempt to forward excess traffic on best effort basis</td>
<td>No target QoS parameters&lt;br&gt;No reshaping&lt;br&gt;Arrival traffic can be non-conforming</td>
</tr>
<tr>
<td>Committed-rate (CRS) [8]</td>
<td>TSpec: ((m; M; p; r; b)) RSpec: ((R))</td>
<td>Not required</td>
<td>• Must ensure packet with no or minimal queuing losses&lt;br&gt;• No delay guarantee&lt;br&gt;• Non-conformance packets are sent as best effort&lt;br&gt;• Should try to provide contracted QoS to conformance packets</td>
<td>Policed at edge of the network&lt;br&gt;Reshaped at intermediate nodes</td>
</tr>
<tr>
<td>Guaranteed (GS) [9]</td>
<td>TSpec: ((m; M; p; r; b)) RSpec: ((R; S)) where (r \leq R) (0 \leq S)</td>
<td>AdSpec: ((C; D))</td>
<td>• Delay must not exceed fluid flow model by more than specified error bounds&lt;br&gt;• Must provide end-to-end delay bound (D_e)&lt;br&gt;• Not control min delay but max queuing delay&lt;br&gt;• No jitter bound&lt;br&gt;• Non-conformance traffic should be sent as best effort</td>
<td>Policed at edge of the network&lt;br&gt;Reshaped at intermediate nodes&lt;br&gt;Link Must count datagrams which are smaller than (m)&lt;br&gt;Non-conformance if number of datagrams at a time period (T) more than (M + \min{pT, rT + b - M})</td>
</tr>
<tr>
<td>Predictive (PDS) [10]</td>
<td>TSpec: ((m; M; p; r; b)) RSpec: ((R))</td>
<td>Not Required</td>
<td>• Pre-established target delay bound (D_p)</td>
<td>Admission based on measured characteristics&lt;br&gt;Policing applies</td>
</tr>
</tbody>
</table>

\[
C \equiv \text{rate-dependent error [B]; i.e. delay due to rate error} \\
D \equiv \text{rate-independent, per-element error in delay [ms]} \\
R \equiv \text{desired rate [B/s]} \\
p \equiv \text{peak rate [B/s]} \\
M \equiv \text{max. packet size [B]} \\
S \equiv \text{slack term} \\
m \equiv \text{min. policed unit [B]} \\
r \equiv \text{bucket rate [B/s]} \\
b \equiv \text{bucket depth [B]} \\
\begin{align*}
D_e &= \frac{(b - M - p - r)}{R} + \frac{M + C}{R} \\
D_p &= \frac{M}{R} + D_{\text{tot}}
\end{align*}
\]

Table 2.2.1 Summary of Int-Serv service proposals
2.2.2 Differentiated Services

The IETF Diff-Serv WG adopts a quite different methodology from Int-Serv. As of the date of writing, Diff-Serv model is still under development. Its ultimate goal is to provide a wide range of differentiated service classes according to different business requirements. These services can be used by different types of the Internet traffic which are generated by various types of applications. To achieve this goal, Diff-Serv attempts to define a set of mechanisms instead of service specifications, together with the architecture and framework that enable service providers to construct their own services. Moreover, differing from Int-Serv, Diff-Serv provides scalable service discrimination without the need for per-flow state and signalling information at every network node. It is unnecessary to perform a unique resource reservation for each flow. Therefore, flows from various applications are naturally aggregated into a few classes of forwarding mechanisms.

Service in Diff-Serv is defined as the Service Level Agreement (SLA) [11]. The SLA is a service contract between a customer and service provider that specifies the details of the traffic classification and the corresponding forwarding service the customer's traffic should receive. Diff-Serv architecture [3] proposes that an end-to-end service can be constructed by concatenation of per-domain services which the customer's traffic will cross. These services are associated with customer-provider SLAs. Table 2.2.2 lists some service examples. These examples are intended to illustrate a wide range of services that can be derived using the Diff-Serv model, and by no means, be an exhaustive list.

<table>
<thead>
<tr>
<th>Services</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Better than Best-Effort (BBE) [11]</td>
<td>This is a qualitative service, which promises to carry specific type of traffic at a higher priority than the competing best-effort traffic.</td>
</tr>
<tr>
<td>Virtual Lease Line or Premium (VLL) [12]</td>
<td>It promises to deliver customer traffic with very low latency and very low drop probability up to a negotiated rate.</td>
</tr>
</tbody>
</table>
Assured (AS) [13,14] | It promises to deliver traffic with a high degree of assurance and with variable but bounded latency, up to a negotiated rate. Above this rate, traffic is subject to significant delay or drop.

"Olympic" (OS) [15] | It consists of three service classes: bronze, silver, and gold. Packets are assigned to these three classes so that packets in the gold class experience lighter load (and thus have higher probability for timely delivery) than packets assigned to the silver class. Similar relationship exists between the silver class and the bronze class.

User-Share Differentiated (USD) [16] | It ensures that the bandwidth allocated to a flow is commensurate with some pre-defined metrics, such as the price paid by the originator to the service provider.

| Table 2.2.2 Service examples of Diff-Serv |

2.2.3 Summary

To summarise our discussions in the last two sections, Table 2.2.3 characterises all mentioned service proposals using the framework discussed in Section 2.1.

<table>
<thead>
<tr>
<th>QoS Model</th>
<th>Service</th>
<th>Performance Metric</th>
<th>Policy Metric</th>
<th>Service Commitment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Int-Serv</td>
<td>GS</td>
<td>Delay, rate</td>
<td></td>
<td>Elastic</td>
</tr>
<tr>
<td></td>
<td>PS</td>
<td>Delay</td>
<td></td>
<td>In-elastic</td>
</tr>
<tr>
<td></td>
<td>CRS</td>
<td>Rate</td>
<td></td>
<td>Qualitative</td>
</tr>
<tr>
<td></td>
<td>CLS</td>
<td>Unloaded</td>
<td></td>
<td>Quantitative</td>
</tr>
<tr>
<td>Diff-Serv</td>
<td>VLL</td>
<td>Rate</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AS</td>
<td>Rate, delay</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>USD</td>
<td>Rate</td>
<td>Proportional fair-share</td>
<td></td>
</tr>
<tr>
<td></td>
<td>OS</td>
<td>Levels of loading</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>BBE</td>
<td>Priority to type of traffic</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Legacy</td>
<td>BE</td>
<td>Equity</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| Table 2.2.3 Characterisation of various service proposals |
2.3 QoS mechanisms

In previous Sections, we have discussed how QoS requirements are specified. To realise these QoS requirements, we need to have mechanisms that can provide differentiation of traffic in the network. In other words, QoS mechanisms can be regarded as a set of tools that help build the QoS models. According to user or provider specified QoS requirements and management policy, network providers select and configure a set of mechanisms to manipulate and control their available resources. In the following, we discuss three major categories of QoS mechanisms: provisional, control and extrinsic.

2.3.1 Provisional mechanisms

Provisional mechanisms deal with service establishment, support and negotiation. They include:

- **Network planning or engineering** - it concerns network design issues, e.g., network topology and link capacity. To cope with the growth of network usage, a network needs to be reviewed regularly and sometimes reconfigured. It is because the best solution for networks that are almost always congested is to install additional high capacity links or redesign the topology to match the demand pattern.

- **QoS Interpretation** - it performs function of translating the QoS requirements into another set of parameter formats that is meaningful to the underlying control mechanisms.

- **Admission control** - it is responsible for comparing the resource requirement of the requested service against the available resources in the network. The decision whether a new request is admitted may also depend on some management policies. Once admission check has been successfully completed, local resources are reserved immediately and all resource modules along the flow path are committed if the request is end-to-end.

- **Reservation protocols** - the major functions of this mechanism are: (1) to obtain a set of QoS parameters from QoS interpretation; (2) to interact with the routing module to establish a path through the network; and (3) to carry the parameters to initiate an admission check at each node along the end-to-end path. The end result of a
reservation is that sufficient end-to-end resources for delivering the requested QoS are reserved at each participating network nodes.

2.3.2 Control mechanisms

While differing from provisional mechanisms in time-scales and granularity, control mechanisms usually operate at or close to the data transfer speeds. They provide traffic control over data flows according to requested levels of QoS established using the provisional mechanisms. Similar to provisional mechanisms, control mechanisms can be combined or can interact with each other to become a more powerful mechanism. Several primitive control mechanisms are listed below:

- **Dropping** is the simplest form of control mechanism. It discards packets under certain conditions and according to some criteria. For example, when the number of packets inside a buffer exceeds a certain threshold, the dropping mechanism will start discarding non-conforming packets. Dropping may also perform under some precedence or priority rules. Common dropping schemes include drop-tail (drop from the tail of a buffer), drop-front (drop at the head of the queue) and random drop (randomly select packets to drop). Depending on the nature of a flow, dropping may be unfavourable because the discarded data can hardly be recovered.

- **Flow shaping** regulates flows based on the performance specifications. It involves delaying packets until a desired specification is met. Shaping can be done at the input or output of a network node depending on the service contract.

- **Flow scheduling** manages the forwarding of flows based on a specific rule (or scheduling policy). It has been proven that the combination of traffic shaping and scheduling can provide deterministic performance guarantees (quantitative, inelastic service commitment). Parekh [17] has shown that if a source flow is shaped by a token bucket with leaky bucket rate control and scheduled by the weighted fair queuing service discipline [18], it is possible to achieve strong guarantees on delay.

- **Flow policing** observes whether the traffic specification, contracted by a customer is being honoured. Policing is often appropriate where administrative and charging boundaries are being crossed. The action taken by the policing mechanism can range from accepting violations with caution (tagging or marking), shaping incoming traffic
to an acceptable specification, to discarding all packets that are out of the contracted profile.

- **Flow control or congestion control** includes both open-loop and closed loop schemes. Open loop flow control allows user to inject data into the network at the agreed levels, given that resources have been committed in advance. In contrast, closed loop flow control requires the user to adjust the rate based on feedback from the receiver or network. In closed loop flow control, either the source or the network access point has to be able to rapidly adapt to fluctuations in the available resources.

- **Flow re-sequencing and synchronisation** controls the packet ordering and precise timing of departure or arrival. In most cases, it is desirable to receive packets in the same order as they are originally sent. During the course of delivery, packets may be re-ordered for whatever reasons. Re-sequencing mechanism is therefore required at the intermediate node or often at the receiving end to re-order packets. Apart from re-sequencing within a flow, flow synchronisation concerns re-ordering and delaying packets among flows in order to meet certain requirements. Flow synchronisation is often used for real-time traffic flows.

- **Flow classification** is a mechanism that re-organises incoming flows or packets according to certain policy specification. It can be considered as both segregation and aggregation of incoming traffic. A common goal of performing classification is to treat certain traffic differently from the other.

- **Flow measurement and monitoring** can be viewed as a superset of policing. It observes traffic against both performance and policy specification. Its algorithms usually operate over different time-scales and granularity than flow policing. For example, they can run as part of the flow control mechanism to measure the loading of the network. In some cases, measured statistics can also be used to control the scheduling, shaping or dropping mechanism at the network entrance.

### 2.3.3 Extrinsic mechanisms

Since IP is merely a network layer protocol, it is logical to examine what QoS mechanisms are available at the lower layer of the OSI protocol stack, particularly data-link layer. These link-layer mechanisms can potentially be used to the support of QoS at
the network-layer. Our discussion will focus on what QoS they can offer and how their QoS models are achieved.

2.3.3.1 ATM QoS

ATM is one of the few transmission technologies, which provides high-speed data-transport, a complex matrix of traffic management services, Virtual Connection (VC) establishment controls and various QoS parameters for these VC's. Table 2.3.1 summarizes ATM services specification [19].

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Attribute</th>
<th>CBR</th>
<th>rtVBR</th>
<th>nrtVBR</th>
<th>ABR</th>
<th>UBR</th>
</tr>
</thead>
<tbody>
<tr>
<td>QoS</td>
<td>CLR</td>
<td>Specified</td>
<td>Specified</td>
<td>Specified</td>
<td>Specified</td>
<td>Unspecified</td>
</tr>
<tr>
<td></td>
<td>CTD &amp; CDV</td>
<td>Specified</td>
<td>Specified</td>
<td>Specified</td>
<td>Specified</td>
<td>Unspecified</td>
</tr>
<tr>
<td>Traffic</td>
<td>PCR &amp; CDTV*</td>
<td>Specified</td>
<td>Specified</td>
<td>Specified</td>
<td>Specified</td>
<td>Specified</td>
</tr>
<tr>
<td></td>
<td>SCR &amp; BT*</td>
<td>NA</td>
<td>Specified</td>
<td>Specified</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td></td>
<td>MCR</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>Specified</td>
<td>NA</td>
</tr>
<tr>
<td>Flow Control</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

* These parameters are either explicitly or implicitly specified for PVCs or SVCs as defined in Sec. 3.6.2.4.1 of the UNI 3.x/4.0 specifications

Table 2.3.1 Summary of ATM Service Specification

Apart from its rich service specification, ATM is known to be one of the most complex protocol-suites. It has been argued that the complexity of ATM is unavoidable for the reasons of providing predictive, proactive, and real-time services, such as dynamic network resource allocation, resource guarantees, virtual circuit re-routing, and virtual path establishment to accommodate subscriber QoS requests. Moreover, the problem in delivering QoS with ATM is its incapability to achieve end-to-end QoS when the true end-to-end bearer service is not pervasive ATM. Such partial QoS measures often have their effects masked by the effects of the traffic distortion created from the remainder of the end-to-end path in which ATM does not reside, and thereby leading to an ineffectual efforts to deliver QoS. Apart from its limited reachability and complexity, ATM is still an attractive mechanism for supporting network-layer QoS.
2.3.3.2 **Frame Relay QoS**

Frame Relay (FR) is another link layer technology, which attempts to provide a simple mechanism for arbitration of network over-subscription. FR decouples the characteristics of the network access link from the characteristics of the virtual circuits, which connect the access system to its group peers. Each virtual circuit is configured with a traffic committed information rate (CIR), which conforms to a commitment on the part of the network to provide traffic delivery. However, any virtual circuit can also accept overflow traffic levels - bursts which may transmit up to its excess information rate (EIR). Such excess traffic is marked by the network access gateway using a single bit indicated in the frame header, termed the "Discard Eligible" (DE) bit.

The interior of the network uses three basic levels of queue threshold to manage switch queue congestion. At the first level of queue threshold, the network starts to mark frames with Explicit Congestion Notification (ECN) bits: Forward ECN and Backward ECN. FR uses these bits to perform congestion mentioned in the previous section. At the second level of queue threshold, the switch discards frames which are marked as DE, honouring its commitment to traffic which conforms to committed information rates on each circuit. Finally, if the switch passes the third threshold of the queue, it discards frames which form part of committed flow rates.

FR indicates that it is possible to provide reasonable structures of basic service commitment, together with the added capability of provision of over-commitment using a very sparse link level signalling set - the DE, FECN and BECN bits. FR networks operate within a locally defined context of using selective frame discard as a means of enforcing rate limits on traffic as it enters the network. This is done as the primary response to congestion. The basis of this selection is undertaken without respect to any hints provided by the higher-layer protocols. This makes the support of QoS at the higher layer, especially network-layer, an unclear solution.
2.3.3.3 IEEE 802.1D QoS

The IEEE 802.1 Internetworking Task Group has recently extended the original IEEE 802.1D standard [20] to include support for traffic classes and dynamic multicast filtering [21]. This provides a method to allow preferential queuing and access to media resources by traffic class, on the basis of a priority value (user_priority) signalled in the frame.

The 802.1D draft standard defines the user_priority as a 3-bit value, resulting in a variable range of values between 0 and 7, with 7 indicating the highest relative priority and 0 indicating the lowest relative priority. The standard does not make any suggestions on how the bits should be used by the end-system or by network elements - it only suggests that packets may be queued by LAN devices based on their relative user_priority values.

While it is clear that this user_priority may indeed be useful in some QoS implementations, it remains to be seen how it will be practically beneficial. IETF ISSLL WG has been studying how the user_priority values may be used in conjunction with the Subnet Bandwidth Manager (SBM) - a proposal that allows LAN switches to participate in RSVP signalling and resource reservation objectives [22] - to support Int-Serv model [23].
2.3.4 Discussion

In this section, we have discussed various mechanisms that are available and can potentially be used to build a QoS framework for the Internet. It should be emphasised that this list of mechanisms is by no means an exhaustive one. New mechanisms can be derived from these mechanisms by combining some of them together or by re-designing a mechanism to fit a particular purpose or environment. In choosing or designing a mechanism for the Internet, one may consider the following general criteria:

- **Scalability**
  This is one of the most important criteria. It has been observed that technologies that are too complex or have poor scaling properties do not integrate well into the diversity of the Internet. Therefore, a mechanism suitable for the Internet tends to be simple and work well under a heterogeneous environment.

- **Functionality**
  It is generally desirable that a mechanism contains a rich set of functionality, e.g., multicast support; security, etc. However, the functionality should be balanced with the complexity of the mechanism. A fundamental requirement is that a mechanism should not exclude or should integrate well with other mechanisms.

- **Flexibility**
  Internet today is a huge and fast growing network. To cope with this rapid changing network infrastructure, a QoS mechanism should be easily configurable. In some cases, it may also be desirable to be programmable for adapting to the development of new protocols.

- **Robustness**
  It is noted that the Internet is a global network that spans across numerous administrative domains and a variety of transmission technologies. In order to provide end-to-end QoS, a mechanism should be robust enough in the sense that it cannot be too sensitive to parameter settings. It is desirable that a mechanism can absorb any minor control error and traffic distortion.
2.4 Summary

This chapter has discussed two major issues of QoS. It has described how QoS requirements are specified in terms of three basic attributes: performance, management policy and service commitment. Performance measures capture the network-imposed traffic characteristics that the service incurred, while policy measures describe features and high-level requirements that the service can offer. Besides these two sets of measures, a service commitment is required to describe the level of agreement that the service provider will promise. To help illustrate the QoS models, this chapter has reviewed the QoS specifications embedded in the current IETF service models.

Another issue discussed in this chapter is the QoS mechanism. QoS mechanisms are tools that allow service providers to build their desirable QoS requirements. Currently available QoS mechanisms can be classified into three major categories: provisional, control and extrinsic mechanisms. Provisional mechanisms deal with service establishment and negotiation. According to established or negotiated levels of QoS, control mechanisms provide necessary traffic control functions over data flows. Besides these primitive mechanisms, several well-defined link-layer (or extrinsic) mechanisms can be used to deliver the network-layer (i.e., IP) QoS requirements. This chapter has briefly reviewed these mechanisms and presented a general guideline on choosing them.

In summary, this chapter has provided a background understanding of issues about QoS. From the following chapters, we will focus on solving several important issues that arise from supporting QoS models over the Internet.
Chapter 3

Integrated Services with RSVP over ATM

3.1 Introduction

While Section 2.2.1 has outlined the Int-Serv model, this chapter discusses how it can be effectively supported. ATM has been chosen as our candidate QoS mechanism because:

(1) ATM is widely available today. With support from the telecommunication industry, ATM has become an important link-layer technology. High-speed backbone networks with mainly ATM permanent virtual circuits (PVCs) have emerged recently;

(2) ATM has the ability to construct a virtual circuit (VC) with a specified QoS. Its QoS model is, to a certain extent, similar to the Int-Serv model. Therefore, it is logical to examine if they can be integrated together; and

(3) ATM has the ability to handle multicast (point-to-multipoint or p2mp connection) VCs associated with a specified QoS. Even though the existing form of multicast VCs is still very limited, it may be used as a basic means for supporting IP multicast.

These features of ATM have motivated our investigation of putting Int-Serv over ATM. It should be emphasised that other QoS-aware link-layer mechanisms, mentioned in Section 2.3.3, can also be utilised for this purpose. However, this chapter focuses only on ATM.
3.2 Related Work

There are at least three fundamental problems that need to be resolved in supporting Int-Serv model over ATM: (1) mapping IP flows onto ATM VCs; (2) integrating RSVP into ATM; and (3) integrating routing protocols into ATM.

3.2.1 Mapping IP flows

Early work of Classical IP over ATM (CLIP) [24] and Multicast Address Resolution Server (MARS) [25] from IETF have solved part of this problem. CLIP addresses IP unicast best effort traffic over ATM. It is based on a Logical IP Subnetwork (LIS), which is a separately administered IP subnetwork (subnet). Hosts within a LIS communicate using native ATM mechanism, while hosts from different subnets communicate through an edge device. CLIP provides an Address Resolution Protocol (ATMARP) for ATM edge devices to resolve IP addresses to ATM native addresses. For each pair of IP/ATM edge devices, a single VC is created on demand and shared for all traffic between them. To address the IP multicast flow, MARS complements ATMARP by allowing an IP address to be resolved into a list of native ATM addresses, such that a collection of VCs can be created to serve the corresponding IP multicast flow. In addition, if the ATM "cloud" is made up of a number of LISs, it is then possible to use "short-cuts" from a node on one LIS directly to a node on another LIS, thereby avoiding router hops between the LISs. The IETF ION WG's Next Hop Resolution Protocol (NHRP) [26] is a mechanism for this purpose. Given a destination IP address, NHRP determines the ATM address of the egress point on the ATM network.

On the ATM side, the ATM Forum has also proposed similar means to address the support of IP best effort traffic over ATM, known as LAN Emulation (LANE) [27,28] and Multiprotocol Over ATM (MPOA) [29]. The goal of MPOA is to achieve efficient transfer of inter-subnet unicast data under a LANE environment. MPOA integrates LANE and NHRP to preserve the benefits of LAN Emulation, while allowing inter-subnet communication over ATM VCs without requiring routers in the data path. MPOA provides MPOA Clients (MPCs) and MPOA Servers (MPSs) as well as protocols that are
required for MPCs and MPSs to communicate. MPCs issue queries for ATM addresses and receive replies from the MPS using these protocols. To address IP multicast, MPOA also employs the concept of MARS for resolving the ATM addresses.

Given that the above mentioned protocols address only parts of the problem, the key remaining issues are the translation of QoS and the support of native IP multicast with heterogeneous QoS. The IETF ISSLL WG has defined a RFC [30] to supplement the QoS translation issue as well as [31] to address the QoS multicast issue. Ref. [31] suggests four possible models for supporting QoS heterogeneity over ATM: (1) the full heterogeneity model in which a single reservation is forwarded onto several VCs each with a different QoS; (2) the limited heterogeneity model where exactly one QoS VC is used along with a best effort VC; (3) the homogeneous model where reservation is mapped to a single ATM VC; and (4) the aggregation model where multiple reservations are aggregated into a single VC.

3.2.2 Integrating RSVP signalling

The ISATM subgroup of the ISSLL WG has proposed a suite of RFCs [31,32,33] to address this problem. These documents provide a guide to the issue of integrating RSVP into ATM. Ref. [31] discusses modes and models for RSVP operation over ATM as well as the management of ATM VCs for RSVP data and control. Refs. [32 and 33] discuss implementation requirements and guidelines for RSVP over ATM, respectively.

3.2.3 Routing protocol integration

The problem of routing integration has not been fully explored. Obviously, RSVP is not a routing protocol because it only conveys QoS information. However it provides valuable input, such as QoS and network load information, to the routing protocol for ease of path determinations. In other words, instead of only asking the IP next hop for a given destination address, it might be worthwhile for RSVP to provide QoS information of the flow if the routing protocol can use this information to determine a route.
ATM routing has also considered the similar problem of QoS routing through the Private Network-to-Network Interface (PNNI) [34] routing protocol. It attempts to route ATM VCs on a path that can support users' needs. As of the date of writing, it is still unclear how RSVP and native IP routing protocols can be integrated into an ATM environment running PNNI.

3.3 Proposed integration model

The approaches proposed by the IETF and ATMF, mentioned in the last Section, can be summarised as a peer model. In this model, when a RSVP signalling message reaches the boundary of an ATM cloud, the edge node or gateway first resolves the target ATM address(es) and then converts the signalling message into a native ATM signalling message. This signals a hypothetical VC connecting the ingress and the egress node. At the egress point, the original RSVP message is re-generated. This RSVP message carries on its trip to the designated receiver. Once the receiver decides to make a reservation and returns a RSVP message (RESV), a similar mechanism takes place on the reverse path except for the fact that an actual QoS VC is created rather than a hypothetical one. Although this model confines different signalling and control mechanisms to individual domains, it has unavoidably introduced complexity at the edge devices or gateways between different signalling domains. It has been noted that Q.2931 signalling is a complete protocol and has already imposed a significant burden for signalling in all public and private connections. The additional signalling conversion between RSVP and Q.2931 further loads the edge devices or gateways. Eventually, it may lead to a serious performance issue. Moreover, unless the original RSVP message is tunnelled\(^1\) to the egress point, conversion of QoS parameters may introduce error and discrepancy. As indicated in the last Section, apart from concerns about performance, the interaction between RSVP and PNNI is still an outstanding issue.

For the ATMF solution, the current version of MPOA does not explicitly mention how RSVP signalling can be integrated and how Int-Serv model can be supported. Moreover,

\(^1\) Tunneling refers to the operations of encapsulation and delivery.
MPOA incorporates protocols such as LANE, NHRP and MARS, in additional to the native ATM signalling protocol and has become a very complicated protocol suite. Therefore, its performance and scalability are still an unknown issue.

While working independently yet with a similar time frame (Spring 1996) as the IETF activities, we have developed a different integration model, called fusion model. Inspired by the concept of IP switching, e.g., Ipsilon's IP Switch [35], our model concentrates on the QoS control mechanism rather than handling best effort flows. Our model merges ATM and IP control mechanisms into a unified one. As a result of minimising all unnecessary signalling conversion and duplication of operations, it anticipates a better efficiency than the peer model. Since the fast data switching feature of ATM is well preserved, the overall data forwarding efficiency is maintained. Moreover, this model enables a unified IP based routing protocol in both native IP and IP/ATM domains, therefore, the issue of routing integration becomes a trivial problem. Last but not least, since limitations of the native ATM signalling are relaxed, a more flexible and efficient heterogeneity model can be designed. Details of the heterogeneity model and mechanism are discussed in Section 3.5.3.

It is important to note that both models mentioned above are not mutually exclusive. Network providers can implement part of their network using different integration models, according to their requirements of efficiency, scalability and management policy. In fact, while requiring no modification on switch hardware as well as native ATM signalling and control, our model works gracefully alongside the normal native ATM operation.
3.4 Proposed Architecture

Figure 3.4.1 depicts the overview of the proposed Int-Serv with RSVP over ATM shortcuts (ISAC) environment. This architecture, derived from the fusion model, consists of two types of devices: ISAC client (ISAC-C) and ISAC router (ISAC-R), interconnected using ATM links. An ISAC-C is a conventional host equipped with ISAC software and running user applications, while an ISAC-R consists of a core ATM switch tightly coupled with a controller running another ISAC software.

![Diagram of ISAC architecture](image)

Figure 3.4.1 Overview of an ISAC environment

3.4.1 Software Architecture

Our software system design strives to achieve three fundamental objectives: (1) ensuring compatibility with existing IETF and ATMF standards, applications and system software, (2) maintaining data path efficiency, and (3) assuring portability for different OS and switch control protocols.
Figure 3.4.2 and Figure 3.4.3 depict the building blocks of the client and router controller software. Both ISAC-C and ISAC-R controller contain a stack of software modules commonly found in the kernel of a host equipped with an ATM network interface card (NIC). This software stack consists of: (1) a socket based and a proprietary ATM based application interface (API); (2) a transport protocol layer supporting TCP and UDP over IP; (3) an ATM device driver equipped with a network interface such as BSD ifnet or DLPI; and (4) a firmware running on the ATM NIC. Beside this software stack, two additional modules, RSVP daemon (RSVPD) and VC manager (VCM), are included in the ISAC-C and ISAC-R controller. Instead of residing in the kernel space, they are kept in user space for maintaining portability. Table 3.4.1 summarises the major functions of these modules.

<table>
<thead>
<tr>
<th>Module</th>
<th>Functional Description</th>
<th>Client</th>
<th>Router Controller</th>
</tr>
</thead>
<tbody>
<tr>
<td>RSVPD</td>
<td>Translating RAPI calls into RSVP messages</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>Handling RSVP messages</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>Network resource management (PSB and RSB)</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>Binding flow with reservation</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>Interacting with VCM via TCI</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>VCM</td>
<td>link-layer resource management (VSB)</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>Binding reservation with VC</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>VCI allocation</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>parameter mapping</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>switch control and VC configuration via SCI</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Others</td>
<td>policing &amp; shaping</td>
<td>NIC&lt;sup&gt;b&lt;/sup&gt;</td>
<td>switch&lt;sup&gt;c&lt;/sup&gt;</td>
</tr>
<tr>
<td></td>
<td>Admission control</td>
<td>switch</td>
<td>switch</td>
</tr>
<tr>
<td></td>
<td>QoS provisioning or scheduling</td>
<td>switch</td>
<td>mroute&lt;sup&gt;d&lt;/sup&gt;</td>
</tr>
<tr>
<td></td>
<td>Maintaining routing information</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3.4.1 Functions for major software building blocks

Generally, an RSVPD performs most of the functions described in [36]. It generates, interprets and forwards RSVP messages (e.g. PATH, RESV, TEAR, etc.). Particularly in an ISAC-C, application programs use the RSVP API library [37] (RAPI) to access the
available services. Upon receiving a request, RSVPD then translates the RAPI call into signalling messages. In the ISAC-R controller, RSVPD uses a general framework known as Routing Support for Resource Reservation (RSRR) for interfacing the multicast routing protocol daemon (mrouted). With regards to resource management, RSVPD is also responsible for allocating, maintaining, and binding resource states\(^2\) with the established flows. In order to complete a reservation process, an RSVPD may interact with VCM via a traffic control interface (TCI).

A VCM, on the other hand, focuses on managing link-layer resources. It maintains and associates resources with reservations by manipulating a virtual connection state block (VSB). For an ISAC-R controller, a VCM is also responsible for mapping the requested Int-Serv services into ATM services with an appropriate set of parameters. According to the request issued from RSVPD, the VCM creates, modifies or deletes a virtual connection by sending commands defined in the switch control interface (SCI) to the underlying switch. In order to keep our design portable and maintain data path efficiency, the VCM does not include functions, such as traffic policing, shaping, scheduling and admission control. Since these functions are supported by the switch and NIC, they are usually either proprietary or inaccessible by an ordinary user.

Figure 3.4.2 ISAC-C software architecture

\(^2\) According to [36], resource states are organized into a path state block (PSB), a reservation state block (RSB) and a traffic state block (TSB).
3.5 Functional Details

To illustrate the operational details of the proposed architecture, our discussion refers to a typical reservation scenario with one sender $S_1$ and two receivers $D_1$ & $D_2$ on a multicast session as shown in Figure 3.5.1. All end users are located in three different logical IP subnets (LIS). Since our proposal concentrates on supporting flows with associated QoS, we assume all LIS are connected using some form of basic hop-by-hop routing mechanism with no QoS guarantee. Such best effort connection can use CLIP, ELAN or any IP switching approach that is available in the LIS. All RSVP signalling messages are transported over these basic connections.

In the referenced example, the sender initiates a RSVP session and sends a PATH message to a multicast group address. This PATH message travels through the network to all members of the multicast group and PSBs at all ISAC-Rs along the multicast tree are updated. When these PATH messages reach the receivers, each receiver decides independently if it intends to request a reservation for the session. If the receiver decides to make a reservation, it responds with a RESV message. This RESV message traverses the reverse path back to the sender. On the way back to the sender, it is intercepted by the
ISAC-Rs. If sufficient resources are available, a soft reservation state is established in the routers. Otherwise, a RESV-ERROR message is generated and issued back to the receiver. The RESV-ERROR message may also be intercepted by other routers, which delete all corresponding reservation states. Eventually, after a reservation tree has been established with the sender as the root and the receivers requesting reservations as the leaves, the data flow starts. During the lifetime of the flow, the sender periodically sends PATH-REFRESH messages to refresh PSBs established at the routers involved. Similarly, the receivers also send RESV-REFRESH messages to keep their reservations active. If no refresh message is received by the time a timer has expired or an explicit TEAR message is received, the router will issue a PATH-TEAR or RESV-TEAR message to the receiver or the sender respectively. Finally, the corresponding state blocks along the flow path will be cleared.

![Figure 3.5.1 A Typical RSVP reservation scenario](image)

In the following sections, we discuss in details three major operations: data VC management, QoS parameter mapping and multicast support.

### 3.5.1 VCI allocation and data VC management

To set up an end-to-end virtual connection shortcut, each ISAC-R along the flow path is responsible to create its own VC segment. In order to ensure that these segments can be connected, a virtual circuit identifier (VCI) allocation protocol (VCAP) is developed. VCAP is a simple protocol that assigns VCIs to the downstream neighbour(s). It is currently implemented in our RSVPD. Figure 3.5.2 summarises VCAP into two phases. The detailed allocation mechanism for the reference example is illustrated Figure 3.5.3.
In VCAP, all VCIs are carried\(^3\) in the logical interface handler (LIH) field of the RSVP messages. Upon receiving a PATH message, \textit{RSVPD} first extracts the VCI from the LIH field. This VCI will be registered in the local PSB, if it has not already been used. Then \textit{RSVPD} accesses the VSB and selects an unused VCI for each outgoing link. This also triggers the creation of a VSB entry, which specifies the selected VCI and the associated link. Finally, the received PATH message is updated with the selected VCI and is forwarded to the downstream node.

In the resource reservation phase, after registering the VCI attached with the received RESV message in its local RSB, \textit{RSVPD} passes the requested service parameters and VCIs to its \textit{VCM}. \textit{VCM} then triggers an admission check and resource allocation procedure at the underlying switch. If the response from the switch is positive, \textit{RSVPD} will replace the VCI in the RESV message with the one kept in the corresponding PSB and forward the modified message to the upstream node. Once the resource reservation phase in all ISAC-Rs along the path are successful and completed, an end-to-end virtual connection is established. However, if there is an admission failure at any intermediate node along the path, an RESV-ERROR message will be generated and forwarded back to the receiver. Upon receiving an RESV-ERROR message, \textit{RSVPD} clears the VCI entry of the corresponding RSB and contacts \textit{VCM} to release any allocated resources at the switch.

\[\text{(RESV) resource reservation Phase}\]

- strip off the VCI from the RESV message and register it in RSB
- perform admission check and proceed downward if passed, otherwise clear the VCI entry in RSB
- Replace the VCI in the RESV message with the VCI of the corresponding PSB
- Forward the modified RESV message upstream
- send RESV ERROR message

\[\text{(PATH) establishment phase}\]

- strip off the VCI from the PATH message and register it in PSB
- Select a new VCI for each outgoing link
- register the VCI in VSB and associate it with the outgoing link
- forward the PATH message downstream
  With the VCI encoded in LIH field

\[\text{Figure 3.5.2 VCAP flow diagram}\]

\(^3\) The 24-bit long VCI resides in the least significant three bytes of the LIH field.
3.5.2 Parameter Mapping

Parameter mapping, which concerns mapping the QoS parameters from Int-Serv model to appropriate ATM services, is necessary before an admission check at the ATM switch can be performed. Our mapper design considers only the traffic management specification since no ATM signalling is required in our architecture. With reference to [30], we have implemented the mapping matrix shown in Table 3.5.1.

Mapping Controlled-load Service flows (CLS flows) is straightforward. They are mapped onto the ATM VBR service with PCR01 equal to the peak rate $p$, SCR0 equal to the bucket rate $r$ and MBS0 set to the bucket depth $b$ specified in their TSpec.

---

4 Our ATM switch currently implements only UNI 3.x traffic management specification [38,39].
Guaranteed Service flows (GS flows), however, are more difficult to deal with. They are mapped to the CBR service in ATM with \( PCRO \) equal to their requested service rate \( R \). \( PCRO1 \) is set to the peak rate \( p \) which implies tagging is always enabled. \( CDVT \), which is the bucket depth parameter for \( PCR \), is set to \( b/p \). Note that under UNI3.x, \( CDVT \) is not signalled explicitly, but our ATM switch currently allows setting it independently for each connection. If the switch does not support this feature, additional buffering might be needed at the switch to account for the depth of the token bucket. Otherwise, the value of \( b \) should be restricted to reflect the \( CDVT \) setting at the switch. For the setting of \( AdSpec \), we simplify the process by setting \( C = 0 \) and advertising the maximum \( CDV \) in \( D \). If the scheduler of the switch is known, a slack term can also be used to lower the required resource reservation. However, in order to maintain portability, we do not use the slack term.

Furthermore, all Int-Serv-IP parameters that are mapped to ATM parameters must be corrected for the per-cell or per-AAL-frame overhead and the effect of packet segmentation. Since parameters in ISIP are specified in bytes or bytes/sec, whereas they are in cells or cells/sec in ATM, a unit conversion is also necessary. Similar to [30], our conversion formula assumes the worst case packet lengths and is given in Table 3.5.1.

\[
X_c = \frac{X}{48} \times \left(1 + \frac{55}{m}\right)
\]

where \( m \) = minimum policed unit (byte)

\( X \equiv \) parameter in byte or byte/sec \( \in \{p, r, b, R\} \); 

\( X_c \equiv \) parameter in cell or cell/sec \( \in \{p_c, r_c, b_c, R_c\} \)

<table>
<thead>
<tr>
<th>ISIP [7,9] Parameters</th>
<th>ATM UNI 3.x Traffic Management Parameters [38,39]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service</td>
<td>PCRO</td>
</tr>
<tr>
<td>GS ((p, r, b, R))</td>
<td>CBR</td>
</tr>
<tr>
<td>CLS ((p, r, b))</td>
<td>VBR</td>
</tr>
</tbody>
</table>

Table 3.5.1 Parameter mapping matrix
3.5.3 Multicast Support

In this section, we consider the multicast aspect of the reservation process that we have omitted in the previous discussion. Referring to the previous example, as the RESV messages issued by receivers travel upstream to the sender, they are merged by the ISAC-R. If both receivers request the same QoS, the merge point router will forward the RESV message upstream after a p2mp (in this example, 1-to-2) VC is created. This p2mp VC will have VCIs registered in PSB and RSB. When the RESV message reaches the sender, a homogeneous multicast tree is established.

Nevertheless, since each receiver makes reservations independently, heterogeneity can occur when receivers request different QoS's within a session. This means that the amount of requested resources may differ on a per next hop basis. This contradicts the native ATM signalling in which a multicast VC tree has to be homogeneous type. To resolve this incompatibility, [31] proposed four models. Table 3.5.2 summarises their pros and cons.

In general, it is desirable that a VC mapping model should give users exactly what they have requested if resources are available. The model should also avoid sending duplication of packets for network efficiency concern. From the network provider's perspective, the model should be easily configured and provisioned. A good mapping model should also maintain fairness in resource sharing, for example, the problem of "free-riding" should be avoided.

5 "free-riding" problem is where users reserving no or less resource would share a QoS VC with more resources which are originally established for other users.
<table>
<thead>
<tr>
<th>Model</th>
<th>Description</th>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full heterogeneity</td>
<td>• provide separate VC for each distinct QoS</td>
<td>• give users exactly what they requested</td>
<td>• more resources are needed</td>
</tr>
<tr>
<td>Figure 3.5.4(a)</td>
<td>reservation</td>
<td></td>
<td>• many duplication of packets are sent on the network</td>
</tr>
<tr>
<td>Limited heterogeneity</td>
<td>• provide separate VC for a limited class of QoS</td>
<td>• only two VC are needed</td>
<td>• limited packets are duplicated at the network layer</td>
</tr>
<tr>
<td>Figure 3.5.4(b)</td>
<td>• only best effort and a single alternate QoS are</td>
<td></td>
<td>• request can be rejected even though actual network resource is still available</td>
</tr>
<tr>
<td>Homogeneous</td>
<td>• a single QoS VC for all reservations</td>
<td>• only a single QoS is needed</td>
<td>• supporting only two QoS classes is not satisfactory</td>
</tr>
<tr>
<td>(No heterogeneity)</td>
<td>• even best effort receiver is forced to use QoS</td>
<td></td>
<td>• &quot;free-riding&quot; problem</td>
</tr>
<tr>
<td>Aggregated</td>
<td>• a &quot;large&quot; VC is setup for different reservations</td>
<td>• multiple sessions can be multiplexed onto the same VC</td>
<td>• best effort receiver may receive no data or be rejected.</td>
</tr>
<tr>
<td>heterogeneity</td>
<td></td>
<td>• no or less signalling latency for VC setup</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• both heterogeneity and dynamic QoS requests are possible</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• difficult to choose an appropriate QoS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• no aggregation model is available and need further study.</td>
</tr>
</tbody>
</table>

Table 3.5.2 Summary of VC assignment models supporting multicast heterogeneity

(a) Full heterogeneity

(b) Limited heterogeneity

Figure 3.5.4 VC Assignment for heterogeneous reservation
3.5.3.1 Piping Concept

Unlike the end-to-end nature of the native ATM signalling, our reservation mechanism in VCAP is per next hop basis. This feature enables us to establish VCs in a more flexible way. To support such RSVP heterogeneity over ATM, we introduce the concept of *piping*. In this concept, an end-to-end heterogeneous multicast tree is decomposed into a collection of pipes, which can be of different sizes and shapes. A pipe in this case represents a VC segment in an ISAC-R.

Figure 3.5.5 illustrates the heterogeneous multicast tree of the previous example. It is composed of two different pipes as illustrated in different colours and shapes. In the following paragraphs, we describe details of how they are created.

With reference to the previous example, as router R_2 has received a RESV message requesting a certain level of QoS (e.g. QoS_2), it processes the reservation as usual. If the admission check is passed, R_2 will create a pipe with size equal to QoS_2 and forward the RESV message upstream to R_1. When both RESV messages from D_1 and R_2 have arrived at R_1, no matter which one has come first, the RSVPD in R_1 will compare them. The VCM will try to allocate resources for the one with a larger flowspec (e.g. QoS_1). It essentially creates an ATM multicast (1-to-2) VC with parameters equal to QoS_1. Finally, after the RESV message has arrived at the sender, an end-to-end heterogeneous multicast tree having various segments is established.

![Diagram](image)

*Figure 3.5.5 An example of heterogeneous multicast using piping concept*

---

6 As indicated in [4], "A RESV message forwarded to a previous hop carries a flowspec that is the "largest" of the flowspecs requested by the next hops."
3.5.3.2Handling excess traffic

Since R₂ reserves less resource for D₂, the ATM switch in R₂ will tag all excess traffics. To handle these excess traffics, we propose two possible approaches:

1) Packet Discarding

When congestion occurs, tagged traffic may be discarded. This mechanism, however, is completely implementation specific. Tagged cells, in ATM perspective, are treated with a behaviour which is "best effort" in the sense that they are transported when bandwidth is available, queued when buffers are available, or dropped when resources are over-committed. We suggest that Early Packet Discard (EPD) feature, if available, should be used. When a tagged cell is discarded, EPD ensures that all cells belonging to the same packet are discarded at the same time since a partial packet is useless at the receiving end.

This best effort concept is not identical to the one described in Int-Serv services [7, 9]. Nevertheless, with the existing implementation of our ATM switch, we argue that it is the best possible option.

2) Alternative-path forwarding (APF)

Similar to EPD, in which cells belong to the same packet are tagged at the same time during policing process, but instead of discarding these tagged cells, we propose to forward them onto an alternative path. APF not only can improve the probability of packet loss, but it also provides a hook for supporting multi-layer service, in which data are forwarded using more than one paths, each of which can have different QoS.

To realize the APF, we suggest the following modification to RSVP mechanism. At the resource reservation phase, when R₂ detects a condition where resource allocated to D₂ (i.e. Receiver_TSpec) is less than the Sender_TSpec specified in the PATH message, it extends the original PATH message to include a child object. This
object contains a TSpec of the excess traffic and a new⁷ VCI specifying the alternative VC for forwarding the excess traffic. R₂ then sends this PATH-C message to its outgoing link(s). Similarly, when a downstream router receives this message, it processes the message in the same way as an ordinary PATH message. When the PATH-C message arrives at the receiver(s), each receiver can respond with one of these three cases described below:

(i) Ignoring the child object
It merely signals the splicing node (i.e. R₂ in this example) to discard all excess traffic during congestion.

(ii) Returning a RESV-C message with an corresponding child reservation object
It requests a particular reservation along the alternative path and essentially requests for a multi-layer service mentioned earlier. In this case, the reservation process and all RESV-C messages are terminated at the splicing node.

(iii) Returning a RESV-C message with an empty child reservation object
In this case, an alternative path with best effort service will be set up.

For the latter two cases, when the RESV-C message reaches the splicing node, a heterogeneous multicast tree having alternative path(s) is established.

The key remaining issue for APF is cell sequencing and reassembling. It refers to the situation where cells belong to the same flow, but forwarded on different paths, may arrive out of sequence. We suggest two methods to handle this problem:

(1) Frame-level interleaving (FLI)
In order to preserve cell sequencing and ensure packet can be reassembled, the forwarding module can restrict all data switching to occur only at the packet or AAL5 frame boundaries. It implies that a mechanism of detecting the frame boundaries is required at every splicing node. Since packets from the alternative path

⁷ This VCI is allocated in the same way as in VCAP path establishment phase.
may still arrive out of sequence relative to the main data path, a transport protocol such as TCP or RTP is required to re-order the received packets. A major advantage of this alternative is that it requires only minor modification of the forwarding module. However, in case of long packet, the per-alternative-path QoS may be difficult to achieve because cell interleaving occurs only at packet boundaries. Moreover, if the difference in QoS between alternative paths is too big, the transport protocol may not be able to re-order the received packets.

(2) **Token forwarding**

Figure 3.5.6 depicts the design of an output module employing the concept of *Token Forwarding*. It is based on a common per-VC, shared-memory design with an extension of *token*\(^8\) handling mechanism. In common shared-memory architecture, when a cell enters the module, it is appended to an appropriate link-list in the shared-memory and waits for delivery. At the same time, a token is generated and placed into the corresponding token queue. A token in this case is a *pointer* which contains an input time-stamp indicating the time-of-arrival and the address of its corresponding cell. According to its scheduling policy and the resource reservations associated with each queue, the token scheduler selects a token from one of the queues. Then, the cell selector delivers the corresponding cell and updates the link-list with reference to the selected token. The token policer, at the input of a token queue, examines the incoming traffic profile of a flow. For ATM, the out-of-profile token and cell will be tagged. These tagged cells are subjected to being discard during congestion.

To support the Int-Serv model and in particular APF, this common shared-memory architecture is augmented at the token handling mechanism, as shown in Figure 3.5.6. For the cell queues, instead of per-VCI basis, they are organized in a per-flow basis. For the token queues, instead of tagging the out-of-profile tokens or cells, our

---

\(^8\) The term "token" in this Section is different from its conventional meaning in the context of leaky bucket filter or shaper. A token in this case refers to a pointer that contains an input time-stamp indicating the time-of-arrival and the address of its corresponding cell.
design places the tokens into the alternative token queues. This is, in effect, employing more than one token queues (usually less than three) to represent a flow. According to the resource reservations for the alternative paths, the token queues are allocated with certain scheduling priorities. Similar to the original design, the token scheduler selects a token, which represents a cell in the per-flow cell queue. Since all cells within a flow en-queue to the same cell queue, the cell-transmission order is well preserved, i.e., delivering order is identical to incoming order.

As an example, suppose a receiver has requested a flow with an alternative path associated with $QoS_m$ and $QoS_a$, respectively, using the extended RSVP signaling mentioned in Section 3.5.3.1. These QoS profiles will first be translated into the appropriate parameter formats. Then, a set of two QoS token queues, in addition to the common best-effort token queue, will be bound with a packet queue for this flow. Each of these token queues is allocated a weight according the required QoS profile. At the input of the module, two instances of token policers will be used to direct the incoming token traffic. The first policer directs all conformance token traffic into the main token queue ($QoS_m$) while excess traffic will be subjected to the second level policing. If the excess traffic further exceed the second profile ($QoS_a$), the over-committed traffic will be put into the best-effort (or common) token queue. Under this example, the flow is actually characterized by two profiles, each of which is serviced independently, but the flow sequence is well preserved.

Although this approach requires modification to the common switch design, it does not have the limitations found in the FLI approach. In particular, with TF, the APF mechanism is strictly an internal operation within a switch, there is no additional requirement at the receiver end, whereas with FLI, receiver is required to listen to more than VCs (the main path and the alternative paths). In addition, this overall mechanism of piping and token forwarding matches the spirit of the Int-Serv and RSVP models, i.e., "non-conforming packets should have best-effort behavior" [7,9]. And more importantly, the overall piping mechanism does not have the problems found in other proposed heterogeneity model, as mentioned in Table 3.5.2.
3.6 Implementation Details

To verify the design and evaluate performance of the ISAC framework, we have implemented a prototype ISAC network using the ATM testbed at NAL. In this section, the experimental setup and testing strategy are discussed.

3.6.1 System modules

The architectural building blocks described in the previous sections are implemented on Sun workstations running Solaris 2.5.1 and equipped with SunVideo graphics cards. All workstations are connected to a Fore ASX-200WG ATM switch using the Fore SBA200 ATM network interface cards. The ISAC-Cs are UltraSparcStations with ForeThought 4.1.0 driver software installed that contains both socket based and Fore's proprietary ATM API. They run client version of RSVPD and VCM in user space. The RSVPD is implemented based on a Solaris porting [40] of the ISI's rsvpd implementation [41] while the VCM is developed completely in-house according to the specifications mentioned.
earlier. The ISAC-R controller is implemented on a SparcStation20. In addition to RSVPD, the controller runs a router version of VCM which includes a switch control interface, a parameter mapper, etc. Since SNMP is currently the only switch control protocol available on our switch, we adopt it as the SCI to access the content of the switch's management information base (MIB). By manipulating parameters inside the MIB, we can access the switch status and configure VCs on demand. Table 3.6.1 summarises all modules developed in our lab.

<table>
<thead>
<tr>
<th>Software Component</th>
<th>Development</th>
</tr>
</thead>
<tbody>
<tr>
<td>RSVPD</td>
<td>Based on SUN's Solaris porting of ISI rsvpd rel-4.1a</td>
</tr>
<tr>
<td>VCM</td>
<td>In-house</td>
</tr>
<tr>
<td>SCI (SNMP interface)</td>
<td>In-house</td>
</tr>
<tr>
<td>Transport of RSVP messages</td>
<td>Fore's ELAN v1.0</td>
</tr>
<tr>
<td></td>
<td>Fore's CLIP</td>
</tr>
<tr>
<td>ATM API &amp; Device Driver</td>
<td>Fore's ForeThought v4.1.0</td>
</tr>
</tbody>
</table>

Table 3.6.1 List of software modules.

3.7 Evaluation Methodology

Our testing strategy is twofold: (1) verifying an end-to-end operation and (2) evaluating performance of the proposed ISAC architecture. Based on a well-known video conferencing tool called vic [42], we have developed an application running on ISAC. vic, is a real-time, multimedia application for video conferencing running over RTP. Our r-vic application enhances the original graphical user interface (GUI) by incorporating an RSVP menu. It shows the active RSVP status and allows users to make reservations. Since the socket based protocol stack from Fore does not currently support binding to a specific VCI, we have modified the ATM transport module of vic, such that the allocated VCI is acquired from RSVPD through RAPI and data is delivered using the ATM API. By using r-vic, we have demonstrated a live video conferencing application over an Int-Serv-IP environment. It shows that our ISAC works properly and complies with the IETF recommendations.
To evaluate the performance of ISAC implementation, we have also developed another instrumentation tool called *rcm*, which acts as a simple CBR MPEG2 client-server application. *rcm* emulates the delivery of MPEG2 Transport Stream (TS) over RTP/UDP/Int-Serv-IP over ATM. It incorporates all protocol overhead\(^9\) and delivers the encapsulated MPEG2 TS packets using the ATM API. Our evaluation involves measurement of signalling and control latency as well as performance of the data path. Figure 3.7.1 depicts our experimental setup, which includes two ISAC-C running *rcm*, an ISAC-R and a GN Nettest IWA95000 ATM protocol analyser. The analyser serves two purposes. First, it monitors traffic at both client and server ends by recording the arrival time of all packets that belong to a reference flow. Second, it generates background traffic with different service types and parameters. Furthermore, as indicated in Figure 3.7.1, additional loopback paths are constructed to mimic a network environment with multiple hops of ISAC-Rs.

\begin{figure}
\centering
\includegraphics[width=0.8\textwidth]{experimental_setup.png}
\caption{An experimental setup for performance evaluation}
\end{figure}

\(^9\) The overhead includes protocol headers in RTP, UDP and IP.
3.8 Results

3.8.1 Functional testing

We have demonstrated an operational videoconferencing application over ISAC running \textit{r-vic}. Figure 3.8.1 shows a videoconferencing session running under an arbitrary session with group address 224.0.222.222:1200. There are two hosts, 128.100.244.166 and 128.100.244.147 participating in this session. These hosts reside in two different subnets with netmask 0xffffff0 and are running their own IP-over-ATM mechanism with no QoS capability within their subnets. On the "Active RSVP PATHs" menu in Figure 3.8.1, it shows a live session with a set of parameters which has been reserved by the host (128.100.244.147). Through demonstrating this application, we have verified the operation of the ISAC system. The following sections focus on evaluating the performance of the system.

![Figure 3.8.1 r-vic Graphical User Interface (GUI)](image)

50
3.8.2 Performance of signalling and control

Figure 3.8.2 illustrates a flow diagram of various signalling sequences. It indicates the major events that occur at the server, client and router controller. Table 3.8.1 summarises the average measured processing time between different events over 20 runs of experiments.

As shown in Table 3.8.1, $t_1$ represents the total processing time required to issue a RSVP message after a user request has been received. Depending on the type of request, this operation may involve managing the resource state blocks and composing a corresponding RSVP message. For the case of delivering the PATH message, $t_1$ includes the time required to initialise and register all resource state blocks; format the requested service and prepare a PATH message with the appropriate parameters. It is usually larger than those of all other signalling sequences. In contrast, since $t_{1,r}$ includes only simple operations such as deleting resource states and preparing a RESV-TEAR message with no parameter, its value should always be smaller than those of other sequences.

$t_2$ and $t_3$, on the other hand, represent the total times spent on a router controller. They include the time required for processing a message, configuring a virtual connection at the underlying switch and handling resource states. While reflecting the configuration time and communication time between controller and the switch, $t_2$ contributes a large portion to the total processing time of a router. This merely confirms that SNMP is not efficient enough to be used as a switch control protocol. Furthermore, it should be noticed that $t_{2,r}$ roughly equals to the sum of $t_{2,r}$ and $t_{2,r}$. The reason for this is that modifying a reservation on an ISAC router involves tearing down the existing VC and then creating a new one. Even though $t_3$ is composed of a similar set of operations as $t_1$, it ends up taking longer time than that of $t_1$ because these operations are running on a slower machine (SparcStation 20). For the measured value of $t_4$, it shows that, except for the PATH event, the time required to generate an API event upcall for different signalling types are comparable. Similar to the situation in $t_1$, when RSVPD receives a PATH message, it initialises the necessary data structure for accommodating the resource state.
blocks. Therefore, \( t_4 \) in the PATH event experiences extra delay before an API event upcall is asserted.

\[
\begin{align*}
\text{Delivering PATH messages: } T_p & \\
\text{SNR(Rcv API)} & \xrightarrow{\tau_1} \text{SNR(Snd PATH)} & \xrightarrow{\tau_2} \text{CON(Rcv PATH)} & \xrightarrow{\tau_3} \\
\text{CON(Snd PATH)} & \xrightarrow{\tau_4} \text{RCR(Rcv PATH)} & \xrightarrow{\tau_5} \text{RCR(Upc API)} \\
\text{Making reservation: } T_R & \\
\text{RCR(Rcv API)} & \xrightarrow{\tau_1} \text{RCR(Snd RESV)} & \xrightarrow{\tau_2} \text{CON(Rcv RESV)} & \xrightarrow{\tau_3} \\
\text{CON(Set VC)} & \xrightarrow{\tau_4} \text{CON(Snd RESV)} & \xrightarrow{\tau_5} \text{SNR(Rcv RESV)} & \xrightarrow{\tau_6} \text{SNR(Upc API)} \\
\text{Modifying reservation: } T_M & \\
\text{RCR(Rcv API)} & \xrightarrow{\tau_1} \text{RCR(Snd RESV)} & \xrightarrow{\tau_2} \text{CON(Rcv RESV)} & \xrightarrow{\tau_3} \\
\text{CON(Mod VC)} & \xrightarrow{\tau_4} \text{CON(Snd RESV)} & \xrightarrow{\tau_5} \text{SNR(Rcv RESV)} & \xrightarrow{\tau_6} \text{SNR(Upc API)} \\
\text{Tearing down reservation: } T_T & \\
\text{RCR(Rcv API)} & \xrightarrow{\tau_1} \text{RCR(Snd R-TEAR)} & \xrightarrow{\tau_2} \text{CON(Rcv R-TEAR)} & \xrightarrow{\tau_3} \\
\text{CON(Del VC)} & \xrightarrow{\tau_4} \text{CON(Snd R-TEAR)} & \xrightarrow{\tau_5} \text{SNR(Rcv R-TEAR)} & \xrightarrow{\tau_6} \text{SNR(Upc API)} \\
\end{align*}
\]

![Figure 3.8.2 Event flow of various signalling sequences](image)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNR</td>
<td>Sender (Server)</td>
<td>Rev</td>
<td>Receive</td>
</tr>
<tr>
<td>RCR</td>
<td>Receiver (Client)</td>
<td>Snd</td>
<td>Send</td>
</tr>
<tr>
<td>CON</td>
<td>Router Controller</td>
<td>Upc</td>
<td>Event Up Call</td>
</tr>
<tr>
<td>Mod VC</td>
<td>Modify VC</td>
<td>Del VC</td>
<td>Delete VC</td>
</tr>
<tr>
<td>Set VC</td>
<td>Create VC</td>
<td>API</td>
<td>API Event</td>
</tr>
</tbody>
</table>

\[
\begin{align*}
\text{Table 3.8.1 Average signalling and control latency} & \\
\begin{array}{|c|c|c|c|c|}
\hline
\text{Symbol} & \text{\( t_1 \) (\mu s)} & \text{\( t_2 \) (\mu s)} & \text{\( t_3 \) (\mu s)} & \text{\( t_4 \) (\mu s)} \\
\hline
T_p & 66.3 & 95.6 & 34.1 \\
T_R & 54.1 & 130.5 & 91.2 & 14.7 \\
T_M & 46.2 & 161.3 & 64.9 & 13.0 \\
T_T & 20.2 & 30.1 & 38.7 & 14.0 \\
\hline
\end{array}
\end{align*}
\]
3.8.3 Performance of data delivery

For the performance of the data path, our primary concern is on how packet transfer delay is affected when: (1) the background traffics have different service types and produce different levels of loading; and (2) the flow path traverses several nodes. To study these effects we have conducted two sets of experiments.

First, we set up our equipment as shown in Figure 3.7.1 with the loopback path disconnected. A flow with GS and data rate\(^{10}\) of 625kBps or 985kBps (i.e., 5Mbps or 7.88Mbps) is reserved using \textit{rcm}. Then an encapsulated CBR MPEG2 transport stream is delivered over this flow and the statistics are collected using the analyser. Figure 3.8.3 plots the distributions of packet transfer delay of the reserved reference flow against various background loads. The set of background loads shown in Figure 3.8.3a are all GS but having different aggregated loading factors whereas the ones in Figure 3.8.3b are all CLS in type. From Figure 3.8.3a, we notice that the higher the background load, the larger the variance of the distribution is. Since the router (or the underlying switch) is loaded by the traffic of the same type, additional queuing delay, which depends on the level of loading will contribute to the overall transfer delay of the reference flow. Nevertheless this situation does not occur for the CLS case as illustrated in Figure 3.8.3b. It is because a CLS is mapped onto a VBR service in ATM under our implementation. Since VBR service is non-real-time in nature in ATM UNI 3.x, it always has a lower priority than the real-time service that is CBR service or GS in ISAC. This characteristic fits very well with the Int-Serv model in which GS should have a higher priority over CLS.

Furthermore, Figure 3.8.4 plots our second set of results collected from the measurement with the loopback paths being connected. It shows that the variance of packet transfer delay distribution becomes larger as the hop count of a data path increases. This characteristic holds for both types of background load (GS and CLS).

\(^{10}\) The data rates are chosen to represent two types of MPEG2 TS.
Figure 3.8.3  Experiment Set 1 (service type and level of utilization)
Figure 3.8.4 Experiment Set 2 (service type and number of hops)
3.9 Summary

This chapter discusses an architectural framework for supporting Int-Serv Internet with RSVP over ATM short-cuts. Our design integrates RSVP and native IP routing protocol into ATM such that duplication of functionality in ATM domain can be eliminated. Therefore, the overall system efficiency can be improved. Our system comprises ISAC-Cs interconnected using ISAC-Rs. The ISAC-R consists of a core ATM switch tightly coupled with a controller. There are two major software building blocks, RSVPD and VCM, being developed. RSVPD handles all RSVP messages and network-layer resources whereas VCM manages the link-layer resources. Particularly, the router version of VCM inside a controller is responsible for managing and controlling the underlying ATM switch.

Furthermore, we have introduced a concept of piping to efficiently and effectively support heterogeneous multicast. Concerning the fundamental conflict on the definition of "best effort" between ATM and Int-Serv model, we have also proposed a simple Token Forwarding mechanism, in co-operation with piping and Alternative-Path Forwarding to support heterogeneous multicast. Our approach not only matches the spirit of Int-Serv and RSVP models, but it also eliminates the limitations found in other heterogeneity models proposed in [31].

We have implemented a prototype ISAC network over our ATM testbed. By running a videoconferencing application, we have demonstrated an operational ISAC environment that supports video over RTP/UDP/Int-Serv-IP/ATM. We have conducted a series of experimental measurements on the latency of signalling and control path as well as the transfer delay of data delivery. Our results show that SNMP is not efficient enough to be used as a switch control protocol, even though it is a common protocol supported in most ATM switches today. Moreover, results in transfer delay measurement have confirmed that background traffic of the same service type affects the distribution of packet transfer delay of the reserved reference flow. However, for the case having mixed service types, our results have also confirmed that GS (i.e. CBR service in ATM) has a higher
scheduling priority over CLS. This agrees with the IETF recommendation of Integrated Services model.
Chapter 4

VC-Merging Support

Last chapter has presented two innovative mechanisms, *alternative path forwarding* and *token forwarding*, together with a piping concept to support QoS under a distinctive style of IP multicast, i.e., heterogeneity. This chapter further discusses a more general multicast paradigm, which is the *anypoint-to-anypoint*\(^{11}\) (ap2ap) multicast. A crucial component of supporting QoS under this multicast paradigm is the mechanism of VC-merging. This chapter presents a novel approach to perform VC-merging. In the following sections, motivation and problem of VC-merging is discussed. Next, a summary reviews various proposals found in the literature and highlights their pros and cons. The section followed describes details of the proposed approach including the concept, architectural model and operational details. Section 4.4 reports the comprehensive analytical and simulation analysis of its performance. Section 4.5 discusses various issues of practical implementation.

\(^{11}\) is the union of point-to-multipoint, multipoint-to-point and multipoint-to-multipoint.
4.1 Motivation

Recently, various cut-through layer-3 (L-3) switching proposals have been developed by several organizations, notably the Ipsilon's IP switching[43], Cisco's Tag switching[44], Toshiba's CSR[45], IBM's ARIS[46], NBase Comm's DirectIP[47] and 3Com's Fast IP[48]. In spite of the differences in objective and implementation, the ISAC architecture proposed in the previous chapter shares a similar concept: the L-3 flow information is mapped onto L-2 label\(^{12}\) so that L-2 switches (such as ATM switches) can perform data forwarding very quickly that avoids the complicated multi-field (MF) classification and header processing. Several methods of such mapping exist, but research results [49] indicate that a typical Internet backbone requires a L-2 label space as large as 64K to perform well. As the volume of Internet traffic continues to grow exponentially, it is clear that L-2 switches will soon run out of label space. One scalable solution to resolve this problem is to introduce some form of flow aggregation [50], i.e. flows with certain characteristics are grouped together in L-3, or flow merging, i.e. a group of flows is forwarded using the same label in L-2.

Besides the scalability concern, L-2 switches also need merging capability to support IETF protocols. In RSVP [4], a receiver can request a wildcard filter (WF) specification within a session such that it can receive data from all participating senders. A brute-force approach to support this filter specification is to use separate L-2 label to identify each sender. However, this requires a large number of labels and becomes inefficient if there are a large number of senders in a session. A more promising approach is to merge senders' L-2 labels at the nodes that are shared by all senders so that fewer number of labels are required from those nodes onwards. Besides RSVP, MPLS also proposes a special option called Stream Merge for scalability reasons. In [51], an LSR (Label Switch Router) is called Stream Merge capable if it can receive two packets from different incoming interfaces having different labels, and send both packets out on the same

\(^{12}\) L-2 label refers to VPI/VCI in ATM.
outgoing interface having the same label. This in effect takes two incoming streams and merges them into one.

Furthermore, a merging capability is also useful in the native L-2 domain for supporting a more general style of multicast (such as ap2ap) efficiently. Recent studies [52] have indicated that the shared tree multicast approach performs better than other approaches, such as multicast server, sender-specific tree multicast, etc., in terms of bandwidth consumption, network resources required to maintain states, membership management complexity and operating system overhead. As most of the emerging IP multicast protocols such as PIM-SM, CBT, etc. will support the shared tree paradigm, it is definitely a concern for L-2 switches to be able to inter-operate with them. Particularly, the increasing deployment of ATM as a major backbone network infrastructure makes an imperative support of a more general multicast paradigm. The primary concern to support the shared tree multicast and the general anypoint-to-anypoint multicast in ATM is the capability of VC-merging.

VC-merging refers to the ability of mapping different incoming VC labels (VPI/VCI) for the same outgoing VC to a single VC label. Without a special arrangement, cells from different incoming VCs are indistinguishable at the output of a merge point. Therefore, cells belonging to different frames cannot be interleaved arbitrarily with each other, or else the receiver will not be able to re-assemble the frames assuming that AAL5 is used. Beside this cell interleaving concern, there is also a need to maintain the individual per-flow QoS when merging flows of cells.

It is worth mentioning here that cell interleaving is allowed in AAL3/4 as the MID field in each cell can be used to identify the sender uniquely. However, using AAL3/4 for this purpose has several drawbacks: (1) the size of MID field is too limited and inflexible; (2)

---

1 Without loss of generality, we assume AAL5 is used throughout this chapter.
the additional per-cell AAL overhead lowers the protocol efficiency even further; (3) its error checking capability is not comparable with AAL5.

Although merging can be implemented using VP connections (VPCs), it is not a favourable option because of the limited VPI space (only 12bits in NNI and 8bits in UNI). This means the maximum number of merged connections passing any merging port is limited to 4096 at any given time, not to mention that part of the VPI space may have been used for other purposes, such as the emerging Virtual Private Network (VPN) provisioning.

4.2 Related Work

This section discusses an overview of different VC-merging approaches reported in the literature. According to their characteristics, they can be generally classified into three major categories: (1) frame-level interleaving (FI) [53, 54]; (2) header extension (HE) [55, 56]; and (3) explicit notification (EN) [57].

4.2.1 Frame-level interleaving

In this approach, as illustrated in Figure 4.2.1, cells belonging to an AAL frame are always sent together without any interleaving from other frames. In SEAM [54], when the first cell of an AAL frame arrives at a merging node and if the outgoing VC is currently being used for sending cells of another frames, the incoming cells of the new frames are buffered. After the current frame is sent, frames from different inputs being buffered will be sent in a round robin fashion. This scheme uses the AUU bit of AALS in detecting the end of frame (EOF) transmission. In order to recover from the deadlock situation due to loss of an EOF, SEAM proposes a timeout mechanism and generates a dummy EOF cell. When the input ports differ in speeds, SEAM proposes a store-and-forward mode such that the whole frames from the slow inputs have to arrive before forwarding them onto the output. This avoids slow input ports from blocking the high-speed input ports.
It is clear that this approach requires additional\textsuperscript{14} buffer space per source, which depends on the length of AAL frames, to accommodate cells of all incomplete or blocking frames. This introduces delay jitter for all buffered traffics due to variable blocking duration. The delay is unpredictable as it depends on the variable frame size, arrival pattern and the rates of all other sources. This may lead to the QoS contract being violated or a conservative call admission policy. In addition, the output of this merger tends to produce bursty traffic that is not desirable at the downstream node.

\textbf{Figure 4.2.1} Frame-level interleaving approach to VC-merging

\subsubsection{4.2.2 Header extension}

This approach extends the cell header to accommodate a unique source identifier. As indicated in Figure 4.2.2, since cells from different sources are uniquely distinguishable, they can be multiplexed in any order into a common outgoing VC. As a compromise between AAL5 and AAL3/4, both proposals \cite{55,56} use 16 bits to carry the source ID. This leaves only 46 bytes for the cell payload.

The merits of this approach are: (1) it is simple to be implemented and (2) it needs only slight modifications to switch hardware and AAL. However, it has a major drawback of further reducing the protocol efficiency (or increasing the cell-tax) from 90.57\% (\frac{4}{3}) to 86.79\% (\frac{4}{3}).

\textsuperscript{14} In addition to the per-VC queues
4.2.3 Explicit notification

Similar to HE, this approach identifies each cell using the source ID (SID) in order to avoid cell being mixed-up at the destination. However, instead of carrying the SID in each cell, CRAM [57] proposes to carry the multiplexing information externally in a RM cell\(^\text{15}\). When cells going to the same outgoing VC arrive, the output module will buffer all cells and construct a RM cell which contains all SIDs of the buffered cells. The order of SIDs inside the RM cell corresponds to the order of the cells at the outgoing VC forwarded by the merging node. After 24 cells\(^{16}\) (or less due to EOF) have been received, the RM cell is put onto the outgoing interface and then followed by the buffered cells. Figure 4.2.3 illustrates a simple example of this approach.

If an EOF cell is received while constructing a RM cell, the RM cell is shipped immediately without waiting for the whole RM cell to be filled up. Although this eliminates delay due to buffering, the overall protocol efficiency is reduced. To address the deadlock situation where the next cell never arrives, CRAM proposes a timeout.

\(^{15}\) RM cell is defined in the context of ATM ABR service. It is used to convey bandwidth availability, state of congestion, and traffic management function.

\(^{16}\) the maximum number of SIDs that a RM cell can accommodate.
mechanism in constructing a RM cell. The RM cell and the buffered cells are sent immediately when the timer expires.

Like HE, this approach allows cell-level interleaving and is relatively simple to be implemented. It requires only slight changes to the switch hardware and a minimal additional\(^\text{17}\) buffer of 24 cells at the output module. However, there are several disadvantages: (1) it has a protocol efficiency of \(\frac{48k}{53(k+1)}\), \(2 \leq k \leq 24\). This is between 86.94% and 60.38% depending on the occurrence of EOF and is much lower than all other approaches mentioned above; (2) the additional delay introduced varies depending on the incoming traffic. This is because the buffered cells are not sent until the RM cell is filled up or an EOF cell is received. This variation makes QoS contract difficult to guarantee.

\(^{17}\) In addition to the per-VC queues
4.3 VC-Merge Capable Scheduling

4.3.1 Concept of implicit notification

Our proposed approach belongs to a new category, namely *Implicit Notification* (IN). It requires neither extension to the cell header nor explicit control cell to identify cells from the interleaved cell stream. IN shares the same goal as HE and EN to enable native cut-through forwarding and cell-level interleaving without increasing per-cell overhead.

In a typical connection-oriented network, resources reserved for all VCs are usually known at all participating nodes, no matter what signalling protocol is used. Therefore, the merging node and its downstream nodes should know which and how many VCs are merged upstream. If the merging order is enforced and made available to all downstream nodes, incoming cells from different source VCs can then be uniquely identified. Since the merging order is known prior to the de-merging node, no explicit notification is required. However, this approach requires the states at both merging and de-merging nodes to be locally synchronised. The following sections describe how this synchronisation can be achieved.

4.3.2 Merge-capable scheduler

The proposed merge-capable scheduler attempts to preserve the cell scheduling order of the merging VCs such that cells from different sources can be uniquely identified at the input module of its downstream node. The design resembles a hierarchical scheduler [58, 59] structure originally proposed for the link-sharing paradigm. Figure 4.3.1 depicts the design of the proposed scheduler. It outlines an output module of a typical memory-based output-queuing ATM switch. In a per-VC queuing design, each outgoing VC is allocated a queue, such that the cell transmission order among different queues is determined by a scheduler (SCH).
Where

- $i$: Number of non-merging VCs (physical queues)
- $j$: Number of merged VCs (virtual queues)
- $N_k$: Number of merging VCs (sub-queues)
- $Q_k$: Cell queue $k$
- $Q_{k,l}$: Cell sub-queue $l$ in virtual queue $k$
- $\Phi_k$: Service share (reservation) of queue $k$
- $\Phi_{k,l}$: Service share of sub-queue $l$ in virtual queue $k$
- $HOL_k$: Head Of Line indicator for queue $k$

Define

- $m$: Number of consecutive snooper cells
- $M$: Maximum number of consecutive snooper cells
- $\text{SnoopInd}$: Snooping indicator

**Figure 4.3.1 Merge-capable scheduler**

---

18 Native cut-through forwarding refers to the ability to switch an incoming cell immediately, assuming the outgoing VC is not transmitting another cell and the incoming cell is traffic compliant.
This design introduces a concept of virtual queue\(^{19}\) (VQ) and has a two-level hierarchical structure. A virtual queue consists of a sequencer (SEQ) and a set of sub-queues for different incoming VCs having the same outgoing (merged) VC. The SEQ is a weighted round robin (WRR) sequencer by which the polling sequence of sub-queues is determined according to the resource reservation (weights) of each sub-queue. Figure 4.3.2 illustrates a typical operation of a sequencer. The SEQ orders the transmission of its sub-queues according to their reservations.

![Normalized Sub-queue Reservations / Weights](image)

**Figure 4.3.2 An example of merging sequence**

The master scheduler, as represented by SCH in Figure 4.3.1, can have any type of scheduling policy, but fair queuing is preferable. According to the resource allocated to each queue (physical or virtual) and its scheduling policy, SCH selects a queue with the highest instantaneous priority to transmit. Once a virtual queue is selected, its corresponding SEQ will then determine which sub-queue to transmit.

Before discussing the detailed operation, we first introduce the concept of *snooping*. Snooping refers to a situation where a sub-queue being selected by SEQ has no cell to transmit. It happens when the resource reserved for this sub-queue is not fully utilised. In order to preserve the cell sequencing order at the output of SEQ, a snooping signal is generated to indicate an empty slot. Upon detecting a snooping signal, SCH will insert a snooper cell into the empty slot. A snooper cell is a cell selected from the next highest instantaneous priority queue and tagged with a special identifier at its header. We propose

---

\(^{19}\) In this paper, virtual queue and merged queue are used interchangeably.

67
to use the most significant bit (MSB) of the VPI field for this purpose. When the input module of the downstream node receives a snooper cell, it recognises there is an empty slot for that particular virtual queue. Then the input module will update the state of de-sequencer accordingly. Hence, cells from different sub-queues within a virtual queue can still be uniquely identified by the downstream node, even though they carry an identical VPI/VCI label.

4.3.3 Operational details

In ATM, before any cell can be transferred over the network, a connection setup procedure has to be completed. This usually involves signalling traffic parameters and reserving required resources at each node along the end-to-end virtual connection. After the connection setup phase, an additional operation is introduced to determine the cell sequencing of all merging nodes. Since connection information will be passed through all participating nodes, each node should be able to determine the number of merging and de-merging VCs. In addition, the merging and de-merging sequences of all virtual queues involved should also be determined. For a complete discussion on the signalling requirement, please refer to Section 4.5.2.

Figure 4.3.3 illustrates the operation of an output module of a merging node. In each scheduling instant, SCH selects one of the active queues having the highest instantaneous priority to transmit. This instantaneous priority is computed based on the policy of the scheduler and resource reservations of all queues. If an active physical (non-merging) queue is selected, its first cell in the line will be serviced. However, if an active virtual queue is selected, SEQ will review its current state and the pre-computed merging sequence to determine which of its sub-queues to take this turn. Then, after a cell has been delivered, the state of the SEQ is advanced.

When the input module of a downstream de-merging node receives a cell indicating a merged VC, the corresponding de-sequencer (de-SEQ) will determine which sub-queue the cell is originated, with reference to its current state. This identified cell will then be
put into an appropriate output queue and wait for delivery. Like the SEQ at the upstream node, every time a cell from a merged VC is received, its corresponding de-sequencer state is advanced. Operation of an input module of a de-merging node is summarised in Figure 4.3.4.

![Flowchart]

**Figure 4.3.3** Operation of the output module of a merging node

---

A queue is active if its Head-Of-Line (HOL) indicator is asserted.
### Table 4.3.1 Event decoding logic

<table>
<thead>
<tr>
<th>A merged VC</th>
<th>Snooper</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>False</td>
<td>False</td>
<td>A cell from a physical queue is received.</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>Snooping is asserted at a virtual queue.</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>A cell from a virtual queue is received.</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>Snooping is cleared at a virtual queue.</td>
</tr>
</tbody>
</table>

**Figure 4.3.4** Operation of the input module of a de-merging node
So far, we have assumed no snooping occurs. If the selected sub-queue of the virtual queue is empty, a snooping signal ($SnoopInd$) will be generated. The SCH will then insert a snooper cell into the empty transmission slot. A snooper cell is a cell from a physical queue with the next highest instantaneous priority and tagged with a special snooping identifier at the cell header. Upon receiving a snooper cell, the downstream de-merging node will examine its VCI/VPI. If the cell is from a non-merging VC, the snooper counter ($COUNT$) will be advanced and the cell will be put into an appropriate output queue according to the VC translation table. Otherwise, the corresponding de-SEQ state will be updated with a value indicated by $COUNT$. The received cell will then be treated as an ordinary cell coming from a merged VC. Table 4.3.1 summarises the logic of decoding an incoming cell. These decoded events trigger different sequences of actions described in Figure 4.3.4.

Since, in any instant, only one virtual queue is allowed to snoop, under a low utilisation situation, collision or blocking occurs when there are more than one merging queues contain empty sub-queues. It is because the current snooping queue will block all other queues from committing snooping until its snooping status is deserted. To avoid a situation where a particular merging queue monopolises snooping, we propose limiting the number of consecutive snoopers that a merging queue can take. Equation 4.3.1 defines the HOL indicator, which determines if a queue is active. Since only an active queue will receive a chance to transmit, Equation 4.3.1 ensures that only one merging queue can snoop and its maximum number of consecutive snoopers is limited by $M$.

\[
HOL_k = \begin{cases} 
1 & \text{if } C_k^0 \neq 0 \\
0 & \text{if } C_k^0 = 0 \\
1 & \text{if } (m_k \leq M) \bigcap (SnoopInd) \\
0 & \text{Otherwise}
\end{cases}
\]

for non-virtual queue

\[
HOL_k = \begin{cases} 
1 & \text{if } C_k^0 \neq 0 \\
0 & \text{if } C_k^0 = 0 \\
1 & \text{if } (m_k \leq M) \bigcap (SnoopInd) \\
0 & \text{Otherwise}
\end{cases}
\]

for virtual queue

where $m_k$ is the number of consecutive snooper cells for $k^{th}$ virtual queue;

$M$ is the maximum number of consecutive snooper cells;

$C_k^0$ is the head of line cell of $k^{th}$ queue; and,

$SnoopInd$ is the snooping signal

**Equation 4.3.1 Definition of HOL indicator**
4.4 Performance Evaluation

In this section, we discuss the performance of our proposed scheduler. We first study analytically issues regarding fairness and delay guarantee under the worst case situation. Depending on the choice of scheduling discipline, fairness and delay bounds among non-merging queues and merged queues (virtual queues) can be guaranteed by SCH. The focus here is on the fairness among sub-queues (i.e. merging flows or flows being merged) and the delay bound of a sub-queue under the overall scheduler.

Apart from the worst case performance, we have also conducted a simulation analysis on the performance in average sense. The analysis is based on an instance of the proposed scheduler, WFQ based VC-merge capable scheduler (VMWFQ). This VMWFQ is implemented using a modified version of ns-2 simulator [60]. The metrics for comparison are the average goodput and average queuing delay experienced by a cell.

4.4.1 Analytical analysis

4.4.1.1 Worst case fairness

In [61], the authors introduce a metric called Bit Worst case Fair Index (B-WFI) that generally applies to a hierarchical system. In this section, we will show that our scheduler also guarantees fairness in B-WFI sense.

Definition 4.1 [61]: A server node \( s \) is said to guarantee a Bit Worst-case Fair Index (B-WFI) of \( \alpha_{i,s} \) for a session \( i \), if during any time interval \([t_1, t_2]\) when the queue is continuously backlogged, the following holds

\[
W_i(t_1, t_2) \geq \frac{\phi_i}{\phi_s} W_s(t_1, t_2) - \alpha_{i,s}
\]

where \( \frac{\phi_i}{\phi_s} \) is the service share guaranteed to queue \( i \) by server \( s \); and.

\[
W_i(t_1, t_2) \text{ is the amount of session } i \text{ traffic serviced in } [t_1, t_2]
\]
To show that SEQ can guarantee a B-WFI, we consider the SEQ alone, i.e. 
\[ \phi_s = 1 \quad \therefore \quad W_s(t_1, t_2) = r(t_2 - t_1). \] 
Therefore, 
\[ W_s(t_1, t_2) \geq r_s(t_2 - t_1) - \alpha_s. \]

Note that, 
\[ W_s(a^k_t, d^k_t) = Q_s(a^k_t) \]

where \( Q_s(\tau) \) is the number of bits in the session queue at time \( \tau \); \( a^k_t \) & \( d^k_t \) are the arrival and departure time of \( k^{th} \) packet of session \( i \), respectively.

Putting \( t_1 = a^k_t \) & \( t_2 = d^k_t \), \( Q_s(a^k_t) \geq r_s(d^k_t - a^k_t) - \alpha_s \)

\[ \Rightarrow \quad d^k_t - a^k_t \leq \frac{Q_s(a^k_t)}{r_s} + \frac{\alpha_s}{r_s} \]

That is, \( \frac{\alpha_s}{r_s} \) is the worst case waiting time \( \beta_i \). It represents the maximum time a packet coming to an empty session queue needs to wait before receiving its guaranteed service rate.

For a virtual queue \( k \) contains \( N_k \) sub-queues and is allocated a guaranteed rate of \( R_k \),

\[ \beta_i = T + \frac{N_k L}{(1 - \phi_s) R_k} \]

where \( T \) is the maximum inactive time for any sub-queue and \( L \) is the packet length.

Hence, SEQ for virtual queue \( k \) also guarantees a B-WFI for its sessions.

**Theorem 4.1**: For any sub-queue \( i \) under a VC-merge capable scheduler (VMSCH), the following holds,

\[ \alpha_{i,VMSCH} = \alpha_i + \frac{\phi_s}{\phi_{SEQ(i)}} \alpha_{SEQ(i)} \]

where \( \alpha_{i,VMSCH} \) is the B-WFI for sub-queue \( i \) and

\[ \alpha_{SEQ(i)} \] is the B-WFI for the virtual queue containing sub-queue \( i \).

---

21 defined as the data rate received at the end node.
The proof is referenced to [61] and is given in the Appendix A. Basically, Theorem 4.1 states that the WFI for the proposed scheduler guaranteed to a sub-queue is the sum of WFI for a sub-queue and WFI for a virtual queue weighted by the reservations of the sub-queue and its virtual queue. In other words, if SCH is fair in the B-WFI sense, the overall scheduler (VMSCH) will guarantee fairness in B-WFI sense.

4.4.1.2 Worst case delay bound

Before we proceed to establish a delay bound for VMSCH, we first state the relationship between the delay bound and guaranteed service burstiness index. In [62], the author proposed the following definition.

**Definition 4.2 [62]:** A server node $s$ is said to guarantee a service burstiness index (SBI) of $\gamma_{i,s}$ to session $i$ if for any time instant $t_2$ when session $i$ is backlogged, there exists another time instant $t_1$ within the same server busy period of $t_2$, where $t_1 < t_2$, $Q_i(t_1^-) = 0$, and $Q_i(t_1^+) \neq 0$ hold, such that,

$$W_i(t_1, t_2) \geq \frac{\phi_i}{\phi_s} W_s(t_1, t_2) - \gamma_{i,s}$$

where $\frac{\phi_i}{\phi_s}$ is the service share guaranteed to queue $i$ by server $s$; and

$Q_i(t)$ is the number of bits in the session queue at time $t$.

Note that if a session is guaranteed a B-WFI, it is also guaranteed a SBI. It has been shown in [61] with the following lemma that guaranteed SBI and guaranteed delay bound to a leaky bucket constrained session are related.

**Lemma 4.1 [61]:** Consider session $i$ that is leaky bucket constrained by $(\sigma_i, r_i)$. If a server guarantees a SBI of $\gamma_{i,s}$ to the session, it can guarantee a delay bound of

$$\frac{\sigma_i + \gamma_{i,s}}{r_i}.$$
Theorem 4.2: For a sub-queue $i$ under VMSCH, if it is constrained by a leaky bucket $(\sigma_i, r_i)$, then the delay of any cell in the sub-queue is bounded by $D_{i,VMSCH}$,

$$D_{i,VMSCH} = D_{i,SEQ} + \frac{\alpha_{SEQ}(i)}{r_{SEQ}(i)} \frac{\phi_{SEQ}(i)}{\phi_i}$$

where $\alpha_{SEQ}$ is the B-WFI for the virtual queue of sub-queue $i$; and $r_{SEQ}(i)$ is the guaranteed rate for sub-queue $i$ from its virtual queue.

The proof of this theorem is given in the Appendix A. Theorem 4.2 states that the delay bound provided by the proposed scheduler to sub-queue $i$ is the sum of the delay bound provided by SEQ to sub-queue $i$ and the WFI weighted by the guaranteed rate for the sub-queue $i$ from its virtual queue. Since we have shown that SEQ guarantees delay bound, if the SCH can also guarantee a bound on delay, the overall VMSCH will offer a guaranteed delay bound.

4.4.2 Simulation analysis

This section presents performance evaluation of the proposed scheduler in terms of the average queuing delay and fairness in the average sense. The subject of this analysis is the VMWFQ scheduler, which has a WFQ embedded as the master SCH. This VMWFQ is implemented using a modified ns-2 simulator. As depicted in Figure 4.4.1, the simulation scenario contains 100 sources and 20 sink nodes connected through two nodes implementing VMWFQ. All sources are exponentially distributed ON-OFF source with a mean burst time of $2.12\text{ms}$, a mean idle time of $0.70667\text{ms}$ and a randomly generated peak rate. Each flow is leaky-bucket regulated according to its assigned reservation. A reservation includes a fixed bucket depth of 5 cells and token rate randomly generated between 0.1Mbps to 0.5Mbps for merging flows or between 0.1Mbps to 4Mbps for non-merging flows. Flows are divided into 5 groups. Each flow within a group terminates with one of the sink nodes. Group 5 contains all non-merging flows whereas all other groups contain flows that are merged according to the merging patterns as shown in
Table 4.4.1. Each merged flow (virtual queue) is assigned a reservation equal to the sum of all merging flow (sub-queue) reservations. For ease of comparison, we have chosen six of the 100 flows as references. While all of these reference flows have the same reservation, they are selected from different groups and merged flows.

Table 4.4.1 Flow arrangement

Furthermore, we have performed another set of simulations, which implemented a non-merging WFQ scheduler in the intermediate nodes. Since flows in this scenario are treated independently or non-merging, results obtained here are performance bounds in comparison with the merging scenario.

22 Unless otherwise specified, a uniform distribution is assumed.
Various loading conditions have been simulated. These conditions are constructed by varying the resource utilisation of the sources. For example, in the overloading situation, all sources transmit more than their resource reservation. All simulation experiments are 50 seconds long. For the fairness analysis, the average goodput is estimated by taking the average of data received in every 0.5 second interval and then averaging over 100 intervals. We observed that most of the estimated averages have a 90%-confidence level of the order, which is 3 times smaller than the estimated average value. It implies that the variation of goodput is rather small. For the delay analysis, the queuing delay for each cell of the reference flows is calculated and averaged over the entire simulation duration. The 90%-confidence levels of the calculated average are generally found to be two-order of magnitude smaller than the average value.

### Fairness

Figure 4.4.2 shows the fairness of VMWFQ in terms of the average goodput under two loading conditions, while Figure 4.4.3 & Figure 4.4.4 illustrate the effect on goodput as the loading increases. It can be seen from Figure 4.4.3 that goodput decreases as the loading increases beyond 90%. It is because some cells are being dropped when their queues overflow. Although cell dropping occurs in overloaded situation, Figure 4.4.2(b) confirms that VMWFQ maintains fairness among all reference flows. It is because the average goodputs of all reference flows, which have requested the same level of resource, are comparable especially in the over-utilized case. Although under the lightly loaded condition or under-utilized condition, the variation of average goodputs is larger. Figure 4.4.2(a) shows that VMWFQ operates at a similar performance as non-merging WFQ.

Moreover, for the merging flow under the lightly loaded condition. Figure 4.4.4(a) shows that VMWFQ with $M = 3$ can give a better performance to the merging flow than the one with smaller value of $M$. It is because a small value of $M$ restricts virtual queues, which contain empty sub-queues, from being active until their number of consecutive snoopings required is less than or equal to $M$. This essentially blocks most of the VQs when utilisation of reserved resource is low. However, further increasing $M$ shows only a negligible improvement on goodput.
Figure 4.4.2 Average goodput of different reference flows under two loading conditions
Figure 4.4.3 Effect on average goodput under different loading conditions
Figure 4.4.4 Effect on average goodput under light loading conditions
4.4.2.2 Average delay

Figure 4.4.5 and Figure 4.4.6 depict the average delay for different reference flows under various loading conditions. For ease of comparison, the figures plot the average delay in terms of the percentage deviation from the non-merging WFQ scheduler. Under a lightly loaded condition, non-merging WFQ gives a smaller average delay than VMWFQ as Figure 4.4.5(a) shown a positive deviation. It is because snooping in VMWFQ introduces additional queuing delay. From Figure 4.4.5(a) & Figure 4.4.6(a), we notice that VMWFQ with $M = 3$ or $4$ has a smaller deviation than $M = 1$ or $2$. Since $M$ limits the number of consecutive snooping, if $M$ is small, most virtual queues will be blocked until most of their sub-queues are backlogged. It eventually introduces longer delay than the case with a larger value of $M$. However, when the nodes are heavily loaded, the advantage of non-merging WFQ is not noticeable. Figure 4.4.5(b) shows that VMWFQ performs comparably with non-merging WFQ (i.e. significantly smaller deviation). It is because when the node is heavily loaded, all queues are persistently backlogged. Since it is unlikely that any snooping would occur, non-merging WFQ and VMWFQ operate in the similar manner thereby resulting in a similar average delay.

To see the effect of snooping, Figure 4.4.7 plots the snooping rate\textsuperscript{23} under different levels of loading. It verifies that no snooping occurs when the node is overloaded or sources have fully consumed their allocated resources. However, occasional snooping can be observed when loading is close to 100%. It is due to the mismatch in cell arrival patterns of the sub-queues. It is observed that the snooping rate increases when the value of $M$ becomes larger. This increment implies that a larger value of $M$ allows more virtual queues to snoop. However, further increasing $M$ stabilises snooping rate because of the higher probability of snooping collision. Considering the loading effect, it is noticed that the increase in snooping rate is saturated when the loading is reduced to a certain level. Even if reducing load (decreasing utilisation) creates more empty sub-queues, the overall snooping rate cannot increase any further if there is no more cell from other queues available for snooping.

\textsuperscript{23} Defined as the total number of snooping occurs / total number of packet received.
Figure 4.4.5 Average delay for different reference flows under two loading conditions
Figure 4.4.6 Effect on average delay under different loading conditions
In summary, the simulation results demonstrate that VMWFQ can fairly distribute the available resources according to the reservations. It also gives an average delay comparable with a non-merging WFQ. Particularly, VMWFQ with $M = 3$ or 4 works better than $M = 1$. However, since $M = 4$ introduces more snooper cells, we argue that $M = 3$ is a good compromise and can be used as a rule of thumb to implement VMWFQ in practice.
4.5 Practical Issues

This section discusses issues that are related to the proposed merge capable scheduler in supporting VC-merging. These issues include signalling, de-merging modes, operation under special conditions and implementation complexity.

4.5.1 By-pass modes

Previous sections have assumed that the immediate downstream node completely segregates the merged flow. However, in some situations, the immediate downstream node may not choose to perform a full de-merging. Moreover, if the merged flow is not completely de-merged, the number of per-destination queues at the output module can be reduced. This section discusses this by-pass operation in which the downstream node does not segregate (complete by-pass) or only partially segregates (partial by-pass) the incoming merged flow.

In order to support these by-pass modes, operations at the input and output modules need to be slightly modified. Figure 4.5.1 illustrates the modified input module. If the received cell indicates a by-passing VC, instead of updating the state of the corresponding VQ, the input module inserts dummy cells into the corresponding virtual queue. Therefore, at the output module, when a virtual queue detects a dummy cell, it handles it in the same way as an empty slot. Figure 4.5.2 describes the modified operation at the output module. It is noted that both complete and partial by-pass can be handled in a similar fashion. At the output module, while there is only a single queue within the virtual queue for the complete by-pass case, several queues (each of which accommodates a portion of the de-merged flows) are required if the virtual queue is operating in a partial by-pass mode.
Figure 4.5.1  Modified operation of input module of a de-merging node
SCH selects a queue, $Q$

$Q = \text{virtual} \, ?$

Y

SCH selects a Sub-queue, $q$

Advance SEQ($Q$) state

$q = \text{empty or } q = \text{dummy} \, ?$

Y

Set $\text{SnoopInd} = \text{ON}$

SCH selects another physical queue, $Q'$

N

Get first cell from $Q$

Get first cell from $q$

$\text{SnoopInd} = \text{ON} \, ?$

N

Send cell

Y

Clear $\text{SnoopInd}$

Tag cell as snooper

Get first cell from $Q'$

Figure 4.5.2 Modified operation of output module of a merging node
4.5.2 Signalling requirements

To support the general ap2ap connection, the signalling protocol needs to handle two major operations: receiver join/leave and sender join/leave, regardless of its reservation style\(^{24}\). This section describes the signalling requirements for the proposed scheduler in supporting ap2ap connection using VC-merging.

It should be emphasised that ATM, in its current form, supports only point-to-point and point-to-multipoint connection. UNI 4.0 [63] defines two modes of operation associated with receiver (leaf-initiated) join (LIJ) capability: leaf-prompted and root-prompted join. Since multiple root nodes are not currently considered in ATM, no signalling operation has been defined so far for the sender join/leave scenario. Being a receiver-initiated reservation protocol, RSVP not only supports receiver join/leave operation, but it also addresses sender join/leave in terms of filter specification.

(A) Receiver join/leave

Figure 4.5.3 shows a typical receiver join/leave scenario. For the ATM UNI case, when a receiver (leaf node) initiates a join by sending a LEAF SETUP REQUEST message to the network, the network will locate the existing connection tree and forward the request. Once the request is approved, an ADD PARTY message will be sent to the joining receivers. Here we introduce a special information element for the ADD PARTY message to carry the merging information, such that the downstream node will be able to perform de-merging operation. Similarly, for the RSVP case, a special merging information object can be defined and carried by the PATH message to signal the downstream node. It should be noted that the proposed VC-merging approach does not require any modification in signalling mechanism for the case of receiver join/leave in ATM or RSVP. It requires only an extension to the existing signalling messages for carrying the merging information.

---

\(^{24}\) Reservation style refers to sender-initiated (e.g., ATM) and receiver-initiated (e.g. RSVP) style.
(B) Sender join/leave

As indicated earlier, RSVP has been designed for handling sender join/leave operation. Therefore, only a minimal extension, which is an information object in the PATH message to carry the merging information, is required. When a new sender joins an existing connection, the PATH message, which carries the updated merging information, will be forwarded to the downstream nodes.

Nevertheless, since native ATM signalling does not currently support multicast connection with multiple senders, a new set of signalling operation is required. Figure 4.5.4 illustrates the proposed extension to UNI 4.0 for supporting sender-initiated join/leave. When a new sender intends to join a connection, it sends a ROOT SETUP REQUEST message to the network. The network node will then pass a SETUP message, which carries the new sender and updated merging information, along the connection to the receivers. The receiver responds with a SELECT PARTY message, which signals the upstream nodes a set of interested senders. When the SELECT PARTY message reaches the originated network node and the admission check along the reverse path succeeds, the network node will issue an ADD PARTY to the sender for acknowledgement. Whenever a new sender is added to an existing connection, the proposed signalling operation will ensure the merging and de-merging node have the common view of all merging VCs.
4.5.3 Pathological conditions

In this section, we discuss some pathological cases that may affect the operation of the proposed scheduler.

(A) No non-merging cell to snoop

In general, there should be merging and non-merging VCs in any node. In a very unusual situation where all VCs are being merged, i.e., there is no non-merging cell. Under this situation, VMSCH can insert control cells\(^{25}\) as snooper cells to perform snooping.

(B) Some merging queues are persistently inactive

When a merging VC is malfunctioned or idle for a while, the corresponding merging sub-queue will be persistently inactive. Such inactive sub-queue will cause its virtual queue to commit snooping persistently. To overcome this

\(^{25}\) in the format of a special RM cell
situation, we suggest a timeout mechanism to disable the inactive queue. When a pre-defined timer expires, a control cell is sent to the downstream node such that both merging node and de-merging node can update their pre-computed cell sequencing order. Similarly, when the disabled VC becomes active, a control cell can also be used to trigger re-calculation of cell sequencing order at the downstream node.

(C) **Subject to data error**
Since the proposed scheduler relies on local synchronisation between two directly connected nodes, error in transmission may cause misinterpretation of merging cells which in turn may lead to loss of synchronisation. It should be noticed that cell dropping or error in cell payload does not affect the operation of VMSCH. Problem occurs only when the cell header is corrupted. We argue that this situation rarely happens in today’s wire-lined network because of the highly reliable underling optical or SONET link.

In case a header error is detected, a control cell can be sent upstream to request for the current states of all sequencers. Moreover, to avoid error accumulation due to undetectable errors in the cell header, a node can also send control cell(s) carrying current states of all sequencers to its downstream node periodically. Upon receiving the current sequencer states, synchronisation can be re-established.
4.5.4 Implementation complexity

This section discusses the complexity of implementing the proposed scheduler. It is not intended to give a detailed hardware design, but rather discuss the additional complexity in comparison with a common switch design. Common switch architecture today consists of an input module, switching fabric and output module, as depicted in Figure 4.5.5. The proposed VC-merging approach involves only a minimal modification in the output and input module.

![A simplified architecture of a common switch](image)

With reference to Figure 4.3.1, the design of the output module comprises a collection of cell queues and a master scheduler that can also be found in any common switch design. The components introduced include the control logic for snooping operation and a set of per-VQ sequencers. Both snooping control and sequencer can be easily implemented in hardware or software. Particularly, a sequencer can be easily implemented using a set of binary shift registers [64].

For the input module, the required modification is also minimal. As indicated in Figure 4.3.4, the operation involves mainly the components for detecting snooper cells, de-sequencer logic and VC routing table. It should be noticed that the VC routing table and its associated control logic are common in most input module. Therefore, the remaining design work is to integrate the de-sequencer with the routing table. One simple and obvious approach is to extend the routing table to include the states of all merged VCs. When the VC table is accessed, the state of the merged VC can also be updated simultaneously.
4.6 Summary

In this chapter, we have proposed a novel approach to address VC-merging particularly for ATM network. It makes use of a merge-capable scheduler to perform flow merging. The proposed VC-merging approach has the following characteristics: (1) it incurs no additional protocol overhead such that the protocol efficiency of AAL5 is maintained; (2) there is no extra buffer requirement and thus it only introduces a minimal extra delay due to snooping collision; (3) it produces less bursty output stream as cell-level interleaving is allowed; (4) it can deliver a guaranteed worst case fairness and delay bound. This can help provide per-flow QoS. Although average queuing delay is slight increased when reservations are under-utilised, it generally can give a comparable performance with its corresponding non-merging counterpart. However, the non-merging scheduler usually requires more resources; and (5) the proposed scheduler is not only simple to be implemented in either software or hardware, but it can also be easily adapted to any well-known scheduling algorithm. Besides its application in ATM, this concept can also be adapted to any packet network in which flows of packets are required to merge. Table 4.6.1 summarises and compares qualitatively the proposed approach with other approaches reported in the literature.

<table>
<thead>
<tr>
<th></th>
<th>FI</th>
<th>HE</th>
<th>EN</th>
<th>IN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Additional buffering</td>
<td>∞ packet length</td>
<td>No</td>
<td>24 cells</td>
<td>No</td>
</tr>
<tr>
<td>Cut-through forwarding</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>QoS guarantee</td>
<td>Poor</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
</tr>
<tr>
<td>Protocol efficiency</td>
<td>Good</td>
<td>Poor</td>
<td>Poor</td>
<td>Good</td>
</tr>
<tr>
<td>Output burstiness</td>
<td>Extreme</td>
<td>Minor</td>
<td>Medium</td>
<td>Minor</td>
</tr>
<tr>
<td>Hardware modification</td>
<td>Medium</td>
<td>Minimal</td>
<td>Minimal</td>
<td>Medium</td>
</tr>
</tbody>
</table>

Table 4.6.1 Summary of various merging proposals

In summary, the proposed VC-merging approach provides a scalable solution for using L-2 label space. It is particularly useful in the concept of cut-through L-3 switching including ISAC. With the proposed extension to existing signalling protocols, this VC-merging approach can also be used to support QoS over a more general class of multicast - anypoint-to-anypoint.
Chapter 5

Feedback Controlled Differentiated Services

In previous chapters, we have discussed how Int-Serv QoS model can be effectively supported and particularly, proposed several mechanisms, to address issues of supporting it over heterogeneous and anypoint-to-anypoint multicast connections. From this chapter onwards, our discussions will be focused on the another QoS model of the Internet: Differentiated Service (DS).

DS has been designed for scalability through handling aggregates of traffic instead of individual flows as in the Int-Serv. However, we have observed that the DS mechanism in some situations can hardly achieve the desired QoS and may result in unfair conditions. To remedy these problems, this chapter proposes an extension to the DS framework. It describes a general feedback controlled DS paradigm that enables network providers to impose control mechanisms upon their network domains.
5.1 Introduction

As mentioned briefly in Chapter 2, DS provides different levels of network services by employing a set of well-defined building blocks. Its key mechanism is to use a small label (known as DS Code-Point or DSCP) in the IPv4 (TOS octet) or IPv6 (Traffic Class octet) packet header to determine a packet to receive a particular forwarding treatment (Per-Hop Behaviour or PHB) at each network node. At the DS domain boundary, routers enforce the service level agreements (SLAs) by including functions such as traffic conditioning, monitoring and classification, in addition to providing the PHB requirements. References [3, 11, 65] depict the architecture, framework and boundary requirement of the DS network, respectively.

As depicted in Figure 5.1.1, a DS domain is a contiguous set of DS nodes, which operate with a common service provisioning policy and a set of PHB groups implemented on each node. A DS domain has a well-defined boundary consisting of DS boundary nodes as shown in Figure 5.1.2, which primarily condition ingress traffic. These boundary nodes ensure that packets entering a domain are marked appropriately to select a PHB from one of the PHB groups supported within the domain. Inside a boundary node, there is a traffic conditioner (TC) as illustrated in Figure 5.1.3. TC selects a particular traffic stream by using a classifier. Then, a meter measures the stream against the traffic profile. Finally, the state of this meter with respect to a particular packet will affect a marking, dropping or shaping action.

Furthermore, in order to support a variety of services, the boundary routers also include certain functionality: (1) a RSVP or control interface which is used to provision DS operating parameters, e.g. PHB and TC configuration parameters; (2) an admission control component which verifies that these parameters are not in conflict with other configuration parameters and the router's physical constraints; (3) a management interface for the network administrator to interact with the DS provisioning interface [66]; and (4) a monitoring interface that enables collection of statistics regarding traffic.
carried at various DS service levels. These statistics are important for accounting purposes and for tracking compliance to service agreements.

Similar to the boundary nodes, nodes within the DS domain (interior nodes) select the forwarding behaviour for packets based on their DSCP. They map the DSCP to one of the supported PHBs using either the recommended codepoint (PHB mapping) or a locally customised mapping [67].

Service providers combine PHB implementations, traffic conditioners, provisioning strategies and billing models to offer services to their customers. Providers and customers negotiate SLA with respect to the service to be provided at each customer/provider boundary. A SLA includes many aspects of a service, e.g., payment terms, legal terms, etc. One important technical aspect in SLA is the technical specification of the service (SLS). The SLS specifies service characteristics such as those mentioned in Chapter 2 and the traffic conditioning specification, which further specifies detailed service parameters for each service level.

Figure 5.1.1 Diff-Serv Architecture model
BAC: BW aggregate classification; or
MAC: microflow classification

**Figure 5.1.2 Architecture of a Boundary node [68]**

**Figure 5.1.3 Logical view of Traffic Conditioner [68]**
PHBs are the heart of the DS architecture and the building blocks for services. They describe the relative observable traffic characteristics across the node as well as the resources priority relative to other PHBs. A collection of PHBs can form a PHB group. Basic guidelines for specifying a PHB group are given in [3]. As of the date of writing, there are two PHB or PHB groups being defined: Expedited Forwarding (EF) [68] and Assured Forwarding (AF) [69].

Ref. [68] defines EF PHB as a forwarding treatment for the DS aggregate where the departure rate of the aggregate's packets for any DS node must equal or exceed a configurable rate. The EF PHB can be used to build a low loss, low latency, low jitter, guaranteed bandwidth, end-to-end service through DS domains. Since loss, latency and jitter are all due to the queues traffic experiences while transiting the network, providing low loss, latency and jitter for some traffic aggregate means ensuring that the aggregate sees no (or very small) queues. Queues arise when (short-term) traffic arrival rate exceeds departure rate at some nodes. Thus a service that ensures no queue for some aggregate is equivalent to bounding rates such that, at every transit node, the aggregate's maximum arrival rate is less than the aggregate's minimum departure rate.

While there are EF PHB being defined for the high-end and the legacy BE (or default PHB) for the lowest-end of the service spectrum, the AF PHB group is intended to serve for services in between. AF PHB group provides forwarding of packets in $N$ independent AF classes. Within each AF class, a packet is assigned one of $M$ different levels of drop precedence. A packet that belongs to an AF class $i$ and has drop precedence $j$ is marked with the AF codepoint $AF_{ij}$, where $1 \leq i \leq N$ and $1 \leq j \leq M$. Currently, four classes ($N = 4$) with three levels of drop precedence in each class ($M = 3$) are defined for general use. More AF classes or levels of drop precedence can be defined for local use. Within an AF class, a DS node must ensure that packets with $AF_{xp}$ have a higher forwarding probability than $AF_{xq}$, when $p < q$. Since there is no requirement for the relationship among AF classes, a node can implement relationship among classes in terms of buffer space, scheduling priority, dropping priority or any network resources.
5.2 Motivation

5.2.1 Generalised forwarding model

Based on the DS architecture and framework, we can infer a forwarding model as shown in Figure 5.2.1. The model is composed of a PHB classifier, a collection of shared queues and an output scheduler. The PHB classifier directs incoming packet to one of the shared queues according to the DSCP at the packet header. The shared queue generally implements a PHB class, e.g., AF class. It contains at least one logical differential sub-queues, each of which is associated with a relative drop preference. Various schemes can be used to manage these shared queues. For example, the RIO queue proposed for Assured Service [13, 14] is merely a shared queue with two differential sub-queues (IN and OUT) implementing the RED queue management. Since packets inside a shared queue (in any sub-queue) cannot be re-ordered, relative dropping preferences are the only means to prioritise packets for different sub-queues. Nevertheless, shared queues are operated independently among one another. Therefore, besides assigning a simple dropping priority to each shared queue, a more sophisticated scheduler can be used to schedule the outputs of the shared queues. Several types of scheduling mechanisms may be employed, but the implementation must include some means to avoid one shared queue from monopolising the available resources. Candidate schedulers include rate-limited priority queuing, weighted round robin, weighted fair queuing, class-based queuing, etc. All of these mechanisms have their basic properties though different choices result in different ancillary behaviour such as jitter experienced by individual flows.
5.2.2 Problems and suggested solution

A salient feature of this forwarding model (or DS in general) is its scalability. It is achieved by handling aggregated traffic using one or a small number of shared queues within a node rather than a large number of per-flow queues, thereby simplifying the packet processing and storage, such as packet classification, signalling, queue management and scheduling. However, our results and also other recent research reports [70, 71] have shown that this forwarding model may result in an unfair and inefficient resources sharing, thereby failing to achieve the desired QoS. Table 5.2.1 summarises some potential problems under different situations.
<table>
<thead>
<tr>
<th>Conditions</th>
<th>Outcomes</th>
<th>Problems</th>
</tr>
</thead>
<tbody>
<tr>
<td>Excess bw available</td>
<td>- Aggressive and/or non-adaptive flows take most of excess bw</td>
<td>- Unfair bw share</td>
</tr>
<tr>
<td>Insufficient bw</td>
<td>- High profile flows will be hit first</td>
<td>- Unfair service degradation</td>
</tr>
<tr>
<td></td>
<td>- Aggressive and/or non-adaptive flows have advantage</td>
<td>- Some flows cannot achieve assured rates</td>
</tr>
<tr>
<td>Flows with different Round-trip-time</td>
<td>- Flows with short round-trip-time have advantage</td>
<td>- Some flows cannot achieve assured rates</td>
</tr>
<tr>
<td>Flows with different requested profiles</td>
<td>- Low profile flows have advantage</td>
<td>- Some flows cannot achieve assured rates</td>
</tr>
<tr>
<td>Congestion</td>
<td>- Congestion sustained when flows are not co-operative</td>
<td>- Inefficient use of bw</td>
</tr>
<tr>
<td></td>
<td>- Packets only dropped at congested link</td>
<td>- Larger delay &amp; jitter</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- More buffer space required</td>
</tr>
</tbody>
</table>

Table 5.2.1 Summary of problems with Diff-Serv mechanism

Generally, there are at least three major causes for the problem:

1. *No isolation of flow inside the core of the network:* When flows enter the core of the DS network, they are naturally aggregated and forwarded using one or a few number of PHBs according to their DSCP. In other words, flows are aggregated into one or a few number of shared queues, each of which is allocated a certain amount of forwarding resources in terms of scheduling or dropping priority. Since flows are indistinguishable (or intended not to be distinguished) within a shared queue, aggressive flows may deprive other flows of any available resource, thereby resulting in an unfair resource sharing.

2. *No dynamic control at the DS boundary:* in DS, boundary nodes are the only places to control incoming traffics. Once a flow is allowed to enter a DS domain, it is usually policed or conditioned at the ingress node according to its TCA. However, the conditioning function is done in a static manner, which does not respond to the network dynamics.

3. *Reliance only on transport protocol to react:* with the presence of non-adaptive flows (e.g., UDP flows), TCP flows generally receive poorer service than UDP.
flows. This is because TCP sources back off when their packets are dropped, whereas the UDP sources do not react to dropping of their packets. Although RTP/UDP may provide a certain degree of adaptivity, its granularity may not be suitable for network control purposes. Moreover, even for the case of all adaptive flows, recent work [72] has indicated that some modifications to TCP are required in order to achieve the desirable service differentiation. Furthermore, the control offered by all transport protocols today is end-to-end in nature, therefore it may not be able to cope with the rapidly changing dynamics of the individual network domain.

To remedy these problems, we review the QoS mechanisms listed in Section 2.3. From the list of provision mechanisms (Section 2.3.1), mechanisms such as network planning and call admission control (CAC), may help alleviate the problems by preventing the network from severe long-term congestion. However, we argue that they are only necessary but insufficient requirements. Since the problems are associated with the dynamics of the network load and capacity, it has been shown in the literature that static solutions, such as allocating more buffers, providing faster links or tightening the call admission policy, does not solve the problem.

From the list of control mechanisms (Section 2.3.2), apart from flow control and flow monitoring, most of the available mechanisms have been adopted in the DS paradigm. Although flow control does exist in the TCP/IP suite, we have indicated earlier that this type of end-to-end control protocol is not adequate under the DS environment. However, the link-by-link or router-to-router flow control is considered too costly and unsuitable for DS. Therefore, a good compromise between these two extremes is the domain based flow control mechanism. Moreover, since services defined in DS are usually in a per-domain basis, a domain-boundary-to-domain-boundary flow control mechanism is more suitable in this respect.

With these considerations, we propose a dynamic control mechanism in which the boundary routers periodically obtain information from the core of the network and use
this information to update their traffic conditioners. Since a more precise control on the incoming traffic can be achieved at the ingress node, a better resource sharing at the core of the network may be possible. By incorporating this dynamic control mechanism, network providers not only can handle traffic congestion more effectively, but they can also manage their traffic and resources more efficiently.

5.3 Proposed feedback control extension to Diff-Serv

It is generally believed that network vendors may prefer to deploy their proprietary control mechanisms according to their policy requirements. The proposed control framework, therefore, should be generic and flexible enough for this purpose. Moreover, it is desirable that the extended system should be backward compatible with existing DS mechanism for enabling interoperability.

Taken these requirements into consideration, this chapter defines a general Feedback- Controlled-Diff-Serv (FC-DS) in a way that a variety of dynamic control mechanisms can be derived from it. The concept of FC-DS is that the boundary routers periodically probe the core of the network to obtain the current state information. This network information is used by the ingress or boundary routers to update their traffic conditioners such that a more precise control on the incoming traffic can be achieved.

The following sections describe the proposed extensions of the architectural model and framework for constructing the FC-DS. For ease of understanding, they should be read together with [3, 11, 65].

5.3.1 Architectural extension

The FC-DS architecture is built based on the DS architecture. It is therefore a superset of the requirements and functionality defined in [3]. This section defines the additional architectural functions required to construct a feedback control paradigm.
5.3.1.1 FC-DS Domain

A FC-DS domain is a DS domain enhanced with a feedback control mechanism. It is possible that the control mechanism spans across multiple DS domains or within only one domain. However, this section considers only the intra-domain control mechanism while a brief discussion on inter-domain control is given in Section 5.3.3.2.

5.3.1.2 FC-DS Ingress node

An ingress node generally performs traffic conditioning functions to ensure that the traffic entering a DS domain conforms to the rules specified in the traffic conditioning agreement (TCA), in accordance with the domain's service provisioning policy. Since TCA is usually a static agreement, unless re-negotiation is allowed, the traffic profile derived from a TCA is fixed once a flow is admitted. FC-DS proposes to make this TC functions adapt to the state/feedback of the DS domain. The general rules for the adaptive TC (ATC) are:

(1) Under a normal situation, ATC performs the same TC functions as in the conventional DS;
(2) When excess resources are available inside its domain, ATC should modify its policing function such that traffic flow will have a fair share of the excess resource pool;
(3) When congestion occurs, ATC should ensure that each traffic flow will experience a fair service degradation. This can be achieved by tightening the traffic profile of individual flows in a fair manner;
(4) For some specific control purposes, ATC should be re-configured according to the returned explicit control commands; and
(5) All ATC functions should follow the dynamics of the DS domain under control.

Section 5.3.2.4 further discusses the components of an ATC.

Besides ATC, an ingress node is responsible for generating probes. A probe is a control packet that is used to collect network information from the DS domain under control. Depending on the control mechanism, the ingress node may send probe packets to its connected interior node(s) on a per-flow or per-boundary node basis. Finally, upon
receiving feedback (in terms of report) from the domain, an ingress node is also required to have the capability of consolidating report(s) before adjusting its ATC.

5.3.1.3 FC-DS Interior node

In addition to the basic packet forwarding function, an interior node is extended to include a load monitoring function. Upon receiving a probe/report packet, it updates the information carried in the probe/report with its current loading information and then forwards the control packet to other connected node(s). The monitoring function generally includes load measurement and estimation components.

5.3.1.4 FC-DS Egress node

For the case of intra-domain control, a FC-DS egress node is where the probe packets are terminated. The egress node is responsible for composing and returning report packets to the ingress node or other boundary node(s), in response to the received probe packet(s). Depending on the details of the TCA between two domains, egress nodes may perform traffic conditioning functions on traffic forwarded to the peer domain. For these cases, ATC functions may also be included in FC-DS egress nodes.

It is worth mentioning that the report generation mechanism is not only useful in the context of traffic control, but it also provides a hook for other purposes, such as receiver control [73] and QoS monitoring [74].

5.3.2 Framework extension

Having described the extensions of the architectural model, this section details the extended framework in regard to the configurations of the key control mechanisms.

5.3.2.1 Probing granularity

Generally, probing mechanism can be determined by two parameters: temporal resolution and spatial resolution. The temporal resolution refers to the probing period or how frequent a probe packet is generated. Typically, it can be specified in term of a time interval or packet count. The choice of a probing period is related to the dynamics of the
DS domain under control as well as the variation of the incoming traffic. To obtain a higher control precision, the ingress node may choose a shorter probing period, i.e. generate probe packets more frequently. However, this probing frequency should be balanced with the amount of processing power and overhead required at the network nodes.

The spatial resolution, however, refers to the granularity of network statistics. The following lists some possible examples:

- **Per-aggregated-flow or per-microflow basis**, in which one probe is generated per contracted incoming flow. It implies a *flow based control* mechanism, which can generally give the finest grain control precision.

- **Per-BA basis**, in which one probe is generated per behavioural aggregate (BA). If an ingress node has access to more than one DSCPs, multiple probe packets can be generated in each probing interval.

- **Per-egress-node or per-boundary-node basis**, in which one probe is generated per boundary node. Notice that the notion of ingress-egress-pair is defined only when there is a flow. Therefore, this probing scheme can be regarded as a *topology based control* mechanism in which each boundary (ingress) node keeps the statistics of all possible paths having other boundary nodes as egress points.

- A combination of the above. Depending on their control algorithm and required control precision, network providers may choose to have a variant or a combination of the above mentioned schemes.

In general, in choosing a probing granularity, one may consider (1) the required control precision, (2) the processing capability of the routers, and (3) the amount of tolerable control traffic overhead.

### 5.3.2.2 Handling control packet

Since probe/report (control) packets are sent on the same link as data traffic, an interior/egress node needs a mechanism to distinguish the control packets from other data packets. Several possible alternatives exist for constructing a control packet such that it can be easily identified. They include:
Creating a new packet with a special DSCP, in which control information is carried in the data area of this packet.

Extending the IP header of a selected data packet using IP header extensions, in which a special extension is defined for carrying the control information.

Creating a new RSVP packet with a special object, in which the control information is carried by the special object being defined.

After identifying a control packet, a node can process it using either an in-band or out-of-band approach. In the in-band approach, control packet clings together with data packets and therefore, receives the same level of forwarding treatment as other data packets. However, both data and control packets are subjected to being delayed or even dropped when the node is congested. This approach, however, is simple to be implemented. By examining the arrival of the control packets, one can also obtain a sample of the current congestion level of the data-forwarding path.

In the out-of-band approach, control packets receive special service, usually better than data traffic, at an interior node. It requires a special arrangement in the forwarding module of an interior node. However, for a control algorithm that is sensitive to the round-trip-time and integrity of the control packet, the out-of-band approach is more appropriate.

5.3.2.3 Control Information

This refers to the type of information carried in a control packet. The choice of type of information generally affects the capability of the ATC at the ingress node, and therefore, determines the controllability of the overall mechanism. The type of information can be categorised in terms of several attributes:

- **Type of indicator**

  This refers to what format of information is collected from interior nodes. It can be as simple as a binary flag which indicates congestion occurs, an instantaneous or average buffer level or measured load, or a more complicated measure of higher order statistics, e.g., buffer growth rate, rate of change of total load, etc.
- **Type of feedback**
This refers to what type of information is returned to the ingress node. It can be in terms of binary flag(s), explicit rate or a form of credit/token.

- **Granularity**
This refers not only to how coarse a measurement is done, e.g., per-PHB-class, per-PHB or per-port, but it also specifies how frequently a measurement is performed.

- **Directionality**
Direction here refers to when information is collected. Typically, information is collected in a forward direction where the control packet travels from an ingress node towards an egress node. In some cases, it can also be gathered in the reverse direction or even in both directions. However, it should be noticed that the forward and backward paths could be different depending on the routing protocol.

To select a type of information, network providers may consider the required controllability and processing capability of their network nodes. Some routers may already have the capability of monitoring their loading condition for other network management purposes. These loading statistics can be used as a form of network information for this purpose.

### 5.3.2.4 Adaptive traffic conditioning

Generally, the objectives of the adaptive traffic conditioning are to ensure that under any network loading conditions: (1) the traffic entering a DS domain conforms to the rules specified in the TCA; (2) the conditioned traffic will have "fair" share\(^\text{26}\) of the available resource inside a DS domain; (3) congestion can be effectively removed; and (4) resources within a domain are being utilised efficiently. Section 5.3.1.2 have described the general rules of the ATC functions. One way to realise these rules and objectives is to enhance the conventional TC with a *supplementary traffic profile*. Originally, the traffic profile is specified in a TCA and therefore is static in the sense that will not change over time or with network dynamics. The supplementary profile, however, is a profile derived

---

\(^{26}\) subjected to a particular definition of fairness
from the original one but will be updated according to the state of the domain under control.

As in conventional TC, the actions taken on out-of-supplementary-profile packets may include: (1) delaying those packets until they become in-of-supplementary-profile (i.e. shaping); (2) discarding (or dropping) those packets; or (3) re-marking the DS field of the packets to another codepoint. Since the supplementary traffic profile changes with the network dynamics, transient effects on these actions should be handled carefully. The following section discusses what actions can be taken in order to avoid impacts from these transient effects.

- **Dropping**
  A change of traffic profile will trigger a change of dropping threshold. For aggregated TCP flows, an abrupt change in dropping threshold may cause many packets to be dropped at the same time because TCP traffic is bursty in nature. Eventually, it may trigger all TCP sources to back off and results in a poor overall throughput. To remedy this *global synchronisation problem*, one should avoid this kind of "hard-limit" dropping.

- **Shaping**
  When the traffic profile changes, not only should the output rate of the shaper be adjusted, but the size of the shaping buffer should also be updated. Again, if the adjustment causes the shaping buffer to be overflowed, the problem of global synchronisation should be avoided.

- **Re-marking**
  A marker can adapt for the change of traffic profile in two possible alternatives:

  (1) Packets are promoted or demoted to other PHB within the same class. In general, this alternative does not have any side effect. However, since a PHB class is usually realised by a shared queue, re-directing packets to another logical sub-queue may not be able to alleviate congestion at this PHB class.

  (2) Packets are re-directed to another PHB class. Even though this can potentially help clear congestion at a PHB class, it should be noted that this might also cause packets to be re-ordered.
5.3.2.5 Control algorithm

In general, the control algorithm comprises two major components: *fair share computation* and *adaptation algorithm*. The fair share computation first calculates a target fair share value for the incoming traffic. The adaptation algorithm then computes a feedback quantity such that the ATC can enforce this target fair share at the ingress node.

Many algorithms are possible, but one can characterise and evaluate their performance by the following attributes:
- Fairness criteria: min-max fairness, proportional fairness or worst-case fairness
- Computational complexity: the amount of computation required and its relationship with the number of flows.
- Stability and convergence time: the time required reaching a target value, if possible.
- Robustness: sensitivity towards and capability of handling transient periods, errors in control information and other situations.

5.3.3 Other related considerations

5.3.3.1 Standardisation

Since our goal is to enable network providers to implement their own control mechanisms according to their applications and policies, the requirements to be standardised should be kept minimal. Therefore, only a few items are suggested here:

(1) Extended functional requirements for architectural components given in Section 5.3.1 as an informational RFC; and

(2) Probe/report (control packet) format as a standard-track RFC.

This includes only the common template of the control packet and one of the identification methods suggested in Section 5.3.2.2. If consensus is to use the out-of-band approach with a special DSCP, a DS codepoint assignment is required.
5.3.3.2 Inter-domain control

So far, we have assumed an intra-domain control mechanism. In some cases, an inter-domain control may be preferable. Usually, domains are operated by different network providers. To enforce a global control mechanism across multiple DS domains, several problems need to be resolved.

(1) **Policy conflicts**: different providers usually maintain their own policies in terms of management objectives, network provisioning, etc. To resolve any potential conflicts, we suggest that the TCA between two domain operators should be augmented with a *domain control agreement* (DCA). It specifies the configurations and parameters of the control mechanism being used.

(2) **Compatibility among different control mechanisms**: if control packets are not terminated at the boundary of one domain, the control algorithms and information models used in different domains need to be compatible or convertible. This mutation of control packet may also be specified in DCA.

(3) **Longer control packet round-trip-time**: since control packets need to traverse more than one domain, a longer round-trip control delay is unavoidable. The overall adaptation algorithm should take this into consideration.

5.3.3.3 Interoperability with non-FC-extended DS components

An non-feedback-control-capable node (non-FC-capable) is defined as a node which does not interpret control packets (probe / report) and / or implements only some or none of the functions mentioned in Section 5.3.1 and 5.3.2. Although details of the control mechanism may vary, generally all boundary nodes must be upgraded to feedback-control capable nodes in order to obtain a consistent domain control.

Inside the DS domain, the non-FC capable interior nodes are only required to maintain basic forwarding treatment for the control packet. However, it is desirable that they should have enough resources so that they will never become bottleneck points.
5.3.3.4 Multicast

The issue of multicast is still an active research topic in Diff-Serv WG. In order to control multicast traffic in the context of FC-DS, one of the fundamental requirements is to duplicate the probe packet at the point of divergence. At the ingress node, when multiple reports are returned from the leaf egress nodes of the multicast tree, an algorithm is required to consolidate the reports and derive a suitable set of ATC parameters. The simplest example of such algorithm is to choose the most conservative information from the received reports.

5.3.3.5 Security

We only discuss security issues in the context of the control mechanism. There are at least two issues of protection involved:

(1) Protection upon control packets: this mainly refers to the integrity and privacy of the information carried inside the control packet. A FC-DS node should always prevent any control packet from being intercepted, modified illegally or read without authorisation.

(2) Protection against forged control packet attack: a FC-DS boundary node should have a strategy to identify forged control packet and prevent its operation from being affected.

5.4 Possible instances and applications

FC-DS is a general feedback control paradigm over DS networks. Many instances of control mechanism can be derived under FC-DS. A straightforward one is the flow control mechanism which aims at providing a better QoS provisioning, fair resource sharing among users and more effective congestion control. All of these goals can be achieved by carefully controlling the incoming flows at the ingress nodes. Details of this mechanism are discussed in the next Chapter.

Apart from flow control, other control mechanisms can also be derived under FC-DS. The following sections outline some useful examples.
5.4.1 QoS monitoring for network management

Ref. [74] proposes a mechanism at the ingress node to monitor the performance of the data paths. The mechanism merely collects packet round-trip delay information by sending a bouncing packet from ingress node to egress node. Although the authors do not mention a specific usage of this mechanism, obviously, this can be useful at the ingress node to verify if a user is receiving what it has been contracted in the SLS.

Supporting this mechanism under FC-DS is a trivial task. Moreover, under FC-DS, more general applications in the context of network management can be supported. It should be noticed that the information collected from the network is not limited to the edge-to-edge round-trip delay. If more general per-node statistics are collected and made available to the network manager or bandwidth broker, more effective provision mechanisms such as resource allocation and call admission control can be achieved. By examining the collected statistics, network manager can also detect fault or network failure.

5.4.2 Receiver control

Receiver control is a mechanism that allows receiver/user to control its upstream or incoming traffic. In [73], authors describe an application that receiver can control the priority of its packets coming from the backbone network onto its access link. They argue that this mechanism is important particularly when the receiver's access link is a low bandwidth link, e.g. cellular IP hosts. This mechanism can also provide protection for the access link from certain types of denial-of-service attacks.

In fact, this mechanism is a special case of FC-DS paradigm where no control algorithm is required but the receiver provides an explicit control parameters or commands. In FC-DS, egress routers can configure the traffic conditioner at any ingress node by sending a specific report. Therefore, it not only can control the priority of the incoming packets through configuring the marking module, but it can also limit packet admission and impose delay to selected packets by configuring the dropping and shaping module inside the traffic conditioner, respectively.
5.5 Summary

In this chapter, we have presented a generalised forwarding model inferred from the DS framework. We have discussed the potential problems of this model. The major causes of these problems are: (1) there is no isolation of flow inside the core of the network; (2) there is no dynamic control at the DS boundary; and last but not least, (3) the overall system relies too much on transport protocol to react. To address these problems, this chapter has proposed an extension to DS that enables a feedback control mechanism to be implemented on a DS domain. With this feedback control mechanism, network providers can manage their traffic and network more effectively, thereby achieving a better resource sharing and more efficient resource utilisation. By presenting the necessary extensions to the original architectural model and framework, we have outlined the overall feedback controlled DS paradigm and briefly introduced some potential applications. To justify our recommendations, the next chapter will focus on an instance of this extension and discuss about its significance.
Chapter 6

An instance of Feedback Controlled Differentiated Services

In the last chapter, we have described the proposed extension of the DS architecture, namely FC-DS, to accommodate a general feedback control paradigm. This Chapter discusses an example mechanism that is derived from FC-DS. This mechanism serves as a flow control mechanism over DS networks. The objective of this chapter is twofold: (1) to illustrate how a flow control mechanism can be derived from the proposed FC-DS; (2) to show the significance of incorporating this flow control mechanism. This chapter begins by detailing the configurations of the mechanism which include probing/reporting mechanism, traffic conditioning, control algorithm as well as the overall operation. To analyse the network dynamics, the proposed system is subjected to comprehensive simulation studies. Finally, the trade-off of applying this control mechanism is discussed.
6.1 Related work

Undoubtedly, the idea of flow control or congestion control is not a new one. Many works have been done during the development of TCP protocol and the ABR service in ATM. Comprehensive surveys of congestion control can be found in [75, 76, 77].

Our proposed mechanism looks akin to ABR service in ATM. In fact, during the early stage of our design, we reviewed most of the works regarding to the development of the ABR service. Generally, our mechanism is different from the ABR work in the following aspects:

- It works under a network with variable size packets;
- The control algorithm and traffic conditioning operation are tailored for network layer. For example, the design of control packet format, route independence (connectionless) nature and unidirectional probing matches the characteristics of IP protocol;
- It complements the TCP dynamics and works gracefully side-by-side with TCP protocol; and last but not least,
- It is compatible with most of the DS architecture and framework. And more importantly, it matches the spirit of DS development in which the core of the DS network maintains no per-flow information.

Despite of their differences and similarities, we argue that the proposed mechanism is necessary in DS networks. Notice that ABR is merely a congestion control mechanism at the link-layer (L-2) (in particular, ATM) while TCP is a mechanism at the transport-layer (L-4). Our proposed control mechanism; however, is positioned at the network layer (L-3). While we realise that ATM will not be the only link-layer technology for supporting the future Internet, there are other technologies that integrate network-layer protocol well onto the physical layer, for example, IP/MPLS/PPP/SONET and IP/PPP/SONET. Based on our comprehensive analysis of the conventional DS framework, we acknowledge that if a network has only TCP flow control at the transport layer and admission control mechanism at network layer, it is not sufficient to achieve the desirable fairness of
resource sharing and efficiency. Under some circumstances, it cannot even offer the desirable QoS assurance or guarantee. Therefore, the introduction of a flow control mechanism in DS networks is justified for the numerous advantages it can offer.

6.2 System configurations

DS domains are usually operated by different network providers because network providers/operators usually maintain their own policies in terms of management objectives, provisioning mechanism, accounting, etc. Concerning this situation, this chapter proposes an intra-domain control mechanism in which control actions are operated within the domain and are terminated at the boundary of the domain under control.

6.2.1 Probing / reporting

Probing/reporting concerns how and what control information is signalled. It is illustrated as follows:

6.2.1.1 Control resolution

The probing/reporting mechanism can be described in two dimensions: temporal and spatial resolution. In DS, SLS is usually specified in a per-aggregated-flow granularity. Therefore, probe generation or spatial resolution of probing is chosen as per-aggregated-flow basis for ease of reference to SLS. For the temporal resolution, a time-based approach is chosen because it simplifies the necessary synchronisation between probing and load measurement process at the interior nodes. As illustrates in Figure 6.2.1, when the timer expires, the table of all admitted aggregated flows is examined. For each of those active aggregated-flows, a probe packet is created and delivered.
6.2.1.2 Information type

Probing information (Section 5.3.2.3) refers to the type of information being collected by the probe packet. The average measured load per PHB-class is chosen because it can reflect the state of a queue and it is also relatively easy to obtain. Average measured load refers to a measurement that is done over a time interval. This averaging operation smoothens out the variance of the load. An exponential averaging further correlates the average measured load with its value at the last measurement interval. The exponentially averaged measured load, $L_j$, for a PHB-class $j$ is therefore defined as:

$$L_j = wL_j' + (1 - w)L_j'$$

where $0 < w < 1$ is the averaging factor;

$L_j'$ is the averaged load at the last measurement interval;

and the instantaneous measured load $l_j$.

$$l_j = \frac{\text{Total number of bytes received in this interval}}{\text{Measurement Interval}}.$$

Figure 6.2.1 Probe generation operation
For the *reporting information*, an explicit rate indication first proposed in [78] is adopted for the reasons that:

(1) it can simplify the ATC at the ingress node;
(2) it can give a faster convergence time, i.e., the system reaches the optimal operating point quickly;
(3) it is more robust against errors in or loss of control packets. It is because the next control packet, which carries the correct feedback information, will bring the system back to the correct operating point in one step. It is particularly useful for the IP because of its connectionless nature; and
(4) the explicit rate feedback information can also be used by the end-system. For example, the video encoder at the sender can directly makes use of the rate value to tune its parameters.

6.2.1.3 **Control packet processing**

With regard to control packet identification and handling (c.f. Section 5.3.2.2), the out-of-band approach is chosen as it can give a shorter control round-trip-time and better integrity of control packets. We propose to assign a DSCP for the control traffic, namely, Control Forwarding PHB or CF-PHB. All packets marked with CF-DSCP shall receive better service over all other data packets.
6.2.1.4 Control packet format

Figure 6.2.2 depicts the proposed control packet format. Excluding the packet header, it is composed of two parts. The template part consists of information fields that are common to all possible mechanisms while the information objects part contains all vendor-specific information fields.

<table>
<thead>
<tr>
<th>P/R</th>
<th>TS</th>
<th>IID</th>
<th>EID</th>
<th>MI</th>
<th>RC</th>
<th>RR</th>
<th>IR</th>
<th>ER</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Template</td>
<td></td>
<td></td>
<td></td>
<td>Information Objects</td>
</tr>
</tbody>
</table>

P/R: Probe / Report Indication  MI: Measurement Interval
IID: Ingress node Identifier  RC: Referenced PHB-Class identifier
EID: Egress node Identifier  RR: Requested Rate\(^{27}\) (profile)
TS: Time-stamp(sequence no)  IR: Ingress Rate\(^{28}\) (profile)
    ER: Explicit Rate (profile)

**Figure 6.2.2 Control Packet Format (Data area only)**

Since the control packet format is common for both probing and reporting, an indicator (P/R-field) is required to identify its usage. Generally, IID-field indicates where the control packet is originated. Upon receiving a probe packet, the egress router will return a report packet to a node indicated by its IID. The TS-field, which distinguishes the sequence of control packets, together with IID- and P/R-field form the template of a control packet. It is designed to be a common control packet template for any control mechanism. Besides the packet template, the other part of a control is the information object, which carries the control-specific information. Its composition depends on the functionality of the control mechanism. Since an aggregated flow may be contracted to more than one PHB-classes, multiple information objects may be presented in a control packet for controlling the corresponding PHB-classes. In our design, an information object comprises: (1) a RC-field, which specifies the PHB-class under control; (2) a EID-

\(^{27}\) It is usually specified in traffic conditioning specification, TCS.

\(^{28}\) It is an instantaneous measured rate defined as \(\frac{\text{total number of bytes received in this interval}}{\text{measurement interval}}\).
field, which indicates the exit node of the flow (in some cases, an aggregate might have a set of egress nodes); (3) a MI-field that is used to synchronise the measurement interval of all participating nodes; and (4) RR-, IR- and ER-field that describe the current state of the flow.

6.2.2 Traffic conditioning

Figure 6.2.3 illustrates the design of an adaptive traffic conditioner. It is a common dual-leaky profiler with extensions of a soft random discard module and an optional adaptive shaping buffer. Its operational procedures are as follows:

When a data packet arrives at an ATC with empty shaping buffer, it is subjected to a policing operation. The profilers first update the token bucket levels of both buckets accordingly, i.e., the number of tokens inside the explicit rate (ER) bucket and request rate (RR) bucket are increased by \((t_{\text{now}} - t_{\text{last}}) \times ER\) and \((t_{\text{now}} - t_{\text{last}}) \times RR\), respectively, where \(t_{\text{last}}\) is the time when the last packet is departed. Then the size of the data packet is checked against the number of available tokens. If there are enough tokens in both ER- and RR-buckets, the packet is marked as in-profile. Otherwise, if only ER-bucket has enough tokens, the packet will be marked as out-profile. However, if there is not enough ER-token for the packet and the packet is from an adaptive flow (distinguished by the packet classifier), the random discard module will drop the packet with a probability

\[
P(\text{drop}) = 1 - \frac{\text{CurrentTokenLevel}}{\text{MeanPktSize}}
\]

Whenever a packet is dropped, a random amount of tokens is granted to the ER-bucket. It in turn spaces out the next packet dropping for the same flow and avoids the global synchronisation problem mentioned earlier (c.f. Section 5.3.2.4).

For an ATC equipped with shaping buffer, incoming packets are first en-queued and have to wait for sufficient ER-tokens. The shaping buffer implements a RED-like mechanism [79] for preventing global synchronisation problem. Once a sufficient amount of ER-tokens are available, the head-of-line packet of the shaping buffer will be processed in the same way as an ordinary incoming packet.
Upon receiving a report packet, an ATC will update its ER-token generator with the value of the ER-field carried by the report packet. When the rate of the token generator is changed, its token bucket size is also updated accordingly.

Figure 6.2.3 Adaptive traffic conditioner with soft random discard

\[ P(Drop) = 1 - \frac{CurrentTokenLevel}{MeanPacketSize} \]
6.2.3 Control algorithm

As discussed in Section 5.3.2.5, there are two major components in control algorithm: fair share computation and adaptation algorithm. Before describing the algorithm, we first define the chosen fairness criterion. In DS, user/service provider can request traffic profile such that the network operator will attempt to guarantee this profile as a minimum amount of service. The conventional max-min fairness [80] states that when all users' profiles are fulfilled, the excess bandwidth available can be allocated equally among all users. However, in reality, users prefer to get a service, which reflects the amount they are paying (i.e., the requested profile). Here we define a proportional max-min (PMM) fairness such that all active flows are served fairly in PMM sense if these two conditions are satisfied: (1) Each flow must pass through at least one bottlenecked link; and (2) the available bandwidth should be shared in proportional to the requested/committed profile when it is assigned to flows that pass through bottlenecked links.

Denote,

- \( \Phi_i \): a set of active flows traversing link \( l \), \( N_i = |\Phi_i| \)
- \( B_{c,j} \): available bandwidth for PHB class \( c \) on link \( l \)
- \( r^R_f \): requested rate of flow \( f \)
- \( L_{c,j}^{IN} \): total in-profile traffic in PHB class \( c \) on link \( l \), \( L_{c,j}^{IN} = \sum_{i \in \Phi_i} r^R_i \)
- \( w_f \): fair share factor for flow \( f \), \( w_f = \frac{r^R_f}{\sum_{i \in \Phi} r^R_i} \)
- \( A_{c,j}^{IN} = B_{c,j} - L_{c,j}^{IN} \), excess bandwidth available for in-profile traffic in PHB class \( c \) on link \( l \).
**Definition 6.1:** The proportional max-min (PMM) fair allocation \( g_f \) for flow \( f \), is defined as \( g_f = \min_{v \in \Theta_f} \{ g \} \), where \( \Theta_f \) is the set of links flow \( f \) traversed; and

\[
\begin{align*}
g_{f,j} &= \begin{cases} 
B_{c,j} \times w_f & \text{if } A_{c,f}^{IN} \leq 0 \\
R_f + A_{c,f}^{IN} \times w_f & \text{if } A_{c,f}^{IN} > 0
\end{cases}
\end{align*}
\]

Denote, \( G = \{ g : \forall i \in \Phi \} \) as the proportional max-min fair allocation;
\( R = \{ r : \forall i \in \Phi \} \) as the current rate allocation vector, where \( r_f \) is the allocated rate for flow \( f \).

**Definition 6.2:** The *proportional max-min fair allocation problem*

The proportional max-min fair allocation problem is to find the rate allocation vector equal to the proportional max-min fair allocation, i.e., \( R = G \).

### 6.2.3.1 Adaptation algorithm

Before describing the algorithm, the following notations are introduced:

- \( U \): Target utilisation factor;
- \( \Delta \): Target operating level;
- \( \varepsilon \): Target load operating level;
- \( B \): Link capacity;
- \( N \): Number of PHB classes;
- \( R_c \): Resource allocation for PHB class \( c \);
- \( \alpha_c \): Total active profiles in PHB class \( c \) in last measurement interval;
- \( r' \): Value of ingress rate carried in probe packet, IR-field;
- \( r^R \): Value of requested rate carried in probe packet, RR-field;
- \( r^E \): Value of explicit rate carried in probe packet, ER-field;

The adaptation algorithm is given as follows:
At the end of a measurement period, the following values are updated for all class $c$:

$L_c^A$ : Averaged measured total traffic load of class $c$;

$L_c^{IN}$ : Averaged measured in-profile traffic load of class $c$;

$A_c = U \times B - \sum_{i=1}^{N} R_i - \sum_{i=1}^{N} L_i^A$ ;

$A_c^A = A_c - L_c^A$ : Total available bandwidth for class $c$;

$A_c^{IN} = A_c - L_c^{IN}$ : Available bandwidth for in-profile traffic in class $c$;

$\alpha_c \leftarrow \alpha_c$; and then initialise $\alpha_c$.

When a probe packet is received:

$c \leftarrow$ RC-field in probe packet ;

$\alpha_c \leftarrow \alpha_c + r^R$ ;

$w = \frac{r^R}{\alpha_c}$ : Fair share factor;

If $A_c^A > \Delta$ \& \& $A_c^{IN} > \Delta$, i.e., unloaded case

$\bar{r}^E = r^E + A_c^A \times w$ ;

If $A_c^A \leq \Delta$ \& \& $|A_c^{IN} - A_c^A| \leq \Delta$, i.e., overloaded case

If ($r' < A_c \times w$),

$\bar{r}^E = r^E$ ;

else $\bar{r}^E = \max\{A_c \times w, r^E - \varepsilon\}$ ;

If $A_c^A \leq \Delta$ \& \& $|A_c^{IN} - A_c^A| > \Delta$, i.e., fully-loaded case

If ($r' < r^R + A_c^A \times w$),

$\bar{r}^E = r^E$ ;

else $\bar{r}^E = \max\{r^R + A_c^A \times w, r^E - \varepsilon\}$ ;

$r^E \leftarrow \min\{r^E, \bar{r}^E\}$ ;

\[\]
6.2.4 Operation

Table 6.2.1 summarises the system configurations described in previous sections. The operational procedures of the control mechanism are as follow:

1. At the FC-DS boundary, the ingress node periodically samples its incoming flows. For each sampling interval, it generates and delivers a probe packet along with the data packets per active aggregated flow. This probe packet carries the same header information as the sampled data packet, but it is remarked with the CF-DSCP at the DS-byte. The data area of the probe packet is filled with the information of the flow.

2. At any node inside a FC-DS domain, upon receiving a packet with CF-DSCP, the node computes a suggested explicit rate using the control algorithm and the information carried at the probe packet. If the suggested explicit rate is smaller than the one carried at the ER-field of the received probe packet, the ER-field of the packet will be replaced. The updated probe packet is then forwarded to the next node.

3. When an egress node receives a probe packet, a report packet is created and returned to the ingress node indicated by the IID-field of the probe packet. The report packet is identical to the received probe packet with the exception of its P/R- and TS-field, which have been updated accordingly.

4. Finally, when a report packet reaches the ingress node, the parameters of its corresponding ATC is updated.
<table>
<thead>
<tr>
<th>Functionality</th>
<th>Configurations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overall</td>
<td>Control: Intra-domain control</td>
</tr>
<tr>
<td></td>
<td>Fairness Criterion: Proportional max-min fairness</td>
</tr>
<tr>
<td>ATC</td>
<td>Traffic Profile: Token bucket based</td>
</tr>
<tr>
<td></td>
<td>Shaping: Adaptive with proportional buffer size</td>
</tr>
<tr>
<td></td>
<td>Dropping: Adaptive with soft random discard</td>
</tr>
<tr>
<td></td>
<td>Marking: Remarking ONLY within a PHB class</td>
</tr>
<tr>
<td>Probing/Reporting</td>
<td>Info type (Probe): Per-PHB-class based, averaged measured load (Exponential averaging)</td>
</tr>
<tr>
<td></td>
<td>Info type (Report): Explicit rate feedback</td>
</tr>
<tr>
<td></td>
<td>Temporal Resolution: Time based periodicity</td>
</tr>
<tr>
<td></td>
<td>Spatial Resolution: Per-aggregated flow based</td>
</tr>
<tr>
<td></td>
<td>Identification/Forwarding: CF-DSCP / CF-PHB (CF: Control Forwarding)</td>
</tr>
<tr>
<td></td>
<td>Data Collection: Single pass, forward direction</td>
</tr>
</tbody>
</table>

**Table 6.2.1 Summary of system configurations**

### 6.3 Evaluation

To evaluate the performance of the proposed flow control mechanism under a network with a heterogeneous mix of TCP and UDP flows, we have conducted a comprehensive simulation analysis. This section describes our implementation model, simulation configurations and scenarios.

Our simulation analysis targets a range of different services such as Premium Services (PS), Assured Services (AS), Olympic Services (OS), etc. Unless otherwise specified, the following discussion focuses on a generic service derived from the AF PHB. The goal of this service is to assure a minimum throughput (as specified by the user-requested profile or TCS) to a flow. It also enables the flow to get a better performance when the network load is at low level. However, in severe congestion where there is not enough bandwidth to satisfy all contracts, all flows would compete against one another for available bandwidth. Our interests are to study how effective the requested profiles can be assured and what level of fairness among flows can be achieved.
6.3.1 Implementation details

Figure 6.3.1 depicts an implementation of a FC-DS capable interior node using an extended ns-2 simulator. It consists of a PHB Classifier, packet queues and output scheduler that are commonly found in most DS nodes. As a FC-enabled DS node, it also includes:

1. a control module, which implements the control algorithm described in Section 6.2.3 and is tightly coupled with a load estimator (exponential averager);
2. a collection of per-queue measurement modules which count the number of bytes being received in every measurement interval;
3. a set of packet queues (Queue/RIO+) which implement the AF PHB classes with four drop preferences. RIO+ is an extension of RIO [81] (or a dual-RED) with 2 set of parameters for IN/OUT-of-profile packets, respectively. It has four drop preferences, which represent the packet attributes of IN/OUT-of-profile and UDP/TCP. These drop preferences can be ranked according to their dropping probabilities as IN-TCP < IN-UDP < OUT-TCP < OUT-UDP; and
4. a Queue/PQ+ scheduler that controls the outputs of packet queues. Queue/PQ+ is a simple rate-limited priority queuing that schedules packet delivery according to a pre-defined priority configuration (or resource allocation).

![Figure 6.3.1 NS2 implementation model of a FC-DS Interior node](image-url)
Furthermore, for the boundary nodes, an ingress node includes an additional adaptive traffic profiler which implements the specifications given in Section 6.2.2 whereas an egress node has a simple acknowledgement module implemented according to the requirements mentioned in Section 5.3.1.4.

6.3.2 Configurations

Throughout the simulations, there are two types of flows, TCP and non-adaptive UDP flows, each of which carries different types of traffic. While the UDP flows carry the traffic generated by CBR sources, the TCP connections are all infinite sources that simulate FTP applications. Depending on different simulation scenarios, each traffic source may generate a single flow or an aggregate of several flows. All sources, both CBR and FTP, are randomly started with starting times uniformly distributed within the first second of the simulation time for minimising the global synchronisation effect. All access links have a random round-trip delay. Unless otherwise specified, all flows are sent using AF PHB. All data packets are fixed size of 576 bytes long. Table 6.3.1 summarises the common parameters chosen for all simulation scenarios.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delay of an access link</td>
<td>Uniformly distributed between [0, accdelay]</td>
</tr>
<tr>
<td>Max. queue size of all links</td>
<td></td>
</tr>
<tr>
<td>(maxQsize)</td>
<td><strong>Bandwidth \times average RTT</strong></td>
</tr>
<tr>
<td>RIO+</td>
<td>OUT</td>
</tr>
<tr>
<td>(minth/maxth/maxp)</td>
<td>IN</td>
</tr>
<tr>
<td>Profiler token bucket size</td>
<td>CBR flows</td>
</tr>
<tr>
<td></td>
<td>TCP flows</td>
</tr>
<tr>
<td></td>
<td><strong>2 \times packet size</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Requested rate \times RTT</strong></td>
</tr>
</tbody>
</table>

Table 6.3.1 Common parameters for all simulation sets

All TCP agents implement either Reno-TCP or Sack-TCP for representing two distinct types of TCP characteristics. These flows are unidirectional and ACKs are never lost. This implies that the performance of all scenarios will be better than that of practical situations in which ACKs may be lost during congestion at the reverse path.
6.3.2.1 Network topologies

Three different network topologies are investigated: single bottleneck, multiple bottlenecks, and multiple-tier topology. Details of these topologies and the corresponding sets of simulation parameters are as follows:

**Topology 1: Single bottleneck topology**

Figure 6.3.2 shows the single bottleneck network topology. Because of its simplicity, it is useful in studying parameter settings of the proposed control mechanism. In this topology, 40 point-to-point connections share a bottleneck link. Under different cases, the shared link has a bandwidth ranging from 50Mbps to 70Mbps. For all cases, a 10 Mbps access link connects each host to the router such that a flow is setup from host $S_x$ to host $D_x$ and is identified by a flow ID $x$. Three sets of simulations will be conducted under this topology. For ease of illustration, the 40 flows are arranged into 4 groups. At each access point of the router, a profiler that is configured with a requested rate of a flow is attached. Since the CBR sources send at a data rate of 3Mbps, which is higher than the requested profiles, some portions of their data are expected to receive a lower priority service. Table 6.3.2 has lists all simulation parameters together with the expected target rate of all flows under different scenarios.

![Topology 1 - Single bottleneck topology](image)
Table 6.3.2 Simulation parameters for topology 1

**Topology 2: Multiple bottleneck topology**

To further study the fairness under situations where flows sharing a domain have different round-trip-time and traverse different number of routers, a multiple bottleneck topology, as illustrated in Figure 6.3.3, is constructed. In this topology, there are 8 connections from Sx to Dx, each of which carries an aggregate of 10 independent flows. With the exception of S8, which generates an aggregate of 20Mbps, all sources transmit an aggregate flow containing 10 microflows of 1Mbps. For each aggregated flow, a traffic profiler of 10Mbps is requested at the access point of the DS domain. Under this topology, flows may have different characteristics. For example, aggregated flow #0 and #1 traverse FOUR routers while all other aggregated flows pass only TWO routers. Also, flows passing the last link between R2 and R2 are more congested than the other flows. Because the sum of contracted rates at this congested link exceeds the link capacity, it is expected that the contacted rates may not be assured in some flows. Table 6.3.3 lists the simulation parameters together with the expected target fair share of all flows under ideal situations.
Figure 6.3.3 Topology 2 - Multiple bottleneck topology

<table>
<thead>
<tr>
<th>Set</th>
<th><strong>Agg. Flow</strong></th>
<th>Src Type</th>
<th>Src Rate (Mb/s)</th>
<th>Requested Profile (Mb/s)</th>
<th>Target Fair Share Rate (Mb/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>0</td>
<td>CBR</td>
<td>1.0</td>
<td>10.0</td>
<td>7.5</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>TCP</td>
<td>/</td>
<td>10.0</td>
<td>7.5</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>CBR</td>
<td>1.0</td>
<td>10.0</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>TCP</td>
<td>/</td>
<td>10.0</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>CBR</td>
<td>1.0</td>
<td>10.0</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>TCP</td>
<td>/</td>
<td>10.0</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>CBR</td>
<td>1.0</td>
<td>10.0</td>
<td>7.5</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>TCP</td>
<td>/</td>
<td>10.0</td>
<td>7.5</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>CBR</td>
<td>2.0</td>
<td>20.0</td>
<td>15</td>
</tr>
<tr>
<td>5</td>
<td>0 - 7</td>
<td>TCP</td>
<td>/</td>
<td>@ 10.0</td>
<td>7.5 (S0,1,6,7); 15 (S2-5)</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>TCP</td>
<td>/</td>
<td>20.0</td>
<td>15</td>
</tr>
<tr>
<td>6</td>
<td>0 - 7</td>
<td>CBR</td>
<td>@ 2.0</td>
<td>@ 10.0</td>
<td>7.5 (S0,1,6-7); 15 (S2-5)</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>CBR</td>
<td>2.0</td>
<td>20.0</td>
<td>15</td>
</tr>
</tbody>
</table>

**Each aggregated flow contains 10 microflows.**

Table 6.3.3 Simulation parameters for topology 2

132
**Topology 3: Multiple-tier topology**

The last topology for evaluation is a multiple-tier network as shown in Figure 6.3.4. This simulates a practical network scenario, which is composed of multiple interconnected DS domains. Again, there are 8 connections from Sx to Dx, each of which passes a different set of domains. Each traffic source in a connection generates an aggregate of 10 independent flows, each of which has a data rate of 1Mbps. Under this topology, all aggregated flows have different lengths in terms of number of hops and RTT. At each access point of a domain (including user-domain or domain-domain access), a traffic profile with certain parameter stated in Table 6.3.4 is contracted. With this topology, we will study the interaction between the locally controlled domains. More specifically, since flows are naturally aggregated at the long-haul access links between domains, traffic bursts can be accumulated throughout the network. In addition to the fairness among flows, the effect of marking and burstiness at the domain access points are analysed.

![Figure 6.3.4 Topology 3 - Multiple-tier topology](image)

Access DS Domains
\[ Acorebw = 45Mbps \]
\[ Accdelay = 20ms \]

All Access Links
\[ accbw = 15Mbps \]
\[ accdelay = 10ms \]

Long-haul access links
\[ intbw = 45Mbps, \]
\[ intdelay = 40ms \]

Backbone DS Domains
\[ Bcorebw = 45Mbps \]
\[ Baccdelay = 20ms \]
Table 6.3.4 Simulation parameters for topology 3

### 6.3.2.2 Metrics

The following lists the performance metrics being used for the evaluation:

- **Average achieved rate**: an average of the observable data rate at the receiving host, or

  \[
  \text{average achieved rate} = \frac{\text{Total number of bytes receiver in } \text{win}}{\text{win}};
  \]

  where \(\text{win}\) is the time window and is chosen to be 0.5 second.

  Results in terms of average achieved rate may be presented as the percentage error from the target fair share rate, or

  \[
  \%\text{Error from Target Rate} = \frac{\left(\text{average achieved rate} - \text{target rate}\right)}{\text{target rate}}.
  \]

- **Packet transfer delay**: the elapsed time for a packet to be passed from the source, through the network, to the receiving host.

- **Remarking rate**: an average re-marking probability, defined as

  \[
  \text{remarking rate} = \frac{\text{Total number packets being remarked}}{\text{Total number of packets received at the marker}}.
  \]
6.3.3 Results and analysis

This section discusses the simulation results of the proposed feedback controlled DS with the comparison of the conventional DS network. Results presented in the following sections are organised according to various effects influencing the traffic flows. The following notations will be used throughout the presentations:

- **FC-x-y**: microflow *y* within aggregate *x* under a Feedback Controlled DS network
- **UC-x-y**: microflow *y* within aggregate *x* under an Un-Controlled DS network
- **Target-**: Target fair share

### 6.3.3.1 Effect of non-adaptive flows

In the presence of non-adaptive flows, all TCP connections are degraded. This is true even though they are protected inside their requested profile envelope as long as the network has been adequately provisioned. However, excess bandwidth or any scarce resource available during congestion is taken by non-adaptive flows because the TCP sources back off when their out-of-profile (or OUT) packets are dropped. As indicated in Figure 6.3.5(a), non-adaptive UDP (or CBR in the simulation) flows achieve a rate much higher than the expected fair share rate in a conventional DS domain (or uncontrolled DS domain). This advantage towards non-adaptive flows is the result of a significant performance degradation of the TCP flows (c.f. Figure 6.3.5(b)).

This unfair situation can be remedied by employing the proposed control mechanism. In a FC-DS domain, non-adaptive flows are regulated according to the fairness criterion so that they are prevented from monopolising the available resource. Figure 6.3.5 confirms that both CBR and TCP flows under feedback control can achieve their rates very close to the target fair share rates for any loading condition.
Set #1 - Group 1 & Group 2 (CBR flows)

Set #1 - Group 3 & 4 (TCP flows)

Figure 6.3.5 Average achieved rate for Set #1
6.3.3.2 Effect of requested flow profiles

From Figure 6.3.6(a), we notice that connections with small requested profiles (e.g., flows in group 1 of Set # 2) reach or exceed their profiles noticeably in conventional DS. This is due to the variation of TCP congestion window. After the window is closed because of packet losses, the connections with small requested profiles return to their legitimate window size quicker than those with larger profiles. Therefore, they can compete for the excess bandwidth sooner. Since the supplementary traffic profile in FC-DS opens gradually in a fair manner, it provides a fair ground for flows with different requested profiles to compete for the available resources. Hence the percentage errors deviated from the target fair-share rate for all flows are significantly improved. Results of FC-flows shown in Figure 6.3.6(b) confirm these findings.
Figure 6.3.6 Average achieved rate for Set #2
6.3.3.3 Effect of interference from high priority class

To study the influence of inter-class interference, we have repeated the simulation set#1 with an additional connection. This connection injects an interfering traffic of 20Mbps CBR flow in the interval of [20s, 40s] and [80s, 100s] using the EF-codepoint. Figure 6.3.7 shows a trace of the first 60 seconds. Since traffic on EF-PHB has a higher forwarding priority than AF classes, it creates a sudden starvation of resource. While the uncontrolled flows completely fall away from the target rates, the feedback-controlled flows follow closely with the abrupt change of available bandwidth, as illustrated in Figure 6.3.7. This confirms that the proposed control mechanism is free from stability problems during the transition period.

![Transient response](image)

*Figure 6.3.7 Time response of average achieved rate for Set # 3*
6.3.3.4 Effect on long and short flows

A long flow refers to a flow that traverses a number of nodes. In topology 2, microflows within aggregated flow-0 and flow-1 are the longest flows as they traverse all nodes inside the domain. In conventional DS, long flows usually have poorer performance than other flows. This is because every time a packet enters a node, it has to compete with others for available resources. Since packets or flows are indistinguishable inside the core of a DS domain, the larger the number of nodes they travel, the higher the probability that they will experience loss or delay. We observe this effect from Figure 6.3.11 and Figure 6.3.14 that flow-0 and flow-1 receive a poor-than-expected service no matter if they are UDP(CBR) or TCP flows. However, this has an exception of flow-0 in Set # 4 (Figure 6.3.8) because flow-0 is an non-adaptive flow which has an advantage over the adaptive flows such as flow-1.

Although short flows have advantage over long flows in acquiring available resources, this is not the case under FC-DS because their accesses have been fairly regulated according to the fairness criterion at the bottleneck link. In other words, long flows are being protected from being hit by the short or aggressive flows in the FC-DS domain. Results of FC-flows shown in Figure 6.3.8, Figure 6.3.11 and Figure 6.3.14 confirm that long flows can achieve their target rates effectively in spite of the existence of short flows.

Another view of analysing the effect on long flows is from the packet delay perspective. In the uncontrolled version of topology 2 network, the severe congestion at the last link makes all flows passing through that link experience a longer delay or higher loss probability. Figure 6.3.15 shows that flow-0 and flow-1 (or long flows) can experience a delay as large as 0.5 second. By introducing TCP flow control to all flows, Figure 6.3.12 shows that their delay can be brought down to about 0.35 second. However, under FC-DS, these delays can be further lowered to below 0.21 second. This observation implies that a simple source based congestion control mechanism, such as TCP, is necessary but insufficient under this circumstance. It is because the TCP-type flow control does not
consider the fairness among flows. In this situation where there are flows with uneven round-trip-time (RTT), flows with shorter RTTs will take more buffer spaces. This makes the long flows experience a longer delay even though there is no congestion occurs.

Another interesting observation is that the feedback control mechanism also helps improve the performance of short flows under certain circumstances. In topology 2, congestion occurs at the last hop between R2 and R3, i.e., severe packet dropping occurs at R3 while excess resources are available at other nodes. In an uncontrolled environment, if S0 is non-adaptive and not aware of any congestion at the downstream nodes, it will continuously inject packets into the domain. These packets maintain a certain level of buffer occupancy at router R0, R1 and R2 even though they are eventually dropped at R3. In this situation, network resources are wasted at the non-congested nodes. Other flows may either be prevented from accessing the originally available resources (i.e., higher packet loss rate), or they may experience longer delay because of the buffer occupancy. By introducing a FC at the DS edge, packets from S0 can be throttled down earlier at R0 and thus persistent congestion can be prevented in the DS domain. Figure 6.3.10 confirms this observation that delay experienced by the short flows can be significantly improved under FC-DS.
Achieved Rate - aggregated Flows
(TCP Vs CBR Flows)

Aggregated Flow #

Figure 6.3.8 Average achieved rate for Set # 4

Delay distribution of the Long flows
(TCP Vs CBR flows)

Figure 6.3.9 Delay distribution of microflow-2 within aggregate-0 & -1 for Set # 4
Figure 6.3.10 Delay distribution of microflow-2 within aggregate-2 to -8 for Set # 4
Achieved Rate - aggregated Flows
(All TCP Flows)

Figure 6.3.11 Average achieved rate for Set # 5

Delay distribution of the Long flows
(All TCP flows)

Figure 6.3.12 Delay distribution of microflow-2 within aggregate-0 & -1 for Set # 5
Delay distribution of Short paths
(Feedback Controlled: All TCP flows)

(a)

Delay distribution of Short paths
(Un-Controlled: All TCP flows)

(b)

Figure 6.3.13 Delay distribution of microflow-2 within aggregate-2 to -8 for Set #5
Figure 6.3.14 Average achieved rate for Set #6

Figure 6.3.15 Delay distribution of microflow-2 within aggregate-0 & -1 for Set #6
6.3.3.5  **Effect of round trip time**

Figure 6.3.17 shows the performance of TCP flows under a typical multiple-tier scenario as in topology 3. As discussed in previous sections, the influence of RTT on the achieved rate is noticeable. For the case without FC, Figure 6.3.17 also verifies that some flows, such as flow-1, -3, -5 and -7, do not achieve their target fair-share rates, whereas others flows, such as flow-0, -2, -4, -6, severely exceed their targets. It is mainly because flow-0 and flow-4, which have the shortest RTTs, grow their TCP congestion window more quickly and come out of their requested profile envelopes more frequently to exploit excess bandwidth using their OUT packets. Since these OUT packets cannot prevent the in-profile (or IN) packets of other flows from entering the router queue, flows with larger RTTs are at least assured of their requested profile rates if resources are sufficiently provisioned. However they can hardly receive a fair share of the excess bandwidth. Figure 6.3.17 confirms this observation. All flows can generally achieve their requested profiles (i.e., 5Mbps) while only flows with short RTTs can enjoy the excess bandwidth under an uncontrolled DS domain. Again, with the feedback control mechanism, this unfair situation can be effectively minimised.
Figure 6.3.16 Average achieved rate for Set # 7
Figure 6.3.17 Average achieved rate for Set # 8
6.3.3.6 Effect of re-marking

In topology 3, traffics are aggregated at the merge point. Therefore, traffic-bursts can be accumulated throughout the network. When a traffic-burst hits the edge of a DS domain, it is remarked according to the contracted inter-domain aggregated profiles. Hence the higher the remarking rate, the more bursty the incoming traffic is. Figure 6.3.18 and Figure 6.3.19 plot the remarking rate of different aggregated flows at the edges of domain 2 and 3. In these figures, a positive remarking rate (or promotion) refers to the situation where packets are being remarked from out-of-profile (OUT) into in-profile (IN) and vice versa for negative remarking rate or demotion.

Results from both sets of simulation demonstrate that with the absence of a feedback control mechanism, traffic tends to be more bursty at the edge of a domain no matter if all traffics are of the same type or not. It is indicated by the fact that the remarking rates of all flows under an uncontrolled DS are higher than the ones in the FC-DS case.

Furthermore, if we consider the merging point at the last domain D-3, we notice that remarking occurs at an uneven fashion upon different aggregated components (i.e., D3-5 >> D3-1) under UC-DS. This implies traffic is highly unbalanced at the merge point. However, this situation does not occur in FC-DS. From Figure 6.3.19, we notice that remarking rate for D3-1, -3, -5, and -7 are all comparable when feedback control is exercised. In essence, while a feedback control mechanism can help reduce the traffic burstiness, it can also balance the composition of the aggregated traffic.
Remarking TCP Vs CBR Flows

![Graph showing the remarking rate of aggregated flows for Set #7 and Set #8.]

(Notation: \(D_{x-y} = \text{Domain } x \text{; aggregate flow } y\))

**Figure 6.3.18** Remarking rate of aggregated flows for Set #7

**Figure 6.3.19** Remarking rate of aggregated flows for Set #8

151
6.3.4 Evaluation of control algorithm

In order to explain the dynamics of the control algorithm, we refer to a simple scenario where there are only two flows sharing a bottleneck link. Figure 6.3.20 illustrates this scenario by a two-dimensional space in which the axes represent the ingress rates of two flows respectively. Therefore, a point in this space represents an operating point of the control algorithm. Any point along the line of \( x_1 + x_2 = \Phi \) represents the total rates of two flows that is currently at the capacity of the link (\( \Phi \)). Thus, this line is known as the "efficiency line". When the node is operating in a region close to this line, we consider the node to be operating efficiently. Moreover, when \( m_2x_1 = m_1x_2 \) (where \( m_1 \) and \( m_2 \) are the requested profile rates of the flows), the resource allocation to the flows is at the fair share. Therefore, this line is known as the "fairness line". The goal of the control algorithm is therefore to bring the resource allocations from any point in this space to the point \( \bar{G} = (g_1, g_2) \), which is the intersection of the efficiency line and the fairness line.

\[ x_1 + x_2 = \Phi \]

**Figure 6.3.20 Dynamics of the control algorithm**
In the general scenario where there are more than 2 flows, an operating point becomes a point in the multidimensional space. While the fairness line is still a line in this space, the efficient line will become a hyper-plane.

6.3.4.1 Dynamics of the control algorithm

The two fundamental design objectives of any control algorithm are (1) achieving efficiency and (2) achieving fairness. A good algorithm should bring the system operating point closer to the plane of efficiency as soon as possible. Failing in doing so will result in poor resource utilisation. Once the target efficiency is achieved, the algorithm should adjust the allocations to achieve the desirable fair resource sharing.

Based on the algorithm given in Section 0, the following paragraphs describe the dynamics of the proposed algorithm:

- At any given operating point other than full efficiency, the algorithm will bring the operating point to the efficiency plane in one step. It is achieved by increasing the explicit rate carried by the control packet such that the ingress nodes will adjust their traffic conditioners accordingly. To meet the fairness requirement, the increments are proportional to the requested profiles of the flows. That is, graphically, the operating point is shifted towards the plane of efficiency in a direction parallel to the line of fairness.

- Once the target efficiency is attained but the fairness criterion has not been satisfied, the algorithm will shift the operating point towards the fairness line. This operation basically involves maintaining the rates of the legitimate flows and tightening the rates of the over-committed flows. As shown in Figure 6.3.20, the iteration of shifting $\hat{X}^t$ to $\hat{X}^{t+2}$ represents this operation.

- Again, once the link is operating below the target efficiency, the explicit rates are increased again in the similar fashion as mentioned above.

- After several iterations, the algorithm should have moved the operating point to the point of convergence where both fairness and efficiency objectives are met.

153
6.3.4.2 Proof of convergence

To facilitate the proof of convergence of the proposed control algorithm, we have made the following assumptions:

- All data sources are infinite (or greedy) sources, which always have data to send and can utilise all rate allocations.
- Feedback in terms of reports packets are given to the ingress nodes instantaneously and synchronously. This implies the round trip delay and jitter of the control packets are negligible.
- There are no input load changes, e.g., new flow arrived, during the course of convergence.

Now, denote \( \| \tilde{X} \| = x_1 + x_2 + \ldots + x_n \)
\( \tilde{X} = (x_1, x_2, \ldots, x_n) \) is the rate vector of the participating flows; 
\( (m_1, m_2, \ldots, m_n) \) are the requested rates associated with the corresponding flows; and 
\( \mathbb{R}^{n^+} \) is the \( n \)-dimensional non-negative Euclidean space.

**Definition 6.3: (Plane of Efficiency)**

For any \( \tilde{X} \in \mathbb{R}^{n^+} \), \( \| \tilde{X} \| = \Phi \) is the hyperplane of efficiency, or \( POE \), where \( \Phi \) is the target link bandwidth.

**Definition 6.4: (Line of Fairness)**

For any \( \tilde{X} \in \mathbb{R}^{n^+} \), \( m_i \times_i = m_j \times_j \) \( \forall i, j \) where \( i \neq j \) represents a line in the \( n \)-dimensional space and is called the line of fairness or \( LOF \).

**Definition 6.5: (Point of Convergence)**

For any \( \tilde{G} \in \mathbb{R}^{n^+} \), \( \tilde{G} = (g_1, g_2, \ldots, g_n) \) is the point of convergence, or \( POC \) if \( \tilde{G} = POE \cap LOF \).
Definition 6.6: (Region of Adaptation)
For any $\tilde{X} \in \mathbb{R}^{n^+}$, the region of adaptation, or ROA, is a region bounded by
\[
\begin{align*}
\{ & \Phi - k\varepsilon \leq \|\tilde{X}\| \leq \Phi \\
& x_i \geq g, \quad \forall i
\end{align*}
\]
where $k < n$ and $\tilde{G}$ is the point of convergence.

The following three lemmas can be derived from the control algorithm:

Lemma 6.1: \hspace{1cm} \forall \tilde{X} \in \mathbb{R}^{n^+}, \text{if } \tilde{X}' < \text{POE}, \tilde{X}'^{+1} \in \text{POE}.

Lemma 6.2: \hspace{1cm} \forall \tilde{X} \in \mathbb{R}^{n^+}, \text{if } \tilde{X}' \in \text{ROA}, \tilde{X}'^{+1} \in \text{ROA}.

Lemma 6.3a: \hspace{1cm} \forall \tilde{X} \in \mathbb{R}^{n^+}, \text{if } \tilde{X}' \in \text{POE}, \text{for any } \delta > 0, \exists N, \text{ such that,}
\[
\|\tilde{X}^{2k+1} - \tilde{G}\| < \delta \quad \text{for all } k > N. \text{ Or simply, } \lim_{n \to \infty} \|\tilde{X}^{2k+1} - \tilde{G}\| = 0.
\]

For $n = 2$, the following Lemma states that the control algorithm converges with an exponential rate.

Lemma 6.3b: (Rate of Convergence)
\[
\forall \tilde{X} \in \mathbb{R}^{2^+}, \text{ if } \tilde{X}' \in \text{POE}, \text{ for any } \delta > 0, \text{ } \|\tilde{X}^{2k+1} - \tilde{G}\| \text{ converges to } \delta \text{ with a rate of } S_2 (g_1 - x_1') \log S_2, \text{ where } S_2 = \frac{m_2}{m_1 + m_2}.
\]

Theorem 6.1: (Convergence theorem)
Starting at any arbitrary state of the network, if the above assumptions hold, the control algorithm converges to the PMM fair allocation.

The proofs of these Lemmas and theorem are given in Appendix B. Lemma 6.1 states that when the node is operating at any point below POE, the algorithm will attempt to bring the system into full efficiency in the next control interval. During the course of adaptation, Lemma 6.2 ensures that the node will maintain its efficiency within the region of adaptation. Lemma 6.3a mainly states that the alternate control iterations will converge to the point of convergence in which both fairness and efficiency requirements are met. For the two-dimensional case, Lemma 6.3b states that the control algorithm moves faster towards the point of convergence at the early iterations. Then, the algorithm slows down
when the operating point gets closer to the point of convergence. Finally, in the network of nodes, Theorem 6.1 states that the control algorithm will bring the network to a steady state given that the above mentioned assumptions hold.

6.3.4.3 Discussions of the control algorithm

The proposed control algorithm has several merits:

1. The algorithm can achieve the fairness in PMM sense with minimal complexity. It has an $O(1)$ per aggregated flow computational complexity and each computation requires only 2 multiplication operations. Therefore, the algorithm is very suitable for and is an attractive option for hardware implementation;

2. The algorithm requires only the information of per-class measured load. This information is relatively easy to obtain and requires minimal processing overhead. In some cases, it is readily available for other network management purposes;

3. The algorithm considers the activity of the flows. If some flows are currently idle, their requested profiles will not be considered in determining the resource allocations of the active flows, thereby resulting in fairer resource sharing;

4. The algorithm can generally maintain high efficiency. During the course of adaptation, the parameter of target load operating level, $\varepsilon$, can be used to limit the efficiency variation. Generally, the larger the value of $\varepsilon$, the shorter the system convergence time, but the higher the efficiency variation may result. Moreover, the parameter of target utilisation factor, $U$, also affects the overall throughput. By carefully adjusting this parameter and observing the effect of average buffer level, we may compensate the loss in throughput during the rate adaptation; and

5. The algorithm is robust against error or loss in control packets. Given that the input traffics are stable, the algorithm is "self-correcting". It is because at any erroneous operating point, the algorithm will correct the operating point at the next iteration and bring the system towards the point of convergence.

While the proposed algorithm has a number of strengths, its major limitations are (1) its time of convergence when the system configuration is complicated and (2) its sensitivity
to the measured information. Our discussion so far is generally applicable to the situation where the bottleneck link is shared by unconstrained flows (which can consume all given resource allocations). It also assumes that feedback information is given to the ingress nodes instantaneously and synchronously. In the practical situation where these assumptions do not hold, the system may take longer time to converge to the fair and efficient operating point. Moreover, if the variations of the network components are of a time scale smaller than the required convergence time, the system might end up being unstable. However, this argument is generally true for any dynamic control mechanism.

Nevertheless, our algorithm design has considered the trade-off between the speed of convergence and the implementation complexity. More importantly, the design has taken the TCP dynamics into account. More specifically, if a shared queue has not been properly drained out, no matter how much increment in the explicit allocation the algorithm has made to the flows, the ingress rates of the flows cannot be increased immediately. It is because when the TCP flow control mechanism opens up its congestion window, it takes the packet RTT into consideration. This situation generally happens when the system is at the efficient point where the queue has accumulated a certain level of packet, but still attempts to move its state towards the fairness line. Concerning this effect, our algorithm is designed to first tighten the over-committed flows while holding the current allocations of all other flows instead of adjusting the allocations of all flows at the same time to attain the fairness condition.

Furthermore, since the control algorithm is based on measurement, it is slightly sensitive to the measured information such as the sum of active flow requested profiles and measured load. For example, if the sum of active profiles is underestimated, the algorithm will attempt to converge to a higher fair-share value and oscillate around the region of adaptation. However, the node is still maintained at a target utilisation level. Since the sum of active profiles can hardly be overestimated (unless there is error in probe packet), the algorithm always achieves efficiency. Besides the active profiles, another measured quantity is the averaged load of the classes. In situation where the measured load fluctuates (it may due to sudden load changes in higher priority classes or incorrect
measurement), the exponential averaging operation can help alleviate the variation. However, the loading condition of the classes may still be over- or under-estimated for some extreme conditions. In these cases, the system may jump to an arbitrary position before starting to converge and thus takes more time to reach the fairness and efficient operating point.

6.4 Discussions

This section discusses several issues related to the design of a FC-DS network.

6.4.1 Choice of RIO+ parameters

Choosing a right set of parameters for the RIO or RED scheme is always a difficult task. Although several suggestions have been proposed in the literature [13, 70, 71, 72, 79, 81], some of them are conflicting with one another. Our choices of parameters are not by these so called "rule-of-thumb". Instead, they are chosen based on the results obtained from extensive simulations of topology 1. After examining a wide range of parameter values, we have obtained the following two general guidelines:

- If the average queue length oscillates around minth, the early congestion detection mechanism is too aggressive. Thus, the value of maxp should be reduced to slow down the early congestion detection. Otherwise, the link will be operated at a low throughput and high dropping condition. On the contrary, if the average queue length oscillates around maxth, the early detection is too conservative. With this high buffer occupancy level, packet will experience a long queuing delay. To compensate this effect, the maxp value should be increased.

- It is generally more desirable to make (maxth - minth) sufficiently large to avoid TCP global synchronisation problem. However, minth should be large enough to maximise the link utilisation. Also, maxth should be set with reference to the maximum allowable delay of the committed flows.

The parameters given in Section 6.3.2 are chosen based on these guidelines. However, we have noticed that the same qualitative results are obtained when we compare the performance of FC-DS and the conventional DS regardless of the chosen parameter
values. Furthermore, we have also noticed that tuning the RED parameters cannot remedy the problems cited in Section 5.2.2. Generally, performance of a DS network without the feedback mechanism is more sensitive towards the parameter settings. An improper set of RED parameters may result in very poor performance. However, it has been observed that a FC-DS domain is more robust against the choice of those parameters.

6.4.2 Tradeoffs and complexity

Obviously, one of the overheads of deploying this feedback control mechanism over a conventional DS network is the control traffic. It is generally proportional to the chosen probing period and the number of active flows. Depending on the complexity of the network elements, a control packet is usually much smaller than the size of a data packet. Therefore, this control traffic should not consume too much network resource.

Furthermore, as mentioned in the last section, the overall throughput of the network may be affected during the course of rate adaptation. However, by carefully setting the utilisation factor to maintain the packet queue at a certain level, the loss in throughput can be replenished. In addition, another factor affecting the network throughput is the TCP global synchronisation effect. It usually happens when traffic conditioners update their explicit rate profiles. To alleviate any damage on the overall throughput due to this effect, we have demonstrated that a soft random discard mechanism at the ATC is an effective means to avoid this effect.

Apart from the influence on the overall throughput, the following lists the additional processing required when constructing the FC-DS domain:

- At the interior nodes, the additional processing requirements are:
  - (1) measurement operation: since the proposed mechanism uses a simple metric of average load, its measurement operation is very simple and should be easily implemented in hardware. It mainly involves counting the number of bytes received during a measurement interval. Also, the overall measurement per node depends on the number of available PHB classes and the length of a measurement interval;
(2) computation operation: as mentioned in the last section, the proposed algorithm has an $O(1)$ complexity and each computation requires only two multiplications;

(3) control packet processing: the mechanism requires an additional queue and control logics to handle the control packets. Since the proposal suggests using a DSCP for identifying the control packets, packet classification should be easily performed.

- At the ingress nodes, the conventional traffic conditioning operation is augmented to an ATC. This includes an additional profiler and modules for performing soft random discard mechanism. Furthermore, an ingress node is also responsible for generating probe packets, which involves sampling a packet stream and composing a probe using the selected packet.

- Finally, the egress nodes require no special operation but bouncing the control packets back to their ingress nodes accordingly. This is as simple as updating the source and destination addresses.
6.5 Summary

This chapter has presented an instance of the FC-DS specifically for the purpose of flow control. It addresses the problem of fairness and incapability of QoS provisioning under the conventional DS framework. The proposed algorithm and the overall system have been evaluated using analytical and simulation analysis. We have proved analytically that the control algorithm can bring the overall network to a steady state where both fairness and efficiency criteria are met. Furthermore, we have also conducted extensive simulations to analyse the performance and dynamics of the overall system. Three network topologies, from a simple single bottleneck network to a multiple tiers network, have been simulated. Our results have demonstrated that with the proposed flow control mechanism, the performance of a DS network, in terms of resource sharing, congestion avoidance or prevention and QoS provisioning, can be significantly improved.

Finally, we have also discussed the tradeoffs and overall complexities of this flow control mechanism. While the proposed control algorithm can offer a significant performance improvement over the conventional DS network, it is also simple enough and suitable for hardware implementation. Therefore, it is worth considering as an extension of the existing DS framework.

In summary, this chapter has presented only one possible algorithm for performing flow control. Certainly, there are many other algorithms and even other interesting applications (not limited to flow control) that can be derived from the general feedback control paradigm proposed in Chapter 5. These developments will be interesting topics for further investigation.
Chapter 7

Conclusion and Future Work

"Human life is limited, but knowledge is limitless. To drive the limited in pursuit of the limitless is fatal; and to presume that one really knows is fatal indeed!"

Chuang Tzu, The Preservation of Life 3:1 (399-286 BC)

— Translated by Lin Yutang

This chapter concludes the thesis with a summary of research contributions and areas for future work.

7.1 Summary of Contributions

This thesis has discussed a number of issues regarding the development of QoS framework over the Internet. It has described how QoS requirements are specified and has clarified the QoS models proposed by the IETF community. In addition, it has presented a comprehensive review of the primitive mechanisms that are currently available for network control purposes. Having discussed these QoS issues, this thesis has focused on solving several important issues that arise from supporting these QoS models over the Internet.
For supporting the Int-Serv IP, this thesis has examined the feasibility of using the ATM technology. Several mechanisms and recommendations to the basic switch design have been proposed. More specifically, Chapter 3 and 4 have:

1. Developed an ISAC architecture [82] based on the proposed fusion model that efficiently integrates Int-Serv and RSVP into ATM. As part of this architecture, a number of novel mechanisms have been proposed. Two of the major developments are (1) a simple VCI allocation protocol (VCAP) which interfaces RSVP with ATM VC switching mechanism to eliminate unnecessary protocol duplication; and (2) a controller which converts an off-the-shelf ATM switch into an Int-Serv and RSVP enabled switch/router. A prototype ISAC network has been built and tested in the NAL ATM testbed. The results have demonstrated an operational environment that efficiently facilitates the delivery of video over RTP/UDP/Int-Serv-IP/ATM. In addition, the results have also shown that SNMP is not a suitable protocol for switch control purpose, even though it is a common protocol available in most ATM switches today;

2. Investigated the problem of supporting heterogeneous multicast flows given that conventional ATM multicast VCs are homogeneous reservation type. In view of this fundamental conflict, this thesis has introduced a concept of piping. To realise this concept, alternative-path forwarding and token forwarding mechanisms are developed. This piping approach not only matches the spirit of Int-Serv and RSVP models, but it also eliminates the limitations found in other heterogeneity models proposed in [31]; and,

3. Examined another unique characteristic of IP multicast that is anypoint-to-anypoint connectivity. It has shown that supporting QoS under this multicast paradigm requires an additional mechanism, i.e. flow merging at the ATM switches. This thesis has proposed a novel VC-merge capable scheduling [83] which can perform both flow merging effectively and deliver QoS guarantee at the same time. A VC-merge capable scheduler has been developed, implemented and evaluated. The results have shown
that for most cases where user reservations are reasonably utilised, this scheduler not only can perform flow merging efficiently, but it can also offer a comparable scheduling performance as its non-merging counterpart. Apart from handling anypoint-to-anypoint multicast connections, this scheduling mechanism also provides a scalable solution for the usage of L-2 label space. It is particularly useful for the concept of cut-through L-3 switching approaches.

Furthermore, the second part of this thesis has focused on examining the Diff-Serv architecture. It has shown that in order to deliver a QoS guarantee without per-flow resource allocation (i.e. Int-Serv), an over-provisioned (or over-engineered) network is required. Generally, this can be achieved by tightening the admission control policy of the network such that excess resources are available for the admitted flows. However, such network suffers from problems of unfair resource sharing and efficiency. To address these problems, a general feedback control extension [84] to the existing Diff-Serv framework has been designed and presented in this thesis. With this extension, network providers can manage and control their DS network more precisely and efficiently.

To justify this recommendation, a flow control mechanism, tailored for the DS network and derived from the proposed feedback control extension, has been developed and analysed. The results have demonstrated that a flow controlled DS network out-performs a conventional DS network significantly in terms of fairness, resource sharing, congestion avoidance, efficiency and QoS provisioning. In addition, it has shown that this mechanism is simple enough and suitable for hardware implementation.

7.2 Future Research Directions

Though this thesis has addressed a number of important issues and provided a number of solutions, there are several issues that warrant further investigation. As part of the on-going work, several aspects of this thesis can be further extended or are currently being extended. The following lists some of the on-going work and future research directions.
1. As part of the Virtual Networks Resource Management Project, part of the ISAC work has been extended. More specifically, the switch control module in ISAC has led to the development of a **MIBlet concept** [85]. MIBlet is a logical partition of the Management Information Base (MIB) of a network element. It serves as customer control interfaces between the customer management system and the allocated virtual network resource. In addition, a Java porting of ISAC-R controller is being developed. With the MIBlet concept and a portable ISAC-R controller, multiple virtual ISAC networks can be customised on-demand while they are physically located on the same ATM backbone network.

2. Apart from the application in ATM, the VC-merge capable scheduling concept can also be adapted into any packet/frame based network, such as MPLS-over-PPP, in which flows of packet/frames are required to merge.

3. Chapter 6 has presented a powerful flow control algorithm derived from the proposed general feedback control extension; however, other flow control algorithms and control mechanisms (not limited to flow control) can also be developed for various purposes. For example, a topological control algorithm, which scales better than flow based approach, is currently being examined. Furthermore, other useful control mechanisms, such as receiver control, QoS monitoring and inter-domain control mentioned in Section 5.4, are also worth further investigation.

### 7.3 Concluding Remark

Undoubtedly, delivering QoS in the Internet involves many aspects of network design, some of which are beyond the scope of this thesis. Though this thesis has addressed a number of important ones and provided several solutions, there are still open issues that need to be addressed. Nonetheless, the results presented in this thesis can serve as an important step towards better understanding of issues related to the support of QoS over the Internet.
Appendices

Appendix A

Proof of Theorem 4.1

Theorem 4.1: For any sub-queue \( i \) under a VC-merge capable scheduler (VMSCH), the following holds,

\[
\alpha_{i,\text{VMSCH}} = \alpha_i + \frac{\phi_i}{\phi_{\text{SEQ}(i)}} \alpha_{\text{SEQ}(i)}
\]

where \( \alpha_{i,\text{VMSCH}} \) is the B-WFI for sub-queue \( i \) and \( \alpha_{\text{SEQ}(i)} \) is the B-WFI for the virtual queue containing sub-queue \( i \). \( \Box \)

Let \( [t_1, t_2] \) be any time interval that sub-queue \( i \) is continuously backlogged. It follows that the virtual queue is continuously backlogged assuming no snooping collision occurs, if SCH is worst-case fair with virtual queue at SEQ. We have,

\[
W_{\text{SEQ}(i)}(t_1, t_2) \geq \frac{\phi_{\text{SEQ}(i)}}{\phi_{\text{SCH}}} W_{\text{SCH}}(t_1, t_2) - \alpha_{\text{SEQ}(i)}
\]

where \( W_{\text{SEQ}(i)}(t_1, t_2) \) is the amount of service received by virtual queue in \( [t_1, t_2] \).

Multiplying \( \frac{\phi_i}{\phi_{\text{SEQ}(i)}} \) at both sides, we have,

\[
\frac{\phi_i}{\phi_{\text{SEQ}(i)}} W_{\text{SEQ}(i)}(t_1, t_2) \geq \frac{\phi_i}{\phi_{\text{SCH}}} W_{\text{SCH}}(t_1, t_2) - \frac{\phi_i}{\phi_{\text{SEQ}(i)}} \alpha_{\text{SEQ}(i)}
\]

\[
\therefore \quad W_i(t_1, t_2) \geq \frac{\phi_i}{\phi_{\text{SEQ}(i)}} W_{\text{SEQ}(i)}(t_1, t_2) - \alpha_i
\]

Adding the above and eliminate common terms, we have,

\[
W_i(t_1, t_2) \geq \frac{\phi_i}{\phi_{\text{SCH}}} W_{\text{SCH}}(t_1, t_2) - [\alpha_{\text{SEQ}(i)} \frac{\phi_i}{\phi_{\text{SEQ}(i)}} + \alpha_i]
\]

Comparing with Definition 4.1, we have,

\[
\alpha_{i,\text{VMSCH}} = \alpha_i + \frac{\phi_i}{\phi_{\text{SEQ}(i)}} \alpha_{\text{SEQ}(i)}
\]

Q. E. D.
Proof of Theorem 4.2

**Theorem 4.2:** For a sub-queue $i$ under VMSCH, if it is constrained by a leaky bucket $(\sigma, r_i)$, then the delay of any cell in the sub-queue is bounded by $D_{i,VMSCH}$,

$$D_{i,VMSCH} = D_{i,SEQ} + \frac{\alpha_{SEQ}(i)}{r_{SEQ}(i)} \quad \text{where} \quad r_{SEQ}(i) = \frac{\phi_{SEQ}(i)}{\phi_i}$$

where $\alpha_{SEQ}$ is the B-WFI for the virtual queue of sub-queue $i$; and

$r_{SEQ}(i)$ is the guaranteed rate for sub-queue $i$ from its virtual queue.

Consider the $k^{th}$ cell of sub-queue $i$, let $a_k$ and $d_k$ be its arrival and departure time respectively. From Definition 4.2, there exists an instant $t_1$ within the virtual queue busy period that includes also $d_k$ where $t_1 < d_k$, $Q_i(t_1^-) = 0$, and $Q_i(t_1^+) \neq 0$ hold, such that,

$$W_i(t_1, d_k^+) \geq \phi_i \frac{\phi_{SEQ}(i)}{\phi_{SEQ}(i)} W_{SEQ}(t_1, d_k^+) - \gamma_{i,SEQ}$$

Since both $t_1$ and $d_k$ are in the same busy period, virtual queue of $i$ is continuously backlogged when SCH is worst-case fair with virtual queue at SEQ, assuming no snooping collision. Thus, we have,

$$W_{SEQ}(t_1, d_k^+) \geq \phi_i \frac{\phi_{SEQ}(i)}{\phi_{SEQ}(i)} W_{SCH}(t_1, d_k^+) - \alpha_{SEQ}(i)$$

Substituting this into above, we have,

$$W_i(t_1, d_k^+) \geq \frac{\phi_i}{\phi_{SCH}} W_{SCH}(t_1, d_k^+) - \left[\frac{\phi_i}{\phi_{SEQ}(i)} \alpha_{SEQ}(i) + \gamma_{i,SEQ}\right]$$

Comparing with Definition 4.2, we have,

$$\gamma_{i,VMSCH} = \gamma_{i,SEQ} + \frac{\phi_i}{\phi_{SEQ}(i)} \alpha_{SEQ}(i)$$

From Lemma 4.1,

$$r_i D_{i,VMSCH} - \sigma_i = r_i D_{i,SEQ} - \sigma_i + \frac{\phi_i}{\phi_{SEQ}(i)} \alpha_{SEQ}(i)$$

$$\Rightarrow \quad D_{i,VMSCH} = D_{i,SEQ} + \frac{\alpha_{SEQ}(i)}{r_{SEQ}(i)} \quad \text{where} \quad r_{SEQ}(i) = \frac{\phi_{SEQ}(i)}{\phi_i}$$

Q. E. D.
Appendix B

Proof of Lemma 6.1

Lemma 6.1: \( \forall \bar{X} \in \mathbb{R}^n, \text{if } \bar{X}' < POE, \bar{X}^{i+1} \in POE. \)

From the algorithm given in Section 6.2.3.1, we have,

\[
x'_k = A s_k + x'_k \quad \text{where} \quad s_k = \frac{m_k}{\sum m_j}.
\]

Now,

\[
\|\bar{X}^{i+1}\| = (A s_1 + x'_1) + (A s_2 + x'_2) + \ldots + (A s_n + x'_n)
\]

\[
= A + \|\bar{X}'\| = \Phi \quad (\because \quad A = \Phi - \|\bar{X}'\|)
\]

Therefore, \( \bar{X}^{i+1} \in POE. \)

Q. E. D.
Proof of Lemma 6.2

Lemma 6.2: \( \forall \bar{X} \in \mathbb{R}^n, \text{if } \bar{X} \in ROA, \bar{X}^{i+1} \in ROA. \)

Since \( \bar{X}' \in ROA \), \( \therefore \| \bar{X}' \| \leq \Phi \).

For \( \| \bar{X}' \| < \Phi \) (i.e. \( \bar{X}' \in POE \)), from Lemma 6.1, we have,

\[ \bar{X}^{i+1} \in POE \quad \Rightarrow \| \bar{X}^{i+1} \| = \Phi \] ............................................(a)

For \( \| \bar{X}' \| = \Phi \) (i.e. \( \bar{X}' \in POE \)), \( \forall x_i' \in \bar{X}' \),

\[ x_i^{i+1} = \begin{cases} \max\{x_i' - \varepsilon, g_i\} & \text{if } x_i' > g_i \\ x_i' & \text{if } x_i' \leq g_i \end{cases} \]

Therefore, \( x_i^{i+1} \geq g_i \) ......................................................(b)

Now, if there are \( k \)'s \( x_i' > g_i \),

\[ \| \bar{X}^{i+1} \| \geq \| \bar{X}' \| - k\varepsilon \]

\[ \geq \Phi - k\varepsilon \] ..........................................................(c)

Also, if \( x_i' \leq g_i \),

\[ \| \bar{X}^{i+1} \| = \| \bar{X}' \| \leq \Phi \] ..................................................(d)

From Definition 6.6, (c) & (d), we have, \( \bar{X}^{i+1} \in ROA \).

Q. E. D.
Proof of Lemma 6.3

Lemma 6.3a: \[ \forall \, \bar{X} \in \mathbb{R}^{n+}, \text{ if } \bar{X}' \in \text{POE}, \text{ for any } \delta > 0, \exists \, N, \text{ such that}, \]
\[ |\bar{X}^{2k+\varepsilon} - \bar{G}| < \delta \text{ for all } k > N. \text{ Or simply, } \lim_{n \to \infty} \bar{X}^{2k+\varepsilon} = \bar{G}. \]

The following proof illustrates the two-dimensional case where there are only two flows in the system. The proof for the higher dimensional case is left for future work.

Given that \( \bar{X}' = (x', y') \in \text{POE}, \) i.e. \( x' + y' = \Phi, \)
suppose \( x' < g_x = \Phi S_x \) \& \( y' > g_y = \Phi S_y, \) where \( S_i = \frac{m_i}{m_x + m_y} \) and \( \bar{G} = (g_x, g_y), \)
and, from the algorithm given in Section 6.2.3.1, we have,
\[ x'' = x' \]
\[ y'' = \Phi S_y \]
\[ x'^{s+1} = x'' + A'' S_x \]
where \( A'' = \Phi - x'' - y'' = \Phi - x' - \Phi S_y = \Phi S_x - x' \)
\[ = x' + \Phi S_x^2 - x'S_x \]
\[ = x'S_x + \Phi S_x^2 \]
\[ y'^{s+1} = y'' + A'' S_y \]
\[ = \Phi S_y + \Phi S_x S_y - x'S_y \]
\[ = \Phi S_y (1 + S_x) - x'S_y \]
\[ x'^{s} = x'^{s+1} = x'S_y + \Phi S_x^2 \]
\[ y'^{s} = \Phi S_y \]
Generalising $x$ to $2k + i$, we have,

\[ x^{2k+i} = S^i y + \Phi S^2 y (1 + S + \ldots + S^k) \quad \text{........ (\dagger)} \]

\[ y^{2k+i} = \Phi S^k y (1 + S + \ldots + S^k) - x^i S^2 y \quad \text{........ (\ddagger)} \]

Now, for $k \to \infty$, since $1 < S < 0$, we have,

\[ x^\infty = S^i y + \Phi S^2 y \frac{1}{1 - S^k} = \Phi S^i y = g_x \quad \text{and,} \]

\[ y^\infty = \Phi S^i y = g_y \]

Q. E. D.
Lemma 6.3b: (Rate of Convergence)

\[ \forall \tilde{X} \in \mathbb{R}^2^+, \text{ if } \tilde{X}' \in POE, \text{ for any } \delta > 0, |\tilde{X}^{2k^{+}} - \tilde{G}| \text{ converges to } \delta \text{ with a rate of } S_2^k (g_1 - x_1') \log S_2, \text{ where } S_2 = \frac{m_2}{m_1 + m_2}. \]

From (*) and (**) in the Proof of Lemma 6.3a, we have,

\[ x^{2k^{+}} = S_x^k x' + \Phi S_x^2 (1 + S_x + ... + S_x^{k-1}) \]
\[ = S_x^k x' + \Phi S_x^2 \left( \frac{1 - S_x^k}{1 - S_x} \right) \]
\[ = \Phi S_x - (\Phi S_x - x')S_x^k \]
\[ = g_x - (g_x - x')S_x^k \]

and,

\[ y^{2k^{+}} = \Phi S_y (1 + S_y S_y^{k-1}) - x'S_y^k \]
\[ = g_y + (g_x - x')S_y^k \]

Let \( \delta = |\tilde{X}^{2k^{+}} - \tilde{G}| \), i.e., \( \delta = (g_x - x')S_y^k \),

Therefore, the rate of change of \( \delta \) w.r.t. \( k \) is,

\[ \frac{\Delta \delta}{\Delta k} = S_y^k (g_x - x') \log S_y \]

Q. E. D.
Proof of Theorem 6.1

**Theorem 6.1:** (Convergence theorem)
Starting at any arbitrary state of the network, if the above assumptions hold, the control algorithm converges to the PMM fair allocation.

From Lemma 6.1 - 6.3, for any $\tilde{X}' < POE$, $\tilde{X}$ converges to $G$ in $2N + 1$ iterations.

Now, assume a set of flows $F$ passes through a set of links $L$, where $L = \bigcup_{i=1}^{M} l_i$, i.e. $L$ contains $M$ segments of links $l$. Notice that each operation will bring at least one node to become bottlenecked. Each link $l_i \in L$ does indeed converge to $G_i$. Therefore, the algorithm converges in any order of links until the whole network is stable and the allocation is PMM fair. The number of iterations is bounded by $M(2N + 1)$ since each link takes $2N + 1$ iterations to converge.

Q. E. D.
Admission control: the process of deciding whether a newly arriving request for service can be granted.

Anypoint-to-anypoint: a union of point-to-multipoint, multipoint-to-point and multipoint-to-multipoint.

BA classifier: a classifier that selects packets based only on the contents of the DS field.

Boundary link: a link connecting the edge nodes of two domains.

Classifier: an entity which selects packets based on the content of packet headers according to defined rules.

Downstream DS domain: the DS domain downstream of traffic flow on a boundary link.

Dropper: a device that performs dropping.

Dropping: the process of discarding packets based on specified rules.

DS behavior aggregate: a collection of packets with the same DS codepoint crossing a link in a particular direction.

DS boundary node: a DS node that connects one DS domain to a node either in another DS domain or in a domain that is not DS-capable.

DS codepoint: a specific value of the DSCP portion of the DS field, used to select a PHB.

DS domain: a DS-capable domain; a contiguous set of nodes which operate with a common set of service provisioning policies and PHB definitions.

DS egress node: a DS boundary node in its role in handling traffic as it leaves a DS domain.

DS field: the IPv4 header TOS octet or the IPv6 Traffic Class octet when interpreted in conformance with the definition given in [67]. The bits of the DSCP field encode the DS codepoint, while the remaining bits are currently unused.

DS ingress node: a DS boundary node in its role in handling traffic as it enters a DS domain.

DS interior node: a DS node that is not a DS boundary node.

DS node: a DS-compliant node.

DS region: a set of contiguous DS domains which can offer differentiated
services over paths across those DS domains.

**DS Service**  
the overall treatment of a defined subset of a customer's traffic within a DS domain or end-to-end.

**DS-capable**  
capable of implementing differentiated services as described in this architecture; usually used in reference to a domain consisting of DS-compliant nodes.

**DS-compliant**  
enabled to support differentiated services functions and behaviors as defined in [67], this document, and other differentiated services documents; usually used in reference to a node or device.

**End-to-end behavior**  
a description of the behavior that results if all network elements along the path offer the same service, invoked with a defined set of parameters.

**Exported information**  
the information which must be collected and exported by the service module. Exported information is available to other modules of the network element, and by extension to setup protocols, routing protocols, network management tools, and the like.

**Flow**  
a sequence of packets that are sent from source(s) to destination(s). These packets are forwarded either in unicast or multicast fashion with a specified QoS through a set of nodes.

**Flow aggregation / aggregate**  
flows with certain characteristics are grouped together in L-3.

**Flow merging**  
a group of flows is forwarded using the same label in L-2.

**Forwarding mechanism**  
a specific algorithm or operation (e.g., queuing discipline) that is implemented in a node to realize a set of one or more per-hop behaviors.

**"free-riding" problem**  
is where users reserving no or less resource would share a QoS VC with more resources which are originally established for other users.

**Global synchronization problem**  
For aggregated TCP flows, an abrupt change in dropping threshold may cause many packets to be dropped at the same time because TCP traffic is bursty in nature. Eventually, it may trigger all TCP sources to back off and results in a poor overall throughput.

**Goodput**  
\[(\text{the data received in every } T \text{ interval}) / T\]

**Invocation information**  
the set of parameters required by a service module to invoke the service, and a description of how the parameter values are used by the service module.

**IS Behavior**  
the QoS-related end-to-end performance seen by an
application session. This behavior is the end result of composing the services offered by each network element along the path of the application's data flow.

**IS Service**
a named and coordinated set of QoS control capabilities provided by a single network element. The definition of an IS service includes a specification of the functions to be performed by the network element, the information required by the element to perform these functions, and the information made available by the element to other elements of the system. A service is conceptually implemented within the "service module" contained within the network element.

**L-2 label**
a VPI/VCI in ATM.

**Legacy node**
a node which implements IPv4 Precedence as defined in [86,87] but which is otherwise not DS-compliant.

**Marker**
a device that performs marking.

**Marking**
the process of setting the DS codepoint in a packet based on defined rules; pre-marking, re-marking.

**Meter**
a device that performs metering.

**Metering**
the process of measuring the temporal properties (e.g., rate) of a traffic stream selected by a classifier. The instantaneous state of this process may be used to affect the operation of a marker, shaper, or dropper, and/or may be used for accounting and measurement purposes.

**MF Classifier**
a multi-field (MF) classifier which selects packets based on the content of some arbitrary number of header fields; typically some combination of source address, destination address, DS field, protocol ID, source port and destination port.

**Microflow**
a single instance of an application-to-application flow of packets which is identified by source address, source port, destination address, destination port and protocol id.

**Native cut-through forwarding**
the ability to switch an incoming cell immediately, assuming the outgoing VC is not transmitting another cell and the incoming cell is traffic compliant.

**Network element**
is any component of an internetwork which directly handles data packets and thus is potentially capable of exercising QoS control over data flowing through it. Network elements include routers, subnetworks, and end-node operating systems.

**Per-Hop-Behavior (PHB)**
the externally observable forwarding behavior applied at a DS-compliant node to a DS behavior aggregate.
| **PHB group** | a set of one or more PHBs that can only be meaningfully specified and implemented simultaneously, due to a common constraint applying to all PHBs in the set such as a queue servicing or queue management policy. A PHB group provides a service building block that allows a set of related forwarding behaviors to be specified together (e.g., four dropping priorities). A single PHB is a special case of a PHB group. |
| **Policing** | the set of actions triggered when a flow's actual data traffic characteristics exceed the expected values given in the flow's traffic specification. |
| **Pre-mark** | to set the DS codepoint of a packet prior to entry into a downstream DS domain. |
| **Provider DS domain** | the DS-capable provider of services to a source domain. |
| **Rate-dependent error** | the delay a datagram in the flow might experience due to the rate parameters of the flow. An example of such an error term is the need to account for the time taken serializing a datagram broken up into ATM cells, with the cells sent at a frequency of 1/rate. |
| **Rate-independent, per-element error** | the worst case non-rate-based transit time variation through the service element. |
| **Re-mark** | to change the DS codepoint of a packet, usually performed by a marker in accordance with a TCA. |
| **Routing Support for Resource Reservation** | RSRR includes an interface to the multicast routing code (mrouted) which is based on DVMRP protocol and is distributed by Xerox PARC. |
| **Service Level Agreement (SLA)** | a service contract between a customer and a service provider that specifies the forwarding service a customer should receive. A customer may be a user organization (source domain) or another DS domain (upstream domain). A SLA may include traffic conditioning rules which constitute a TCA in whole or in part. |
| **Service Provisioning Policy** | a policy which defines how traffic conditioners are configured on DS boundary nodes and how traffic streams are mapped to DS behavior aggregates to achieve a range of services. |
| **Service Request Specification (RSpec)** | a specification of the quality of service a flow wishes to request from a network element. The contents of a service request specification is highly specific to a particular service. As examples, these specifications might contain information about bandwidth allocated to the flow, maximum delays, or packet loss rates. |
Setup protocol is used to carry QoS-related information from the end-nodes requesting QoS control to network elements which must exercise that control, and to install and maintain to required QoS control state in those network elements. A setup protocol may also be used to collect QoS-related information from interior network elements along an application's data flow path for ultimate delivery to end nodes. Examples of protocols which perform setup functions are RSVP [36], ST-II [88], and Q.2931.

Shaper a device that performs shaping.

Shaping the process of delaying packets within a traffic stream to cause it to conform to some defined traffic profile.

Slack term the difference between the desired delay and the delay obtained by using a reservation level. This slack term can be utilized by the network element to reduce its resource reservation for this flow.

Snooper cell a cell selected from the next highest instantaneous priority queue and tagged with a special identifier (msb of VPI) at its header.

Source DS domain a domain which contains the node(s) originating the traffic receiving a particular service.

Token Bucket a particular form of traffic specification consisting of a "token rate" r and a "bucket size" b. Essentially, the r parameter specifies the continually sustainable data rate, while the b parameter specifies the extent to which the data rate can exceed the sustainable level for short periods of specifically, the traffic must obey the rule that over all time. More periods, the amount of data sent cannot exceed rT+b, where T is the length of the time period.

Traffic conditioner an entity which performs traffic conditioning functions and which may contain meters, markers, droppers, and shapers. Traffic conditioners are typically deployed in DS boundary nodes only. A traffic conditioner may re-mark a traffic stream or may discard or shape packets to alter the temporal characteristics of the stream and bring it into compliance with a traffic profile.

Traffic conditioning control functions performed to enforce rules specified in a TCA, including metering, marking, shaping, and policing.

Traffic Conditioning Agreement (TCA) an agreement specifying classifier rules and any corresponding traffic profiles and metering, marking, discarding and/or shaping rules which are to apply to the traffic streams selected by the classifier. A TCA encompasses all of the traffic conditioning rules explicitly specified within a
SLA along with all of the rules implicit from the relevant service requirements and/or from a DS domain's service provisioning policy.

**Traffic profile**

a description of the temporal properties of a traffic stream such as rate and burst size.

**Traffic Specification (TSpec)**

a description of the traffic pattern for which service is being requested. In general, the TSpec forms one side of a "contract" between the data flow and the service. Once a service request is accepted, the service module has agreed to provide a specific QoS as long as the flow's data traffic continues to be accurately described by the TSpec.

**Traffic stream**

an administratively significant set of one or more microflows which traverse a path segment. A traffic stream may consist of the set of active microflows which are selected by a particular classifier.

**Upstream DS domain**

the DS domain upstream of traffic flow on a boundary link.
References


[23] A. Ghanwani, J. Pace, V. Srinivasan, A. Smith, M. Seaman, "A Framework for Providing Integrated Services Over Shared and Switched IEEE 802 LAN
Technologies", IETF Internet Draft <draft-ietf-issll-is802-framework-05.txt>, May
1998.

RFC 2022, November 1996.
Resolution Protocol (NHRP)", IETF RFC 2332, April 1998.
[27] The ATM Forum, LAN Emulation over ATM, Version 1.0 Specification, af-lane-
0021.000, January 1995.
[28] The ATM Forum, LAN Emulation over ATM, Version 2 - LUNI Specification, af-
lane-0084.000, December 1996.
[29] The ATM Forum, Multi-Protocol Over ATM Specification v1.0, af-mpoa-
0087.000, July 1997.
[31] E. Crawley, L. Berger, S. Berson, F. Baker, M. Borden, J. Krawczyk, "A
Framework for Integrated Services and RSVP over ATM", IETF RFC 2382, August
1998.
[32] L. Berger, "RSVP over ATM Implementation Requirements", IETF RFC 2380,
August 1998.
[33] L. Berger, "RSVP over ATM Implementation Guidelines", IETF RFC 2379, August
1998.
[34] The ATM Forum, Private Network-Network Interface Specification Version 1.0, af-
pnni-0055.000, March 1996.
to ATM", in Proceedings of IEEE INFOCOM, March 1996,
http://www.ipsilon.com/aboutipsilon/staffpages/pn/papers/infocom96.ps

182


[40] D. Hoffman, M. Speer, Solaris RSVP, Sun Microsystems, ftp://playgroun.sun.com/pub/rsvp/SolarisRSVP.0.4.9


[60] UCB/LBNL/VINT, Network Simulator - ns (version 2), http://www-mash.cs.berkeley.edu/ns/


