The candidate confirms that the work submitted is his own and the appropriate credit has been given where reference has been made to the work of others.

I understand that failure to attribute material that is obtained from another source may be considered as plagiarism.

(Signature of student) ________________________________
Summary

This project uses analytical modelling and the simulation tool ns-2 to compare the performance of networks that use only IP and those using RSVP and the Integrated Services architecture. The main aim is to show that the RSVP is capable of enhancing the performance of plain IP networks.

Both analytical models presented in this report reveal that RSVP-enabled networks provide an increase in performance to individual traffic flows. This is achieved at the expense of traffic flows that do require so much resources from the network (e.g., bandwidth or end-to-end delay). In most cases, the simulation experiments and scenarios investigated also show that RSVP does offer QoS for individual traffic flows in IP networks. The QoS offered is at the expense of lower priority traffic.

In order for this project to be successful, it was stated at the start of the project that the following objectives must be completed:

• Learn about network protocols, in particular the IP, the Integrated Services architecture and the RSVP protocol.
• Learn about analytical modelling techniques and how they are be applied to networking.
• Learn how to use network simulation tools.
• Design a set of analytical models and simulation experiments to compare the performance of RSVP and plain IP networks.
• Evaluate and compare the results.

The project has achieved all the objectives and in some cases, they have been exceeded.
Acknowledgements

I would like to thank Karim Djemame for suggesting the subject of this project and for making himself available for help, and Graham Hardman for installing ns-2 as and when asked to. I am also very grateful to Larry Dowdy for the support, advice, and help he has given me throughout the project.

Finally, I would like to thank my housemates and parents for putting up with me for the last three years.
List of Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACK</td>
<td>Acknowledgement</td>
</tr>
<tr>
<td>APNIC</td>
<td>Asia Pacific Network Information Center</td>
</tr>
<tr>
<td>ARIN</td>
<td>American Registry for Internet Numbers</td>
</tr>
<tr>
<td>AS</td>
<td>Autonomous System</td>
</tr>
<tr>
<td>ATM</td>
<td>Asynchronous Transfer Mode</td>
</tr>
<tr>
<td>BGP</td>
<td>Border Gateway Protocol</td>
</tr>
<tr>
<td>CBR</td>
<td>Constant Bit Rate</td>
</tr>
<tr>
<td>CIDR</td>
<td>Classless InterDomain Routing</td>
</tr>
<tr>
<td>DARPA</td>
<td>Defence Advanced Research Projects Agency</td>
</tr>
<tr>
<td>DiffServ</td>
<td>Differentiated Services</td>
</tr>
<tr>
<td>DSCP</td>
<td>Differentiated Services Code Point</td>
</tr>
<tr>
<td>DVD</td>
<td>Digital Video Disk</td>
</tr>
<tr>
<td>FEC</td>
<td>Forward Equivalence Class</td>
</tr>
<tr>
<td>FIFO</td>
<td>First In First Out</td>
</tr>
<tr>
<td>FTP</td>
<td>File Transfer Protocol</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hyper Text Transfer Protocol</td>
</tr>
<tr>
<td>ICANN</td>
<td>Internet Corporation for Assigned Names and Numbers</td>
</tr>
<tr>
<td>ICMP</td>
<td>Internet Control Message Protocol</td>
</tr>
<tr>
<td>IETF</td>
<td>Internet Engineering Task Force</td>
</tr>
<tr>
<td>IntServ</td>
<td>Integrated Services</td>
</tr>
<tr>
<td>IPv4</td>
<td>Internet Protocol Version 4</td>
</tr>
<tr>
<td>IPv6</td>
<td>Internet Protocol Version 6</td>
</tr>
<tr>
<td>ISO</td>
<td>International Standards Organisation</td>
</tr>
<tr>
<td>ISP</td>
<td>Internet Service Provider</td>
</tr>
<tr>
<td>LAN</td>
<td>Local Area Network</td>
</tr>
<tr>
<td>LDP</td>
<td>Label Distribution Protocol</td>
</tr>
<tr>
<td>LIFO</td>
<td>Last In First Out</td>
</tr>
<tr>
<td>LSP</td>
<td>Label Switched Path</td>
</tr>
<tr>
<td>LSR</td>
<td>Label Switched Router</td>
</tr>
<tr>
<td>MIT</td>
<td>Massachusetts Institute of Technology</td>
</tr>
<tr>
<td>MPL</td>
<td>Multi-Programming Level</td>
</tr>
<tr>
<td>MPLS</td>
<td>Multi Protocol Label Switching</td>
</tr>
<tr>
<td>NAM</td>
<td>Network AniMator</td>
</tr>
<tr>
<td>NS-2</td>
<td>Network Simulator</td>
</tr>
<tr>
<td>OSPF</td>
<td>Open Shortest Path First</td>
</tr>
<tr>
<td>OSI</td>
<td>Open Systems Interconnection</td>
</tr>
<tr>
<td>OTcl</td>
<td>Object-oriented Tool Command Language</td>
</tr>
<tr>
<td>PDA</td>
<td>Personal Digital Assistant</td>
</tr>
<tr>
<td>PHB</td>
<td>Per Hop Behaviour</td>
</tr>
<tr>
<td>QoS</td>
<td>Quality of Service</td>
</tr>
<tr>
<td>RIPE</td>
<td>Reseaux IP European</td>
</tr>
<tr>
<td>RFC</td>
<td>Request For Comments</td>
</tr>
<tr>
<td>RIP</td>
<td>Resource Information Protocol</td>
</tr>
<tr>
<td>RR</td>
<td>Round Robin</td>
</tr>
<tr>
<td>RSpec</td>
<td>Request Specification</td>
</tr>
<tr>
<td>RSVP</td>
<td>Resource ReserVation Protocol</td>
</tr>
<tr>
<td>Acronym</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>------------------------------------</td>
</tr>
<tr>
<td>RTI</td>
<td>Real Time Intolerant</td>
</tr>
<tr>
<td>RTT</td>
<td>Real Time Tolerant</td>
</tr>
<tr>
<td>TCP</td>
<td>Transfer Commission Protocol</td>
</tr>
<tr>
<td>ToS</td>
<td>Type of Service</td>
</tr>
<tr>
<td>TSpec</td>
<td>Traffic Specification</td>
</tr>
<tr>
<td>SLA</td>
<td>Service Level Agreement</td>
</tr>
<tr>
<td>SMTP</td>
<td>Simple Mail Transfer Protocol</td>
</tr>
<tr>
<td>TCA</td>
<td>Traffic Conditioning Agreements</td>
</tr>
<tr>
<td>Tcl</td>
<td>Tool Command Language</td>
</tr>
<tr>
<td>TELNET</td>
<td>Virtual Terminal</td>
</tr>
<tr>
<td>ToS</td>
<td>Type of Service</td>
</tr>
<tr>
<td>UDP</td>
<td>User Datagram Protocol</td>
</tr>
<tr>
<td>VINT</td>
<td>Virtual InterNet Testbed</td>
</tr>
<tr>
<td>WFQ</td>
<td>Weighted Fair Queueing</td>
</tr>
<tr>
<td>YATS</td>
<td>Yet Another Tiny Simulator</td>
</tr>
</tbody>
</table>
Contents

1 Introduction
1.1 Project Aim ................................................................. 1
1.2 Minimum Requirements ................................................... 2
1.3 Deliverables .............................................................. 2
1.4 Report Structure ......................................................... 3

2 Background Research
2.1 Introduction ............................................................ 5
2.2 Protocol Architectures .................................................. 5
2.3 OSI Reference Model ................................................... 6
2.4 TCP/IP Networks ......................................................... 8
  2.4.1 IP Networks and their limitations ................................. 11
    2.4.1.1 IP Network Structure ......................................... 12
    2.4.1.2 IP Addressing ................................................ 12
    2.4.1.3 IP Routing .................................................. 14
2.5 Quality of Service ..................................................... 15
  2.5.1 What is Quality of Service? ....................................... 15
  2.5.2 Measuring Quality of Service ................................... 15
  2.5.3 The Need for Quality of Service ................................. 16
2.6 Protocols that provide Quality of Service .............................. 17
  2.6.1 Differentiated Services ........................................... 18
  2.6.2 Multi Protocol Label Switching ................................. 19
  2.6.3 Integrated Services .............................................. 22
    2.6.3.1 Introduction ............................................... 22
    2.6.3.2 Approach .................................................. 22
    2.6.3.3 Service Models ........................................... 25
    2.6.3.4 Classes of Service ........................................ 26
      2.6.3.4.1 Guaranteed Service ................................... 26
      2.6.3.4.2 Controlled Load ..................................... 27
      2.6.3.4.3 Best Effort .......................................... 27
    2.6.3.5 Resource Sharing ......................................... 28
  2.6.3.6 Scheduling ................................................... 28
  2.6.3.7 Policing .................................................... 29
  2.6.3.8 Resource reSerVation Protocol ............................ 30

3 Methodology
3.1 Introduction .......................................................... 34
  3.1.1 Empirical Analysis .............................................. 34
  3.1.2 Analytical Modelling .......................................... 34
  3.1.3 Simulation ...................................................... 35
  3.1.4 Chosen Methodologies ......................................... 35
3.2 Queuing Models ....................................................... 35
3.3 Markov Models ....................................................... 37
3.4 Simulation Tools ..................................................... 38
  3.4.1 ns-2 ........................................................... 38
  3.4.2 OPNET Modeler ................................................ 38
F.3 Packet Loss .................................................................116
F.4 Throughput .............................................................117
## List of Figures

2.1 Example of How the OSI Model is Used ................................................................. 8  
2.2 OSI and TCP/IP Reference Models ................................................................. 9  
2.3 TCP Header ........................................................................................................ 10  
2.4 UDP Header ........................................................................................................ 10  
2.5 Networks and Protocols in the TCP/IP Model ...................................................... 11  
2.6 IPv4 Header ........................................................................................................ 11  
2.7 Internet Address Classes .................................................................................... 13  
2.8 IPv6 Header ........................................................................................................ 14  
2.9 ISPN Router .......................................................................................................... 25  
2.10 FIFO and WFQ Comparison ................................................................................ 29  

3.1 Queuing Model ..................................................................................................... 36  
3.2 Hierarchy of Objects in ns-2 ............................................................................... 42  

4.1 Simple Abstract View of a Router ......................................................................... 45  
4.2 Abstract View of an IP Router ............................................................................... 46  
4.3 IP Router System .................................................................................................. 47  
4.4 Simplistic Markov Model for the IP Router .......................................................... 48  
4.5 Abstract view of an ISPN Router ......................................................................... 49  
4.6 ISPN Router System ............................................................................................ 52  
4.7 Markov Model for the ISPN Router ..................................................................... 53  

5.1 Simulation Topology ............................................................................................ 58  

6.1 Average Response Times for the IP and ISPN Models ......................................... 61  
6.2 Average Response Times per Traffic Class ......................................................... 62  
6.3 Average Queue Length for the IP and ISPN Models ............................................. 63  
6.4 Average Queue Length per Traffic Class ........................................................... 63  
6.5 Average Throughput for the IP and ISPN Models ............................................... 64  
6.6 Average Throughput per Traffic Class ............................................................... 64  
6.7 Power for the IP and ISPN Models ...................................................................... 65  
6.8 Power per Traffic Class ....................................................................................... 66  
6.9 Average End-to-End Delay for the IP and ISPN Topology .................................... 67  
6.10 Average End-to-End Delay per IP Flow ............................................................. 67  
6.11 Average End-To-End Delay per ISPN Flow ....................................................... 68  
6.12 Throughput for the IP and ISPN Topology ......................................................... 69  
6.13 Throughput per IP Flow ..................................................................................... 69  
6.14 Throughput per ISPN Flow .............................................................................. 70  
6.15 Packet Loss for the IP and ISPN Networks ......................................................... 71  
6.16 Packet Loss per IP Flow ..................................................................................... 71  
6.17 Packet Loss per ISPN Flow .............................................................................. 72  
6.18 Average End-To-End Delay per IP Flows ......................................................... 73  
6.19 Average End-To-End Delay per ISPN Flow ...................................................... 73  
6.20 Average End-To-End Delay for IP and ISPN Networks .................................... 74  
6.21 Throughput per IP Flow ..................................................................................... 75
6.22 Throughput per ISPN Flow ..........................................................75
6.23 Packet Loss for IP and ISPN Networks ........................................76
6.24 Average End-To-End Delay per IP Flow ......................................77
6.25 Average End-To-End Delay per ISPN Flow .................................77
6.26 Average End-To-End Delay for IP and ISPN Networks ...............78
6.27 Throughput per ISPN Flow .........................................................78
6.28 Throughput per IP Flows ............................................................79
6.29 Throughput for IP and ISPN Networks .........................................80
6.30 Packet Loss per IP Flow ..............................................................80
6.31 Packet Loss per ISPN Flow .........................................................81
6.32 Packet Loss for IP and ISPN Networks .................................81
List of Tables

1.1 Project Schedule Outline ..............................................................3

2.1 Number of Networks and Hosts per Address Class ..........................13

4.1 Traffic Class Numbers ................................................................50
4.2 Queues for Traffic Classes .............................................................50
4.3 Parameter Values for Analytical Models ...........................................54

D.1 Results for IP and ISPN Routers as a Whole ................................107
D.2 Results for IP and ISPN Routers per Packet Class ..........................107
Chapter 1

Introduction

1.1 Project Aim

The Internet can be traced back to the ARPANET, an experiment of a new technology known as packet switching. Today, the need for a reliable and efficient Internet is becoming increasingly more important as the ability to be able to share information and data across networks is becoming ever more prominent. A reflection of this is the vast amount of research and development being conducted in guaranteeing Quality of Service (QoS) to end-users.

Network protocol suites are the foundations of the networking infrastructure that society has have become so dependent on. Over the past 30 years, the Internet Protocol (IP) has remained virtually unchanged. This project will investigate the performance of networks under this protocol and highlight advantages made available by using a more advanced protocol known as the ReSerVation Protocol (RSVP).

Integrated Services (IntServ) is a set of standards set by the Internet Engineering Task Force (IETF). It assures different classes of traffic QoS when travelling through a network. In this model, applications signal the intermediate network nodes explicitly to reserve resources to meet the QoS characteristics expected by a particular traffic flow. The signalling protocol used is called RSVP. RSVP is thus a protocol that implements the IntServ approach to networking.

The aim of the project is to carry out a performance comparison of the IP and RSVP protocols. Analytical models and a network simulation tool, together with a set of experiments designed to identify factors that affect their performance, will be used in the process of evaluating these protocols.

In order to achieve the aim a good understanding of computer networks and the protocols that run on them must be attained. An insight into simulation, analytical modelling
techniques and the role that they play in the performance evaluation of network protocols is also a necessity.

This final report, written in a language appropriate to this research area, details the findings (i.e. whether IntServ performs better than IP under various workloads) throughout this project.

1.2 Minimum Requirements

The minimum requirements for this project to succeed and ensure that the project aim is satisfied can be summarised as follows.

- Research and understand about network and Internet protocols.
- Learn and apply appropriate analytical modelling techniques.
- Learn and apply network simulation tools.
- Design analytical and simulation experiments to evaluate performance of RSVP and IP.
- Evaluate and compare results of simulations and analytical models to determine which protocol performs best under various workloads.

1.3 Deliverables

The major deliverable is this final report. In order to achieve the best quality result, a project schedule together with a number of sub-deliverables and deadlines have been formed and accomplished. They correspond to the minimum requirements and objectives stated above. The rationale is that each requirement is to be completed before progressing on to the next one.
<table>
<thead>
<tr>
<th>Deadline</th>
<th>Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>31/1/2003</td>
<td>Complete literature review</td>
</tr>
<tr>
<td>27/2/2003</td>
<td>Design analytical models</td>
</tr>
<tr>
<td></td>
<td>Learn how to use the network simulator</td>
</tr>
<tr>
<td>31/3/2003</td>
<td>Design simulation topologies</td>
</tr>
<tr>
<td></td>
<td>Design and carry out experiments</td>
</tr>
<tr>
<td>30/4/2003</td>
<td>Evaluate and compare results</td>
</tr>
<tr>
<td></td>
<td>Conclusion</td>
</tr>
<tr>
<td></td>
<td>Write up report</td>
</tr>
</tbody>
</table>

*Table 1.1 Project Schedule Outline*

1.4 Report Structure

The structure of the report reflects the minimum requirements given in 1.2 and the schedule outlined in Table 1.1. Chapter 2 begins with an introduction to network architectures and protocol stacks before discussing existing IP networks and their limitations. A detailed discussion on the need for QoS and the protocols that attempt to provide it are presented.

Chapter 3 discusses possible methodologies that are regularly used in the performance evaluation of network protocols. Reasons for having chosen analytical modelling and simulation as the methodologies in this project and details on the theory on which these methodologies are built are also presented. A brief description of the simulators available and the importance of Markov models in quantitative analytical techniques is discussed, while the chosen simulator, ns-2, its use, and how performance is measured and interpreted concludes the chapter.

Chapter 4 provides a guide as to how the analytical models for this project are constructed. This guide includes the assumptions made to keep the models tractable, a short description of the parameters chosen and the notation used throughout the report.

Techniques that are considered standard practice or useful when designing topologies for simulation are discussed in chapter 5. The set of experiments designed to compare the performance of the IP and RSVP protocols in ns-2 are presented in detail. Finally, the
performance metrics to determine whether RSVP improves performance on IP networks is also discussed.

The results and findings of the analytical models and simulation experiments are discussed extensively in chapter 6. This precedes the evaluation and conclusions drawn from the project. The evaluation and conclusion are included in chapter 7 and chapter 8 respectively.

Appendix A contains a reflection on the project along with lessons learnt and recommendations for anyone embarking on a similar project in the future. For the interested reader, appendices B and C give details on how the analytical models were solved and the performance results recorded. Appendix D presents the results in tabulated form while appendices E and F include the simulation scripts and AWK scripts used in ns-2.
Chapter 2

Background Research

2.1 Introduction

This chapter considers the complexity involved when attempting to exchange information between two computer systems. It introduces network architectures, protocols, protocol stacks, and their application to networking. Two examples of network architectures are summarised. These include a brief description of the Open Systems Interconnection (OSI) Reference Model and a more detailed look at the Transmission Control Protocol/Internet Protocol (TCP/IP) architecture.

A discussion of some of the limitations present in the TCP/IP architecture and the need for QoS is provided. Performance metrics involved in determining the level of QoS provided are listed. Towards the end of the chapter the fundamental differences of three protocols (i.e., IntServ, Differentiated Services (DiffServ), and Multi-Protocol Label Switching (MPLS)) that attempt to improve network performance and provide QoS are presented.

2.2 Protocol Architectures

When computers, terminals, wireless Personal Digital Assistants (PDAs), or other data processing devices exchange information, the procedures involved can be quite complex. To reduce the design complexity most networks are organised as a series of layers or levels, each built on top of each other. Even though the number, names, contents, and functions of layers differ from network to network, they all offer services to the higher level, while at the same time hiding the detail of how these services are actually implemented.

It takes two peers on different systems using the same set of layered functions to communicate. In order for layer \( n \) on one machine to carry out a conversation with its corresponding layer \( n \) on another machine, an agreement on how communication is to
proceed must be established. The set of rules and conventions that govern this communication procedure are called protocols. Protocols generate formatted blocks of data that obey these rules and conventions. The key features of protocols are commonly summarised in their syntax, semantics, and timing:

- **Syntax** - Concerns the format of the data blocks being exchanged
- **Semantics** - Addresses control information for co-ordination and error handling
- **Timing** - Includes speed matching and sequencing (Stallings, 2001)

An interface defines the primitive operations and services offered by the lower layer. The set of layers and protocols is called a network architecture and the list of protocols used by a system is referred to as its protocol stack.

NOTE: No data is actually transferred from layer $n$ on one machine to layer $n$ on another machine. Instead, each layer passes data and control information down the layers until the lowest layer (i.e., physical layer) is reached through which actual communication takes place.

### 2.3 OSI Reference Model

The OSI Reference Model is based on a proposal developed by the International Standards Organisation (ISO) in an attempt to internationally standardise the protocols used in the various layers (Tanenbaum, 1996). It consists of seven layers with each layer providing a different level of abstraction, performing a well-defined function, and minimising information flow across interfaces. The seven layers are:

- **Physical layer** - provides a physical (e.g., cables) communication channel for transferring actual bits.

- **Data Link layer** - provides error detection/correction codes for acknowledgements and data frames in point to point data transmission.
• **Network layer** - controls routing of packets, congestion control, and the interconnection of different networks.

• **Transport layer** - handles the flow control, quality of service, connection services (i.e. connectionless or connection oriented services), and the preparation and re-assembly of data packets.

• **Session layer** - establishes and manages entire user sessions to allow data transport (e.g., file transfer) and supports some element of synchronisation by introducing checkpoints in data streams to avoid retransmission in case transfer is aborted.

• **Presentation layer** - addresses the syntax and semantics of the information transmitted and manages abstract data structures.

• **Application layer** - contains protocols for email, remote logins, and a host of user specific applications.

These protocols were defined but not implemented as TCP/IP was already in widespread use at the time and the de facto standard. The session and presentation layers of TCP/IP were virtually empty and some functions (e.g., flow and error control) cut across different layers. The OSI model follows good software engineering principles as it distinguishes specification and implementation and provides a good guide when developing new network architectures. Thus, the OSI model is commonly used to describe any protocol stack (see Fig 2.1).
2.4 TCP/IP Networks

The TCP/IP model was developed at a similar time to the OSI Reference Model and was funded by the Defense Advanced Research Projects Agency (DARPA). Its development was conducted on the experimental packet-switched network ARPANET and was developed from a practical (i.e., implementation) point of view. Its tangible result makes TCP/IP the dominant protocol stack in modern communications and is the precursor of all computer networks including the Internet. The driving design principles and goals of TCP/IP provide:

- the interconnection of different existing networks,
- a resilience to breakdown (i.e. survivability of the loss of subnet hardware), and,
- functional flexibility.

This protocol stack has fewer layers than the OSI model and can be summarised as a packet switching network based on a connectionless internetwork layer (see Fig 2.2).
The Internet layer holds the whole architecture together and permits the host to inject packets of data into any network and have them travel independently of other packets. This is analogous to the current postal system whereby letters that are all sent to the same address from the same originating host, might follow different paths or routes. The protocol includes functionality to handle sequencing of related packets, lost packets, and packets of varying sizes. The official packet format and protocol called the Internet Protocol (IP) is defined to facilitate the delivery of packets, packet routing, and provide some level of congestion control.

The transport layer is designed to allow peer entities on source and destination hosts to communicate with each other. They provide an end-to-end service via two protocols. The Transmission Control Protocol (TCP) provides a reliable connection oriented protocol, assembles and reassembles packets, introduces flow control and sequencing, and provides the ability to recover from lost or damaged packets. The User Datagram Protocol (UDP) in turn provides an unreliable connectionless protocol that assembles and reassembles packets as well but does not provide flow control, sequencing, or protection against duplication of packets. TCP is designed for file transfer and remote logins, where reliability and accuracy in a connection oriented environment is important to the end user. UDP is designed for applications such as speech or video transmission, where prompt delivery is more important and where occasional dropped packets (i.e., accuracy) is less important in a connectionless environment. TCP and UDP headers are shown in figures 2.3 and 2.4 respectively.
The application layer contains the high level protocols that provide service to the applications being used by the end users. Examples of common protocols include the:

- virtual terminal protocol (TELNET) for logging on to distant machines,
- File Transfer Protocol (FTP) to provide an efficient method of transferring data from one machine to another,
- electronic mail protocol (SMTP) to facilitate a specialised form of file transfer, designed for the exchange of email and,

Other protocols are under continuing development and review as demands on networks increase and new applications emerge (see Fig 2.5).
### 2.4.1 IP Networks and their Limitations

The network layer of the Internet provides a connectionless datagram service. It consists of three major components:

- **IP**, that defines addressing conventions, a datagram format, and packet handling conventions,
- the path selection components (i.e., routing protocols) that determine the route a datagram follows from source to destination from information available in routing tables, and
- the Internet Control Message Protocol (ICMP) that is used by hosts, routers, and gateways to communicate network layer information to each other and to report errors in datagrams.

![IPv4 Header](image)

*Fig 2.5 Networks and Protocols in the TCP/IP Model*

*Fig 2.6 IPv4 Header*
Routing is thus vital in attempting to prevent congestion in networks and, in turn, determining the performance of networks. The limited functionality of IP routing provides no support for QoS. In fact, IP network nodes or routers provide a best effort service, whereby data is sent implicitly (i.e., whenever it can and without anybody’s permission). It has no means of guaranteeing delivery (Arindam, 2000). It is up to the end systems and transport protocols to verify that the packet goes through the network correctly and without error.

2.4.1.1 IP Network Structure

The Internet is comprised of interconnected networks of local, regional, national and international Internet Service Providers (ISP). Collections of routers that fall under the same administrative and technical control are known as an Autonomous System (AS). Each AS in turn is typically comprised of multiple networks.

Exterior gateway protocols, such as the Border Gateway Protocol (BGP) refer to protocols that are used for inter-AS routing. Interior gateway protocols, such as Open Shortest Path First (OSPF) and the Resource Information Protocol (RIP) are used for intra-AS routing (Kurose et al, 2001).

2.4.1.2 IP Addressing

IPv4 addressing is 32 bits long, giving a total of possibility of 4,294,967,276 IP addresses (see Fig 2.6). Addresses are written in dotted decimal notation whereby each byte of the address is written in decimal form and is separated by a period (‘dot’) from other bytes in the address (e.g., 193.32.216.9). Each interface on every host in the Internet has an IP address that is globally unique. The hierarchical routing architecture seen in the Internet due to AS will determine an interface’s IP address according to the network it is connected to.

The original Internet addressing architecture specifies four defined classes and one undefined class for future use (see Fig 2.7). Classes A, B and C, each consist of a network portion and a host portion. Class D is reserved for multicast addresses whilst class E is
reserved for future use. The number of networks and hosts that they can service are summarised below (see Table 2.1).

![Fig 2.7 Internet Address Classes](image)

<table>
<thead>
<tr>
<th>Address Class</th>
<th># of Networks</th>
<th># of Hosts</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>$2^7 = 128$</td>
<td>$2^{24} = 16,777,216$</td>
</tr>
<tr>
<td>B</td>
<td>$2^{14} = 16,384$</td>
<td>$2^{16} = 65,536$</td>
</tr>
<tr>
<td>C</td>
<td>$2^{21} = 2,097,152$</td>
<td>$2^8 = 256$</td>
</tr>
<tr>
<td>D</td>
<td>$2^{28} = 268,434,456$</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>$2^{27} = 134,217,728$</td>
<td></td>
</tr>
</tbody>
</table>

Table 2.1 Number of Networks and Hosts per Address Class

Rapid growth in the number of organisations with small and medium sized networks has led to a rapid depletion of the class B addresses space as class C networks could only accommodate 256 hosts (i.e., too small for many organisations). The number of class B hosts (i.e., 65,534) was too large for most organisations resulting in poor utilisation of the assigned address space (Kurose et al, 2001).

In 1993, the IETF removed the address classes and introduced a new standard of Classless Interdomain Routing (CIDR). Organisations were given a 21-bit network address and the remaining 11 bits were used to identify the specific hosts within the network using a procedure known as subnetting. This procedure creates its own network within a network.
IP addresses are managed under the authority of the Internet Corporation for Assigned Names and Numbers (ICANN). Regional Internet registries (e.g., American Registry for Internet Numbers (ARIN), Reseaux IP Europeans (RIPE), and the Asia Pacific Network Information Center (APNIC)) control IP address assignment.

With the explosive growth of the Internet, and of private networks attached to the Internet, the address length became insufficient to accommodate all of the systems needing addresses (Stalling, 2001). To meet this need, IPv6 was proposed by the IETF. It includes 128 bit source and destination address fields (see Fig 2.8).

![IPv6 Header](image)

**Fig 2.8 IPv6 Header**

### 2.4.1.3 IP Routing

When networks use IP, the notion of traffic flows is non-existent. Network administrators have little or no control whatsoever over the way data is physically routed. The main principle behind IP routing is that of the 'longest match' on the destination IP address (Kurose et al, 2001).

Routers will access the destination address and perform lookups on its routing table to determine the address of the next router to which to send the packet. Routing is done on a hop-by-hop basis whereby routers only contain information on how to get to other routers or nodes on their own AS or private networks. If the packet is destined for a node on the current network then the router can send it directly to the intended recipient.
Traffic engineering (i.e., the ability to monitor the network for periods of high activity and direct classes of traffic via different routers or paths to ensure traffic receives its required resources) is not available to network administrators. The fact that only the destination address is considered in routing limits the capabilities of IP traffic.

Traffic Engineering is desirable on any network. In order to achieve this, every router would need to know about every other router or node in its domain. This would lead to extremely large routing tables and buffer sizes on routers. It would take longer to direct a packet down a particular route slowing the overall network traffic flow and reducing the performance of the network considerably.

2.5 Quality of Service

2.5.1 What is Quality of Service?

QoS is the capability of a network to provide a guaranteed level of service to selected network traffic over various underlying technologies (Arindam, 2000) be it Frame Relay, Asynchronous Transfer Mode (ATM), IP, or routed networks. QoS is especially needed when transmission demand exceeds network capacity. QoS is not limited to one type of traffic (e.g., voice, video, HTTP, email) but rather encompasses the entire spectrum of packet types and how data networks prioritise and deliver them. It is this method of differentiating between classes of traffic and the way they are treated that influences how much of the performance is improved for selected traffic in the underlying network.

It is important to note that QoS across an entire network involves capabilities in both the end system software (e.g., the operating system) and the networks that carry the data back and forth from one host to another. This project deals with how network protocols are designed and how they provide this QoS. It is not the purpose of this project to address how QoS performance is guaranteed in the end system software.

2.5.2 Measuring Quality of Service
Arindam (2000) and Nortel Networks (1998) present various formal metrics to measure QoS. QoS (and any network design) seeks to maximise service availability and throughput by reducing delay and, in the process, minimise delay jitter, and packet loss. The primary metrics surrounding QoS include:

- **Service availability** - the reliability of users’ connections to internet devices,

- **Delay** - the time taken by the packet to travel through the network from its source to its destination,

- **Jitter** - the variation in delay encountered by similar packets following the same route through the network,

- **Throughput** - the rate at which packets traverse through the network, and

- **Packet loss rate** - rate at which packets are dropped, get lost, or become corrupted while being routed through the network.

### 2.5.3 The Need for Quality of Service

Stallings (2001) identifies significant trends that have altered the current role of personal computers. One of these is the explosive growth of computing power. Due to this explosive growth of computing power, platforms that support graphical, multimedia and data intensive applications and elaborate graphical user interfaces have been made readily available to the end user.

Secondly, organisations recognising the value of Local Area Networks (LAN) as a viable and essential computing platform, resulting in the focus on network computing and the introduction of client/server architectures. This, coupled with the increased performance in servers has thus shifted the bottleneck from the workstations to the network architecture. Finally, the rapid conversion of consumer electronics to digital technology (e.g., Digital VideoDisk (DVD), digital video cameras and digital still cameras) and the ability of
computers to process these applications created an explosive increase of traffic over the Internet.

IP is not designed to cope with the large resources required by high-end (or inelastic) traffic. Demands for a high amount of bandwidth and low delay are now imposed. Video pictures, audio traffic, and ensuring that data reaches its destination quickly is now of primary importance. Data arriving late often renders the application useless. Retransmission of lost or corrupt packets is often too costly or unnecessary since data arriving late is obsolete or of little value. Thus, data redundancy is incorporated if needed.

While this high-end (i.e., QoS) traffic is important, one cannot ignore the low-end (or elastic) traffic requirements that must still be considered. Elastic traffic can adjust to changes in delay and throughput across a network and still meet the needs of its applications. This is the more traditional traffic that TCP/IP networks are designed to support. Examples include FTP, TELNET, and conventional email.

To summarise, a fundamental change towards a QoS-based Internet service approach is inevitable. Without QoS, routers behave even-handedly with arriving IP packets. Therefore, as congestion develops, it is unlikely that resources will be allocated in such a way as to meet the QoS requirements of elastic or inelastic traffic in a fair way. New QoS routing protocols are needed to be developed, implemented, and analysed.

2.6 Protocols that provide Quality of Service

Increasing bandwidth capacity is not enough to cope with the demands imposed by high-end traffic. Sensible methods for managing the traffic and controlling congestion are needed.

In recent years, the IETF has set two different, but complementary standards for traffic management. These are the IntServ and DiffServ models. A further standard for which development was started by the IETF with the introduction of the MPLS working group also provides a framework for QoS management over IP.
IP can also run over Asynchronous Transfer Mode (ATM), a technology in use since its development in 1991. This standard also provides QoS in networks but is beyond the scope of this project.

### 2.6.1 Differentiated Services

The DiffServ architecture defined in RFC 2475 employs a small, well-defined set of building blocks to deliver end-to-end QoS in networks. It is designed to provide a simple, easy-to-implement, low overhead tool to support a range of network services that are differentiated on the basis of performance. DiffServ is sender-orientated meaning that any QoS is purely based on the requests of the sender.

To achieve this, a variety of aggregate behaviours (i.e., classes of traffic) are specified by a Differentiated Services Codepoint (DSCP). DSCPs are used to mark each packet. The packet will then receive a particular forwarding treatment, or per-hop-behaviour (PHB), at each network node. PHBs provide particular forwarding treatment according to the characteristics (e.g., packet loss or jitter) or resources required (e.g., bandwidth) by individual classes of traffic.

This aggregation or per-hop behaviour minimises signalling between nodes and in doing so eases congestion on the network. It also provides good scaling to large networks and traffic loads since all traffic with the same DSCP is handled in aggregate and not individually (Stallings, 2001).

The IPv4 Type of Service (ToS) octet (i.e., 8 bits or byte) or Traffic Class Field in IPv6 can be used to implement the DSCP. Two of the bits are reserved for future use allowing a total of 64 classes or aggregate behaviours to be specified at present. Using the IP ToS field contributes to DiffServ's efficiency and ease of deployment (Stallings, 2001).

A set of nodes or routers, that support a common PHB policy and set of service provision policies are provided within a DiffServ Domain. DiffServ domains have well defined boundaries and consist of two types of nodes, interior nodes and boundary nodes.
Interior nodes (i.e., nodes within the domain) map the DSCP of each packet into the set of PHBs and impart appropriate forwarding behaviour (Arindam, 2000). Edge routers or boundary nodes (i.e., nodes that connect to other DiffServ domains) enforce the Traffic Conditioning Agreements (TCA) between DiffServ domains. DSCPs are domain independent therefore, if a packet were to change domains on its route, then boundary nodes may rewrite the DSCP to match the same PHB in its new domain. In effect when traffic enters a DiffServ domain, boundary nodes are responsible for enforcing the TCA between the domain it is entering and the domain of the sender node. If traffic leaves a DiffServ domain, boundary nodes shape the outgoing traffic to make it compliant with the TCA between its own domain and the domain of the receiver node.

Typically, DiffServ domains are controlled by one administrative entity. The services provided across this domain are defined in a service level agreement (SLA). A SLA is a service level contract between customers and the service provider (Stallings, 2001). Once the SLA is established, customers submit packets with the DSCP marked to indicate the packet class. It is then up to the service provider to ensure that the customer gets the agreed QoS for each packet class. Any nodes residing within a DiffServ domain that are not DiffServ compliant result in unpredictable performance and a loss of end-to-end QoS (Arindam, 2000).

NOTE: customers may include either a user organisation or another DiffServ domain.

2.6.2 MultiProtocol Label Switching

DiffServ and IntServ provide enhancements to an IP based network through the support of QoS explicitly. The performance issue (i.e., how to improve the overall throughput and delay of an internet) is not addressed by these architectures (Stallings, 2001).

MPLS, defined in RFC3031, addresses this performance issue by attempting to integrate the control provided by the intelligent layer 3 protocols like IP with the fast simple switching of layer 2 protocols such as ATM (Dobson, 2002). This allows very fast forwarding of packets while retaining the flexibility of an IP based networking approach (Stallings, 2001).
Proprietary initiatives to merge IP and ATM technologies in the early 90s, such as IP Switching (developed by Ipsilon, now Nokia Telecommunications), Tag Switching (developed by Cisco Systems) and IP Navigator (developed by Cascade), led the IETF to establish a working group for MPLS in 1997 to develop a common standardised approach. Even though the late 90s saw the introduction of routers that were as fast as ATM switches, MPLS still manages to

- reduce the amount of per-packet processing required at each router,
- provide QoS,
- support virtual private networks and
- can work over IP, ATM, or frame relay based networks.

NOTE: This report focuses on the use of MPLS on IP based networks and not MPLS on frame relay or ATM based networks.

MPLS is based on the use of labels to route packets through an MPLS domain. Each packet is assigned a label according to the packet’s Forwarding Equivalence Class (FEC). A FEC is a distinct flow of packets between two endpoints, or in the case of multicast, a flow of packets between a single source and multiple receivers (Stallings, 2001).

Packets are classified into their FEC only once. The ingress router performs this task when a packet first enters an MPLS domain. The packet then maintains the same FEC throughout its life within the MPLS domain. As packets leave the MPLS domain an egress router strips them of their labels and the packet is forwarded to its next destination according to the existing protocol.

When multiple packets are classified into the same FEC, the set of nodes within an MPLS domain, called Label Switched Routers (LSR), have no way of distinguishing between them. Packets are therefore routed via the same path even if their destinations differ (Dobson, 2002).

Labels are assigned once the ingress router has determined the path that defines the FEC through the network. The path defined for the FEC reflects the QoS requirements of the
FEC. Such paths are called Label Switched Paths (LSP) and make MPLS a connection-oriented technology.

Subsequent LSRs along the LSPs need not examine the destination address of a packet or use complex search and routing algorithms to calculate the packet’s next hop. Instead they simply use the label to index into the LSR routing tables to retrieve the address of the next node and a new value for the label (i.e., similar to switching). The label thus provides both a simple and far less complex method of forwarding packets than in standard IP routers. It also provides a much faster method of forwarding packets.

Labels are locally significant in the sense that adjacent routers agree upon using a designated label for a particular FEC among them. This allows for identical labels to distinguish between different FECs by more than one pair of routers in an MPLS domain. The label is, in fact, a function of the packets Class of Service and the FEC. Ideally, the function should be a one-to-one mapping between labels and FECs to prevent confusion when traffic needs to be sent between two non-adjacent LSRs (Arindam, 2000).

This label-based forwarding approach requires LSRs to understand the MPLS protocol and at the same time be ‘aware’ of the other LSRs within the MPLS domain. The former requires some software implementation at the routers making it almost trivial to enable MPLS support in IP networks. The latter requires control information to be exchanged periodically between LSRs. This is carried out by a distribution protocol. RFC3031 describes the most common protocol used, the Label Distribution Protocol (LDP).

Multiple labels can also be combined to form a label stack and provide MPLS with powerful Traffic Engineering capabilities. These labels are organised as a Last-In-First-Out (LIFO) stack and can be used for tunnelling (i.e., a specific LSP defined for a particular packet flow or FEC). Network operators are thus given the facility of being able to direct traffic across a specific set of LSRs in order to measure the performance across these nodes or avoid congested routes (Dobson, 2002). This offers improved services to customers so Internet Service Providers (ISP) frequently deploy MPLS in the cores of large networks.

Due to the simplicity of its implementation, MPLS offers a much more scalable, responsive and simpler network design when compared to IP networks and other protocols providing
QoS. The ability of MPLS and standard IP networks to co-exist together efficiently allows the migration from one network to another to take place over a period of time without affecting the networks performance negatively.

2.6.3 Integrated Services

2.6.3.1 Introduction

The IntServ architecture provides a suite of standards to enhance traditional best-effort mechanisms (e.g., IP). The architecture is defined in RFC1633 and was developed by the IETF. The aim is to provide different QoS for multiple classes of traffic by adding new functionality to routers and adopting a method of reserving and requesting resources for applications prior to data being sent.

Reserving and requesting resources for applications allows for greater flexibility in stating requirements, anticipating demand and denying requests. However how resources are shared and how the available bandwidth is managed are important questions that arise when considering the Integrated Services approach to providing QoS.

The reservation of resources is achieved through RSVP. RSVP is defined in RFC2205. It creates a flow-specific resource reservation soft state in routers and hosts on a network. RSVP was designed to provide a robust, efficient, flexible and an extensible resource reservation service for both multicast and unicast data flows (Braden et al., 1995).

A prototype version of RSVP was developed by research collaboration during 1991-1993. This was further refined and documented as an Internet standard in 1995. To this day, a working group of the IETF continues to modify and develop the protocol to improve its performance.

2.6.3.2 Approach

In the IntServ architecture, each IP packet is associated with a flow. RFC1633 defines a 'flow' as a distinguishable stream of IP related packets that result from a single user activity
requiring the same QoS. It is the finest level of granularity of a packet stream that is
distinguishable by the IntServ architecture (Braden et al., 1994). A flow differs considerably
from a TCP connection in that it is both unidirectional (i.e., simplex) and allows for more
than one recipient of a flow (i.e., multicasting). It also differs from DiffServ that deals with
multiple or aggregated flows. It is interesting to note that the flow identifier in the IPv6
header is not equivalent to an IntServ flow but could be used to identify IntServ flows if the
need arises.

A network that supports the IntServ architecture is called an Integrated Service Packet
Network (ISPN). ISPN routers are not confined to work solely with routing algorithms and
packet discard policies (i.e., to minimise delay and control congestion respectively) as
traditional best-effort services do. It makes use of the following functions to manage
congestion and provide QoS transport.

- **Admission Control** – The admission control algorithm must be consistent with the
  IntServ model. Its task is to accept or deny whether a new flow can be granted the
  requested QoS without affecting earlier guarantees. The accept/deny decision is local at
each node. In addition, to ensuring QoS guarantees are met, admission control is also
  concerned with enforcing administrative policies.

- **Management Agent** – The management agent has access to, and can modify, the traffic
  control database. It is also allowed to direct the admission control module in admission
  control policies.

- **Reservation Setup** – RSVP passes the QoS request originating in the end system to each
  router along the data path (or to the branches of the delivery tree in the case of
  multicasting). The QoS request is composed of a flowspec (i.e., a list of parameters
  stating the desired QoS) and a filterspec that defines the subset of the data stream (i.e.,
  the flow that is to receive this QoS). Once it creates the flow-specific state in the
  endpoint hosts and routers it must also continue to maintain the flow-specific state for
  the remainder of the flow.
- *Routing Protocol* – The routing protocol maintains information as to the next hop for each destination address and each flow.

- *Queuing Discipline* – An effective queuing policy is implemented that takes into account the differing requirements of the different flows.

- *Discard Policy* - To manage congestion, the discard policy determines when and which packet will be discarded according to a packet’s QoS guarantee and service class.

These background functions support the main task of the router, which includes the forwarding of packets. This is done by the classifier and route selection strategy and by the packet scheduler.

The classifier and route selection strategy maps the incoming packets onto classes. The mapping is based on fields in the IP header that store the packets flow characteristics, its IP destination address, and the QoS it requires. A class is an abstraction that is local to a router (Braden et al., 1994). Particular routers can therefore classify the same packet differently. As an example, core routers may choose to map many flows into a few aggregated classes while edge routers may use a separate class for each flow where there is less aggregation. The result is the next hop address for this packet.

The packet scheduler manages the forwarding of different packets using a set of queues for each output port. It determines the order in which packets are queued and transmitted, as well as the selection of which packets are to be discarded. These decisions are based on the router’s discard policy and the state of the admission control module.

The presence of applications at hosts or end systems implies that data originates and terminates at these hosts and end systems rather than being forwarded. Applications therefore require some form of invoking of a local reservation setup agent, possibly through an Applications Programmers Interface (API). Other than invoking such an API, end systems use similar functions to that of a typical router (see Fig 2.9).
2.6.3.3 Service Models

The IntServ architecture attempts to differentiate between a number of applications that frequently use networks to send and receive data. It is these categories of applications (i.e., inelastic and elastic applications) that reflect the classes of traffic that Integrated Services attempts to service in their model.

The core service model is concerned almost exclusively with the time of delivery of packets (Braden et al., 1994). Inelastic applications need the data in each packet by a certain time while elastic applications are able to use the data whenever it happens to arrive. Per-packet delay is therefore the central quantity on which the network makes QoS commitments.

The categories of applications and their delay requirements are summarised by (Arindam, 2000) and are summarised below.

- **Elastic** – Elastic traffic places no specific demand on delay or bandwidth (e.g., email, web browsing). It is up to TCP to do all the work and ensure packets reach their destination. Usually, elastic traffic is handled in a best effort routing policy.

- **Real Time Tolerant (RTT)** – RTT accepts occasional packet losses but demands a maximum delay over the network (e.g., video applications that use buffering).
• **Real Time Intolerant (RTI)** – RTI places strong demands on network in terms of minimal latency and jitter since unexpected delay is unacceptable (e.g., video conferencing).

### 2.6.3.4 Classes of Service

Services for a flow of packets are defined on two levels. The first level consists of three general categories of service classes that provide a general type of QoS guarantee. They include:

- guaranteed service,
- controlled load service, and
- best-effort service.

Within each of these categories, the required QoS for a particular flow can be specified by an application via a set of variables or parameters. These variables or parameters form a Traffic Specification (TSpec) and/or a Service Request Specification (RSpec). The TSpec describes the flow’s traffic characteristics to the network while the RSpec is used to request the QoS required from the network.

The TSpec acts as a contract between the data flow and the service as long as the data traffic continues to be described accurately by the TSpec (Stallings, 2001). Packets that do not form part of the reserved flow will be given a best-effort service by default.

#### 2.6.3.4.1 Guaranteed Service

The Guaranteed Service class is the most demanding service provided by the IntServ architecture. The service imposes a strict maximum upper bound on packet delay, ensures that no packet is lost and that sufficient bandwidth is available for the duration of the flow (Stallings, 2001).

Guaranteed Service is described using both a TSpec and an RSpec but does not control the minimum or average delay as it suggests. It guarantees the requested delivery time by
controlling the maximal queuing delay. It is important to note that due to the constraints imposed on packet delays, delays tend to be set to large values to handle any rare cases of long queuing. In turn, the application if necessary may decrease the delay by increasing its demands for bandwidth. Having this upper bound on delay makes it a useful service for RTI applications (Arindam, 2000).

NOTE: Even though a Guaranteed Service QoS contract ensures that no packets are lost due to buffer overflows at routers, packets may still be lost due to failures in the network or changes in the routing paths (Stallings, 2001).

2.6.3.4.2 Controlled Load

Controlled Load service approximates the end-to-end behaviour seen by applications receiving a best-effort service subject to unloaded conditions (Stallings, 2001). The average delay is guaranteed and ensures that only a small number of packets will be lost due to buffer overflows. The controlled load is described using a TSpec only. The RSpec is not required, as the network does not give any quantitative guarantees.

The controlled service is useful for RTT applications. No upper bound on queuing delay needs to be specified since the receiver can measure the jitter experienced by incoming packets and then set the playback point to the minimum maximum delay that still produces an acceptable low loss rate. As an example, voice traffic can be adaptive by adjusting silent periods. Video traffic can be adaptive by dropping an occasional frame or delaying the output stream slightly (Stallings, 2001).

2.6.3.4.3 Best Effort

The Best Effort service class is analogous to today’s Internet. There are no guarantees by the network whatsoever and, therefore this class, has no need for a TSpec or RSpec. Elastic applications such as TELNET and FTP can use this class so this service will continue to be supported. It is interesting to note that the delay requirements of these elastic applications vary between interactive burst applications (e.g., TELNET), interactive bulk transfers (e.g., FTP), and asynchronous bulk transfer (e.g., e-mail) (Braden et al., 1994).
2.6.3.5 Resource Sharing

Admission control is necessary to ensure that real-time service commitments are met. Similarly, admission control is also necessary to ensure that resource sharing (or, link sharing, as it is most commonly known) is allocated fairly or according to a certain criteria.

The allocation of resources is negotiated on a flow-by-flow basis since each flow requests admission to the network separately (Braden et al., 1994). Resource sharing in the context of link sharing refers to the sharing of the aggregate bandwidth on individual links. Numerous methods of link sharing exist. They include:

- **Multi-Entity Link Sharing** – This occurs when a link is owned by several organisations who may wish to ensure that the link is shared in a controlled way, perhaps in proportion to the capital investment of each entity (Braden et al., 1994).

- **Multi-Protocol Link Sharing** – This method prevents particular protocols from overloading a link by excluding some protocols that may be more aggressive in responding to or detecting congestion.

- **Multi-Service Link Sharing** – Under this strategy, administrators may limit a fraction of the bandwidth to various service classes (e.g., restrict some real-time traffic to avoid pre-empting of elastic traffic).

2.6.3.6 Scheduling

An important component of the IntServ architecture is the queuing discipline or schedule used at the routers. Routers have traditionally used a First-In-First-Out (FIFO) or First-Come-First-Served (FCFS) queuing discipline. Only one queue exists and packets are transmitted in the order in which they arrive. This method of congestion management offers no special treatment for flows of higher priority or flows that are delay sensitive. Also, if smaller packets are queued behind larger packets then a larger average delay per packet is experienced causing flows of larger sized packets to get relatively better service (Stallings, 2001).
In Weighted Fair Queuing (WFQ) one queue is maintained for each flow (or service class). As in Round Robin Scheduling (RR), a WFQ scheduler will serve each queue in a circular manner. WFQ is a work conserving queuing discipline and thus will immediately move on to the next queue in the service sequence upon finding an empty queue (Kurose et al., 2001).

Contrary to RR where each queue receives the same amount of service, WFQ allows each queue to receive a differential amount of service in any given interval of time. Specifically, each queue $i$, is assigned a weight, $w_i$. The queue $i$ will then be guaranteed a fraction of the service equal to $w_i / \sum w_j$ where the denominator is the sum of the weights of all the queues. The weights assigned to each queue reflect the amount of QoS required by each traffic flow (or class).

WFQ ensures that low volume high priority traffic gets the service levels it expects through load balancing. In practice, after accounting for high priority traffic, the remaining service time is divided fairly among the lower priority traffic queues (Arindam, 2000).

WFQ in effect sets up packet classification and scheduling required for the reserved flows. An important advantage though is that it can adapt to network parameter changes. During congestion periods ordinary data packets are simply dropped (Arindam, 2000) (see Fig 2.10).

![FIFO and WFQ Comparison](image)

*Fig 2.10 FIFO and WFQ Comparison*

### 2.6.3.7 Policing
Policing attempts to regulate the rate at which a flow is allowed to inject packets into a network. Kurose et al. (2001) identifies three important policing criteria that differ according to the time scale over which the packet flow is policed.

- **Average rate** – limits the long term-long average rate (packets per time interval).
- **Peak rate** – limits the maximum number of packets that can be sent over a specified short period of time.
- **Burst size** – limits the maximum number of packets allowed to be injected over an extremely small interval of time.

The token bucket provides an abstraction mechanism that can be used to characterise these policing limits. The bucket can hold a maximum number of tokens $b$. Tokens are added to the bucket at a rate $r$ per second. (NOTE: Tokens are only added if the number of tokens in each bucket is less than $b$, otherwise tokens are discarded.)

Before a packet is transmitted, it must first remove a token from the bucket. If the token bucket is empty, the packet must wait for a token or, alternatively, be dropped. Since there are at most $b$ tokens in the bucket, the maximum burst size is $b$. Given, that the token generation rate is $r$, the maximum number of packets that can enter the network at any interval of time of length $t$, is $rt + b$. Thus, $r$ limits the long term average rate.

A token bucket and WFQ scheduler working in tandem provide a maximal delay in a queue. This provides the QoS expected by the Guaranteed Service Class or Controlled Load Service Class in the IntServ architecture.

**2.6.3.8 Resource reSerVation Protocol**

RSVP is a transport layer signalling protocol that synchronises sender and receiver on a flow-by-flow basis (Braden et al., 1997). It is possible to distinguish between sender and receiver since resources are requested solely for unidirectional data (i.e., simplex flows). While this is sufficient for streaming applications, the complexity increases for conversational applications where resources are required in both directions (i.e., both end points would have to send PATH and RESV messages).
Sessions, a concept relating to data flow, forms the basis of RSVP operation. A session is identified by the destination IP address (i.e., unicast or multicast address) of the intended flow.

Initially, the host that intends to send data, issues a special data packet called a PATH message to the receiver (i.e., the packet is sent downstream). This packet contains the traffic characteristics of the intended flow (i.e., TSpec) and the IP address of the previous node/router.

As the message is routed through the network, the upstream node/routers address is stored in a path-state table. In effect, the routers and intermediate forwarding devices form a path from source to destination so that the routers know the links on which they should forward RESV messages. If a path cannot be set up, then a PATH error message is sent back upstream to the sender.

It is important to note that RSVP is not a routing protocol but simply uses the path established by the underlying routing protocols. Any information carried along by RSVP, such as the TSpec or RSpec, is opaque to RSVP.

Once the receiver gets the PATH message, it issues a RESV message. This RESV message returns upstream (i.e., back the path taken by the PATH message) forcing each node along the path to decide independently how much of the demand it should satisfy or refuse altogether. In the event of multicasting, nodes only send one RESV message upstream even though they receive one from each downstream link. If a node refuses the reservation request, a RESV error message is issued downstream so that the receiver can drop the reservation request and try again later.

A RESV message contains the actual QoS characteristics expected by the receiver (i.e., the RSpec). Due to this receiver-initiated reservation, it is possible for different receivers to specify different QoS features for the same multicast flow. The receiver must also maintain the resource reservation for that session as well as initiate it.
The admission control (i.e., the decision as to whether the reservation is to be accepted or not) is not defined by RSVP. This is considered under a separate module within the router. On acceptance of a reservation request, the router must adjust its packet scheduler to accommodate the reservation.

The receiver can also request a number of different levels of reservation. These reservation styles determine or specify whether reservation merging is permissible. They are detailed in (Braden et al, 1997) and are summarised below.

- **Wildcard Filter** – This is where the receiver requests resources to be reserved for all sources in the session. Sources may come and go but they still share the same bandwidth resources to send their traffic. In this way, the receiver can receive from all of them.

- **Shared Explicit** – This is similar to the wildcard filter type except that the receiver chooses a fixed set of senders out of all those within a session to share the bandwidth.

- **Fixed Filter** – This specifies a list of senders from a session, along with the required bandwidth for each data flow the destination wants to receive.

Shared reservations, created by the wildcard and shared explicit filters are appropriate for multicast sessions whose sources are unlikely to transmit simultaneously (e.g., packetised audio). Fixed filter reservations create distinct reservations for flows from different senders. This makes them ideal for video teleconferencing.

All routers or nodes along the reserved path maintain reservations as 'soft state' (i.e., the information about the required resources requested by the flow in the PATH and RESV messages). This state is cached information and is held temporarily until:

- the sender or receiver cancels the request (i.e., by issuing a PATH or RESV Tear message),
- the receiver confirms the request with an RESV message, or
- neither happens and the soft state times out.
RSVP is advantageous for networks that run on a connectionless service where changes in routing occur frequently (e.g., the Internet). The routing protocols used to transmit packets from source to destination employ dynamic strategies to deal with link or node failures. Should there be a router failure on an established path, the remaining routers would continue to reserve that path for the required service level while waiting for the time out period. The communication, however, will only continue at the best-effort service level (Dobson, 2002).

Having routers hold information in a soft state forces the end systems (i.e., the sender and receiver) to periodically interchange PATH and RESV messages to maintain the state for the duration of the flow. This imposes processing demands on routers. It is also this large amount of signalling state information in the core of the network that leads researchers to believe that RSVP is not scalable (Arindam, 2000).

Requesting QoS with RSVP takes at least one round trip time between sender and receiver (i.e., it is necessary for the sender to send a PATH message before the receiver can respond with a RESV message). Since RSVP does not allow forward reservations, this results in a rather slow reservation method. Furthermore, RSVP messages may get lost as they are transported over IP under protocol number 46. This adds even more packets and delay to an already congested network.

The rejection of large reservation requests that are composed of smaller reservations downstream is another problem that faces RSVP developers. This adds further complexity to the RSVP protocol and its implementation.
Chapter 3

Methodology

3.1 Introduction

This chapter discusses three different approaches commonly used by the research community for the performance comparison of computer systems. They include

- empirical analysis,
- analytical modelling and
- simulation.

Details of the theory behind analytical modelling and simulation (i.e., the two methods used in this project to compare the performance of the IP and RSVP protocols) are presented. An overview of Markov models, current available simulators, and further details about how such tools are used is mentioned towards the end of the chapter.

3.1.1 Empirical Analysis

Empirical analysis involves running applications or benchmarks on existing systems and taking measurements using hardware or software monitoring. This is impractical for this project due to the cost, expertise, and amount of time required to set up a physical network. More importantly, empirical measurements only provide results for a specific set of parameters and system configurations rather than predict the performance under a widely varying workload and/or configurations.

3.1.2 Analytical Modelling

With analytical modelling, the behaviour of a system is captured through a set of equations, and the performance derived mathematically. This involves the construction of a number of functions that approximate the attributes of the system components in terms of workload
characteristics. Various assumptions are required to keep the model tractable and to allow for a wide variety of configurations to be modelled. It is a cheaper solution than either the empirical or simulation methods, but the accuracy of the results may be approximate. That is, the validity of these models is dependent on the quality of the assumptions. Its accuracy stems from the extent to which the mathematical models can capture real world behaviour.

3.1.3 Simulation

Simulation incorporates the idea of modelling a system and acting out its dynamic behaviour. A set of programs accomplishes this task by capturing and mimicking the characteristics of the system under test. It can therefore be used to predict the performance of a system and provides an accurate and cost-effective method of observing the effects caused by changing key parameters or properties in systems. However, designing, implementing, and validating a good simulation is time consuming and error prone due to software complexities.

Simulation plays a major role in the testing and evaluation of existing and future technologies. Simulation is used extensively by researchers in all areas of performance evaluation and testing. Therefore, there is substantial support and information readily available on this method of modelling.

3.1.4 Chosen Methodologies

For reasons mentioned above, empirical analysis is not a viable method to undertake for this project. Even though analytical models and simulation are useful and powerful tools, that are constantly being used by researchers in performance testing and evaluation, analytical models and simulation only approximate real world behaviour. Absolute accuracy is difficult to achieve. However, the results obtained should be satisfactory in providing a solution as to whether RSVP does indeed offer enhanced performance over plain IP.

3.2 Queuing Models
The simple queuing model shown in Figure 3.1 denotes customers arriving from time to time. On arrival, these customers join a queue until they are served and can therefore leave the system. The term ‘customer’ refers to any type of entity that is waiting for a requested ‘service’ from a particular system. This abstraction allows for production systems, repair and maintenance facilities, and communication and computer systems to be viewed as queuing systems.

Queuing models, whether solved mathematically or analysed through simulation, provide analysts with a powerful tool for designing and evaluating the performance of queuing systems. Typical measurements of system performance include:

- **Queue length** – Length in terms of the number of customers waiting at a queue or in service.

- **Response Time** – Time taken for customers to receive (i.e., the time waiting in the queue and obtaining service).

- **Server utilisation** – Percentage of time server is busy.

- **Throughput** – The average number of customers served during as a function of time

Queuing theory and simulation analysis predict these measures of system performance as a function of input parameters (Banks et al., 1996). Input parameters include the arrival rate of customers, their service demands, the rate at which servers work, and the number and arrangement of servers.
For relatively simple systems, these performance measures can be computed mathematically but for the more realistic models of complex systems, simulation is often required. Simple models, although tied to simplifying assumptions for tractability, are useful for developing an understanding of the dynamic behaviour of queuing systems.

### 3.3 Markov Models

Quantitative analytical performance techniques are based on Markov models. They are based on state space diagrams that provide an intuitive yet powerful descriptive tool. They are also used for predictive purposes once the model has been constructed and parameterised as they capture the key relationships between the architecture and the workload components are capture in mathematical expressions (Dowdy, 2002).

Markov models provide flexible models that are inexpensive to produce and have the capability to explain high level interactions between the system components. They also form the basis of other analytical techniques such as queuing systems. Two models have been developed in this project using Markov model principles. They serve to demonstrate and compare the performance of an IP-enabled router and an RSVP-enabled router.

Once the system is defined, a state diagram can be developed. Transitions (i.e., the state connections) indicate which states are accessible from/to other states. The weight or label assigned to each of the state transitions indicates the rate or probability of moving from one state to another. The weights are drawn from assumptions made of the original system. The model is then solved through the abstraction of a set of linear ‘balance’ equations and finding the long-term ‘steady state’ probabilities of each system state (Dowdy, 2002). After solving the model (i.e., solving the balance equations), performance predictions (e.g., queue lengths, response times, server utilisations, throughputs) can then be drawn from these probabilities.

NOTE: Markov models have a special property that arises from the fact that a negative exponential is used to model service and arrival times. This distribution has a so-called ‘memoryless’ property that allows time to be factored out of the analysis. Markov processes are memoryless because for any state the system can enter, the next state depends solely on
the current state of the system (Dowdy, 2002). States visited previously to the current state have no bearing or influence on the next transition.

3.4 Simulation Tools

Academia and industry offers a wide range of simulators. Each simulator is designed and used to evaluate or test particular areas of interest. In order to achieve an accurate set of results, the need for a good network simulator is required. Below is a short discussion on three state-of-the-art network simulators available.

3.4.1 ns-2

Ns-2 is a discrete event simulator targeted at networking research that originally began as a variant to the REAL network simulator. It is available free of charge to the research community. Its development was supported by the Defence Advanced Research Projects Agency (DARPA) during the Virtual InterNetwork Testbed (VINT) project in 1995. It is continuously being developed with the help of many researchers. ns-2 provides substantial support for simulation of transport, routing, and multicast protocols over wired and wireless (local and satellite) networks (ns-2).

Ns-2 allows the creation of network topologies via a scripting language called the Object-oriented Tool Command Language (OTcl). OTcl is an extension of the Tool Command Language (Tcl). A network animator (nam) is then used to visualise the output generated by the ns-2 simulator.

Ns-2 is supported across numerous platforms including Unix, Windows, and Sun operating systems. Manuals, tutorials and example script files are readily available from the ns-2 website (http://www.isi.edu/nsnam/ns/tutorial/index.html). An extended module created by Marc Greis provides additional support for RSVP.

3.4.2 OPNET Modeler
OPNET Modeler is the industry's leading network technology development environment. It allows designers or researchers to study communication networks, protocols, and applications. It was originally developed at the Massachusetts Institute of Technology (MIT) and introduced in 1987 as the first commercial network simulator (OPNET Modeler). It provides a comprehensive set of tools to display simulation results. The primary disadvantage of using OPNET Modeler for this project is that a license is required to use the simulator, which is rather expensive. The functionality of OPNET Modeler is not significantly different from other simulators mentioned here.

3.4.3 YATS

Yet Another Tiny Simulator (YATS) is a cell-level simulation tool mainly for ATM networks. It was developed in 1995/1996 (YATS) and is freely accessible to the research community. An input file, using a simple script language, describes the network configuration. The system is written in C++ and provides graphical object classes that are able to display the behaviour of performance variables.

Manuals, tutorials, and example files are readily available from the website (http://www.ifn.et.tu-dresden.de/TK/yats/yats.html). A disadvantage of YATS is that it is no longer supported. Although it provides support for TCP/IP entities, it is mainly applicable towards simulation of ATM networks and as such, does not support RSVP, a serious disadvantage for this project.

3.4.4 Chosen Simulator

Due to its availability, and support for RSVP, ns-2 is the obvious choice for this project. Other factors such as an extensive list of manuals, tutorials and example script files, its widespread use at several universities and local expertise at the University of Leeds adds further support for its use in this project. Ns-2 is widely used for complex simulations other than those presented in this report. Thus, it is quite capable of consistently demonstrating the performance of IP and ISPN networks relative to each other.

3.4.4.1 RSVP/ns
RSVP/ns is an implementation of RSVP for ns-2. It is also part of the VINT project. Marc Greis originally developed RSVP/ns for ns-2.1b3 version. Most recently, Sean Murphy enhanced and updated the code making it compatible with ns2.1b8, the version of ns-2 used in this project.

RSVP/ns is still under development and therefore does not reflect the full specification of RSVP detailed in the relevant RFCs. Some of the methods implemented in RSVP/ns differ to those specified in the RFCs as a result of limitations imposed by the simulation environment and memory and CPU limits. RSVP/ns currently includes the following features:

- controlled load service bandwidth guarantees based on WFQ,
- soft state with freely adjustable refresh intervals,
- fixed filter reservation style,
- gathering of statistics in the links and nodes,
- possibility of reserving bandwidth for RSVP PATH and RESV messages and
- all RSVP message types except the PATHERR message.

As mentioned, various small differences exist between objects in RSVP/ns and RSVP. The most important is the omission of the destination port and the protocol id field in the session object. In RSVP/ns, sessions are defined solely by their flow id.

3.4.4.2 The Network Animator - nam

Nam is a Tcl/TK based animation tool for viewing network simulation traces created by ns-2. nam began at Lawrence Berkeley Laboratory (LBL) in collaboration with the VINT project. Currently, it is being developed at the Information Sciences Institute (ISI) as part of the Simulation Augmented by Measurement and Analysis for Networks (SAMAN) and by the Collaborative Simulation for Education and Research (CONSER) projects.

To use nam with ns-2, the simulation results must be saved in a trace file whose format can be recognised by nam. It supports topology layout, packet level animation, and various data
inspection tools that allow a wide range of features in the simulation to be viewed graphically.

3.4.4.3 xgraph

Xgraph is a freely available utility program that is commonly used with ns-2. It is not part of ns-2 but it is a standard graphical program that takes post processed information calculated from ns-2 trace (log) files as input and displays the information in a series of graphs. This software is well suited for this project since graphs are used extensively to compare the performance of the RSVP and IP protocols because the results of performance metrics are difficult to see directly from the nam trace animation.

3.4.5 Using ns-2

Ns-2 along with RSVP/ns, was installed on a Linux system in the School of Computing. Ns-2 is essentially a Tcl interpreter with access to a large library of network related objects. In other words, to use ns-2, the user constructs a program using the OTcl script language. Therefore, to set up and run a simulation network, users write an OTcl script that initiates an event scheduler, sets up the network topology, and specifies to the event scheduler when to start and stop transmitting packets.

The objects themselves are actually implemented in C++. Ns-2 simply provides a front-end to create and manipulate these objects. Extra functionality can therefore be added to ns-2 by modifying the underlying C++ code, enabling users to create their own objects or modify existing ones.

3.4.5.1 Nodes and Agents

The most simple network element in ns-2 is the node. This is analogous to a router in real world networks. Nodes have routing capabilities and are able to deal with traffic flowing through them. Nodes cannot generate or receive traffic. Special objects required for this purpose are needed.
Traffic is generated through specialised application agents. They fall under two broad categories:

- sources that simply produce packets of specified sizes at a specified rate (e.g., according to an exponential distribution, a pareto distribution, or as a Constant Bit Rate (CBR)), or
- sources that imitate real applications such as FTP or TELNET.

Specified traffic requires both a source and sink to be defined. Transport agents that are analogous to the Internets' transport protocols (i.e., TCP and UDP) are used in ns-2 to simulate this environment. Each transport agent has its own corresponding sink agent. Fig 3.2 shows the hierarchy of objects and agents possible in ns-2.

One important design goal of RSVP/ns is to keep it as independent as possible from the rest of ns-2 to allow for compatibility with later versions of ns-2. As a result, a new agent, an RSVPAgent, can be attached explicitly to each node to maintain path and reservation states for all RSVP nodes, generate RSVP messages, and process incoming RSVP messages.

![Hierarchy of Objects in ns-2](image)

**Fig 3.2 Hierarchy of Objects in ns-2**

### 3.4.5.2 Measuring the network’s performance

Calculating performance measurements in ns-2 can be done using specialised agents known as LossMonitors. LossMonitors are attached to existing network nodes and allow access to variables that record useful information such as packet drops and packet receipts. LossMonitors however have limited functionality and they can only be used with UDP traffic.
Alternatively, calculations can be performed on trace or log files generated by ns-2. Trace files are essentially text files written at the end of the simulation where each line corresponds to a particular event or action performed by an individual packet. This file is typically post-processed to obtain results for any desired performance metric. This trace file can be easily processed using text-processing languages such as AWK or Perl.

Post processing the trace files was chosen to measure the networks performance in this project. This was done to maintain consistency when calculating the performance of UDP and TCP transport agents. The overwhelming support, discussions and tutorials available on the Internet and the ns-2 newsgroup made generating the scripts a relatively simple task.

3.4.6 AWK Scripts

There are three main categories of scripts required to interpret the trace files and generate files suitable for input to xgraph. The categories reflect the three performance metrics (i.e., delay, packet loss, and throughput) that are used to compare the IP and RSVP protocols. The performance categories are discussed below.

3.4.6.1 Delay

Calculating the average delay of packets per flow (or all flows collectively) requires the use of two scripts. The first script, delay.awk, measures and records the end-to-end delay of each packet traversing the network from source to sink. The output of delay.awk then serves as the input to the second script, avdelay.awk, that averages the times for each second of the simulation and provides input to xgraph. Delay.awk is adapted from a script written by Lloyd Wood while the general structure and avdelay.awk are adapted from a Perl script written by Stephen Dobson (Dobson, 2002).

3.4.6.2 Packet Loss

A single script, loss.awk, is used to calculate the number of packets lost. The script can be used to calculate the number of packets lost per flow or over all flows collectively. The
number of packets dropped is simply added on to the number of dropped packets previously recorded resulting in the total packets lost during the simulation. The script’s overall structure closely resembles a script written by Stephen Dobson (Dobson, 2002).

### 3.4.6.3 Throughput

As for delay, two scripts are used to calculate the throughput per flow or over all flows collectively. Throughput.awk calculates the number of bytes received at the traffic sinks at a particular instance in time. This then serves as input to totalThroughput.awk that calculates the total number of bits received at the traffic sink during each second. The output of totalThroughput.awk is the input for xgraph. Both scripts are adapted from work done by Stephen Dobson (Dobson, 2002).

### 3.5 Summary

This chapter presents and discusses the methodologies that are used throughout this project. Further information is also provided regarding what simulators are currently available to the research community. Details of how ns-2 in particular is used for this project is described. The scripts required to perform post-processing of trace files in ns-2 to measure the networks performance are also highlighted. A general introduction to queuing systems and Markov models is introduced to provide some background into the subject.
Chapter 4

Analytical Models

4.1 Introduction

This chapter introduces the complexity seen at routers and the reason why packets are forced to queue or wait in line for service. Details are given on the functionality offered by traditional IP routers and their ISPN counterpart. From this functionality and simplifying assumptions (i.e., assumptions introduced to keep the models tractable), abstract system models for each router are developed. The construction of Markov models for these routers, the notation used, and a summary of the parameters chosen for this project are presented and discussed.

4.2 Routers

Routers form an integral packet of today’s packet switched network (i.e., the Internet). Their task is to route packets through the network from sender to receiver. A simple abstract model is given in Fig 4.1. Since no connection path is established, congestion is likely at routers, due to nodal processing delays, queuing delays, and transmission delays as packets arrive and wait to be served (i.e., routed to their destination). Buffers or queues are required to handle the large number of packets arriving. However, these may also overflow, and packets may be lost due to high congestion, limited buffer space, or refusal for service based on QoS constraints. Routers can therefore be represented as queuing systems.

![Fig 4.1 Simple Abstract View of a Router](image)
4.3 IP Routers

IP routers (i.e., those that form the main backbone of the Internet) do not distinguish between packets. All packets are treated the same. Thus, packets form a single class of customers and routers do not consider a packet’s priority or service requirements. As packets enter the router, they are simply placed in a queue if other packets are also awaiting service. The queue employs a FIFO policy whereby packets are serviced in the order that they arrive at the queue. If the queue or buffer overflows packets are discarded. In the event that no other packet is waiting, or the packets is at the head of the queue, then serviced and routed accordingly.

![Router Diagram](image)

*Fig. 4.2 Abstract View of an IP Router*

4.3.1 Assumptions

Routers make no distinction between packets. Thus, a single class of traffic is used to represent the incoming packets.

Buffers or queues within the router are finite. Otherwise, memory requirements would be too large to accommodate a large number of customers (i.e., packets). This places a limit on the number of customers that are in the system at any particular moment in time. Systems that specify the total number of customers that can be in the system at any moment in time are known as closed systems.

Any customer arriving when the system is full is simply dropped. This can be considered as a form of congestion flow control. The number of customers or packets in a closed system is referred to as the Multi-Programming Level (MPL).
Finite models such as the one developed in this project characterise the arrival process in a completely different fashion to that of infinite (or open systems) models. Customers are pending (or delayed) when outside the queuing system at a particular time $1/\alpha$ (An average delay of $1/\alpha$ time units implies that the rate at which customers arrive from the outside is $\alpha$). The delay server shown in Fig 4.3 represents this delay rate.

Once customers are inside the queuing system they wait in line if necessary and are then serviced at a rate $\mu$ per second. As all packets are treated equally in an IP router, the service rate remains the same for all customers.

![Fig 4.3 IP Router System](image)

### 4.3.2 Markov State Diagram Construction

Construction of a Markov state diagram requires the enumeration of all the possible states in which the system might find itself. This is the system’s state space. In this simplistic model, only three states exist. The possible states are:

- two packets are at the delay server,
- one packet at the delay server and the other is at the queue awaiting service, and
- both packets are in the queue waiting for service.

This state space is mutually exclusive and collectively exhaustive. It is therefore not possible to be in more than one state at a time (i.e., mutually exclusive) and it is not possible to be in any state other than those identified (i.e., collectively exhaustive) (Dowdy, 2002).
Adopting the notation \((X, Y)\), where \(X\) denotes the number of customers at the delay server and \(Y\) denotes the number of packets at the queue waiting for service, the states are \((2,0)\), \((1,1)\), and \((0,2)\), respectively.

The state transitions represent the possible states that the system may find itself in the next time step. (NOTE: Self-loops represent the system remaining in a particular state for a period of time. Such self-loops are implied and not explicitly shown in the models.) In this model the possible state transitions are as follows:

- If in state \((2,0)\) then no packets are currently at the router. Packets can arrive from the delay server and enter the queue (i.e., state \((1,1)\)) with rate \(2\alpha\).

- If in state \((1,1)\) two possible transitions exist:
  1. either the packet waiting at the queue is routed (i.e., leaves the queue and goes back to the delay server, state \((2,0)\)) with rate \(\mu\), or
  2. a new packet joins the queue from outside at rate \(\alpha\) and waits for service (i.e., the packet left in the delay server leaves the delay server and joins the queue, \((0,2)\)).

- Finally, if in state \((0,2)\) then the first packet at the queue gets routed (i.e., goes back to the delay server, state \((1,1)\)) with rate \(\mu\).

The corresponding state diagram is shown in Fig 4.4.

![Fig 4.4 Simplistic Markov Model for the IP Router](image)

### 4.4 ISPN Routers

ISPN routers have additional functions to that of a traditional IP router in order to manage congestion and provide QoS transport. These include a classifier (i.e., maps incoming packets into classes based on their flow requirements) and a packet scheduler (i.e., manages...
one or more queues and determines the order packets are queued and transmitted). The number of queues corresponds to the number of flows (i.e., classes of traffic) currently passing through the router.

Each queue has a particular weight associated with it. This weight determines how much time each queue is allocated at the router for packets to be sent to the next hop (i.e., WFQ). The individual queues themselves employ a FIFO policy. If a particular queue or buffer is full, packets destined for that buffer are discarded. In the event that no packets are waiting at a particular buffer, then the packet is serviced and routed accordingly.

IntServ distinguishes between three different sets of applications that regularly make use of the Internet. These categories of applications reflect the classes of traffic that are serviced by the IntServ architecture. The three classes are:

- Guaranteed Service Class for real time intolerant applications,
- Controlled Load Service Class for real time tolerant applications, and
- Best Effort Service Class for elastic applications.

Fig 4.5 gives a simplified model of this simulation.

**Fig 4.5 Abstract view of an ISP Router**

### 4.4.1 Assumptions
As the classifier distinguishes between types of packets, it is sensible to have multiple classes of traffic to represent the incoming packets. Furthermore, the different classes of traffic can be set to three, since there are three distinct service classes. Table 4.1 summarises the different classes of traffic.

<table>
<thead>
<tr>
<th>Class Number</th>
<th>Traffic Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Guaranteed</td>
</tr>
<tr>
<td>2</td>
<td>Controlled Load</td>
</tr>
<tr>
<td>3</td>
<td>Best Effort</td>
</tr>
</tbody>
</table>

*Table 4.1 Traffic Class Numbers*

The number of different classes of traffic can also be translated to the number of queues that are included in the model (i.e., three). That is, class $i$ traffic is serviced by the router via queue $i$. Table 4.2 summarises the different queues and the classes of traffic they service.

<table>
<thead>
<tr>
<th>Queue Number</th>
<th>Traffic Class Serviced</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Guaranteed (1)</td>
</tr>
<tr>
<td>2</td>
<td>Controlled Load (2)</td>
</tr>
<tr>
<td>3</td>
<td>Best Effort (3)</td>
</tr>
</tbody>
</table>

*Table 4.2 Queues for Traffic Classes*

As ISPN routers employ a WFQ scheduler, each queue is assigned a weight $w_i$, where $w$ is the weight assigned to queue $i$. Queue 1 has a greater weight than queue 2 which in turn has a greater weight than queue 3. The weights themselves reflect the QoS demands imposed by the traffic classes on a network.

The fraction of time that the router dedicates to each queue is dependent on the class weights and on the number of packets present in the systems (i.e., the number of packets present in all three queues). The algorithm used to carry out this calculation is is given below.

For each queue $i$,

\[
\text{fraction of time for each queue} = \frac{n_i \cdot w_i}{\sum_{j=1}^{k} n_j \cdot w_j}
\]
where
\[ n_i \] is the number of packets in queue \( i \)
\[ w_i \] is the weight of queue \( i \)
\( k \) is the number of queues

Due to the varying classes of service and the demands they place on the network, some minimum thresholds are assumed for each queue.

\[
\begin{align*}
\text{if } (30\% \leq \text{queue 1’s fraction} \leq 85\%) \text{ and } (\text{queue 2’s fraction} < 15\%) \text{ then} \\
&\quad \text{Queue 1’s fraction remains the same} \\
&\quad \text{Queue 2’s fraction is assigned } 15\% \\
&\quad \text{Queue 3’s fraction is assigned the remaining percentage}
\end{align*}
\]

\[
\begin{align*}
\text{if } (\text{queue 1’s fraction} < 30\%) \text{ and } (15\% \leq \text{queue 2’s fraction} \leq 70\%) \text{ then} \\
&\quad \text{Queue 1’s fraction is assigned } 30\% \\
&\quad \text{Queue 2’s fraction remains the same} \\
&\quad \text{Queue 3’s fraction is assigned the remaining percentage}
\end{align*}
\]

\[
\begin{align*}
\text{if } (\text{queue 1’s fraction} > 85\%) \text{ then} \\
&\quad \text{Queue 1’s fraction remains the same} \\
&\quad \text{Queue 2’s fraction is assigned the remaining percentage} \\
&\quad \text{Queue 3’s fraction is assigned } 0\%
\end{align*}
\]

Basically, queue 1 (i.e., guaranteed traffic) is assumed to get at least 30% of the routers capacity and queue 2 (i.e., controlled load traffic) is assumed to get at least 15% of the routers capacity.

As an example, consider a snapshot of the system where there is 1 packet in queue 1, 2 packets in queue 2, and 3 packets in queue 3. If queue 1 has a weight 3, queue 2 has a weight 2, and queue 3 has a weight 1 then the fraction of time dedicated to each queue is:

- 3/12 (or 25%) for queue 1,
- 6/12 (or 50%) for queue 2 and,
- 3/12 (or 25%) for queue 3.

Queue 1’s fraction of time at the router can not be less than the minimum threshold assumed above (i.e., 30%). Subsequently, the fraction of time distributed so that 30% is given to queue 1, 50% is given to queue 2, and 20% is given to queue 3.
Buffers or queues within the router are finite as in the IP model. As a result, the system is closed and any customers that arrive and find that the system is full are dropped. Each traffic class will thus have its own maximum number of customers represented by $MPL_i$ where $i$ is the traffic class.

Being a finite model customers are pending (or delayed) outside the queuing system. These customers arrive to the queuing system at rate $\alpha$. Each traffic class has its own delay rate represented by $\alpha_i$ where $i$ is the traffic class.

Once inside the system customers will wait in line, (if necessary), and then be serviced at rate $\mu$. Each traffic class has its own service rate per second denoted by $\mu_i$ where $i$ is the traffic class. The overall model is represented in Fig 4.6.

![Fig 4.6 ISPN Router System](image)

**4.4.2 Markov State Diagram Construction**

Construction of the Markov state diagram requires the enumeration of all possible states in which the system might find itself. This is the system’s state space. In this system, eight states exist. The state space is both mutually exclusive and collectively exhaustive.

In this model the notation $(A, B, C, X, Y, Z)$ is adopted where

- A denotes the number of class 1 packets at the delay server,
- B denotes the number of class 2 packets at the delay server,
- C denotes the number of class 3 packets at the delay server,
- X denotes the number of packets at queue 1 waiting for service,
- Y denotes the number of packets at queue 2 waiting for service, and
- Z denotes the number of packets at queue 3 waiting for service.

The state transitions are single step and represent the possible states that the system may find itself next. If queue 1’s weight is 3, queue 2’s weight is 2, and queue 1’s weight is 1 then the state transitions possible are shown in Fig 4.7.

![Fig 4.7 Markov Model for the ISPN Router](image)

### 4.5 Parameters

To investigate and compare the performance of the IP and ISPN router it is important to keep as much consistency as possible between the two systems. Furthermore, it is vital that the parameters used to define the arrival and service rate of the models are identical. Table 4.3 summarises the parameters used in both systems to yield the results detailed in chapter 6 of this report.
<table>
<thead>
<tr>
<th>Model</th>
<th>Variable</th>
<th>Value</th>
<th>Description of variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP</td>
<td>1</td>
<td>delay rate of packets</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>service rate of packets</td>
<td></td>
</tr>
<tr>
<td>ISPN</td>
<td>1</td>
<td>delay rate of guaranteed service packets</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>delay rate of controlled load service packets</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>delay rate of best effort service packets</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>weight of guaranteed service queue</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>weight of controlled load service queue</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>weight of best effort service queue</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>service rate of guaranteed service packets</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>service rate of controlled load service packets</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>service rate of best effort service packets</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.3 Parameter Values for Analytical Models

The class weights (i.e., 1/2 for guaranteed, 1/3 for controlled load, and 1/6 for best effort) from the guaranteed service queue to the best effort service queue have been chosen to reflect the higher priority given to guaranteed service packets. This difference in weights reflects the fraction of time each queue is allowed at the router. A low service rate for all packet classes has been chosen to create a congestion situation. Congestion situations cause packets to queue at the buffers allowing for the queue length and response times of each queue to be monitored.

4.6 Performance Metrics

Performance metrics are important when carrying out a comparison of different protocols. They highlight the differences between protocols and show how effective the protocols are. The properties chosen to compare the performance of the IP and ISPN analytical models are:

- **Throughput** – The average number of packets served as a function of time,
- **Queue length** – The average number of packets weighting at a queue or in service,
- **Response time** – The average time taken by a packet to receive service, and
- **Power** – The ratio throughput/response time.

Monitoring the queue length and response time gives a good indication as to how fast the system deals with congestion situations. The throughput measures how effective the protocol is in using and allocating resources while power indicates the peak operating region of the network.
4.7 Summary

This chapter discusses some of the key complexities experienced at routers. Using abstraction and numerous simplifying assumptions, two models that mimic the main functionality of IP and ISPN routers have been developed. Finally, the parameters, including the arrival and service rate and the performance metrics to illustrate the difference in performance of the two models have also been discussed.
Chapter 5

Simulation Experiment Design

5.1 Introduction

State space explosion limits the size, complexity, and extent to which the analytical models are described earlier can be investigated. The lack of software tools capable of automating the process also restricts their use. The fact that real world computer systems are rarely simple indicates the need for further investigation of the performance of these protocols using simulation.

It is important to note that although simulation is a powerful and useful technique, it is only an approximation of real world behaviour. As such, simulation does not provide a complete and accurate insight into what is being investigated (Dobson, 2002). However, simulation does provide a good overall picture of the major differences between the two protocols considered in this project.

This chapter introduces some key concepts that are useful in designing a topology for simulation in general. It then defines a topology that highlights the advantages that RSVP brings to applications demanding QoS. A discussion of the parameters and traffic generators used to simulate these applications is also presented. The relevance of the various performance metrics used to compare the protocols are explained towards the end of the chapter.

5.2 Designing Topologies for Simulation

In order to compare and investigate the performance of plain IP networks and ISPN networks it is important to maintain as much consistency as possible. As a result, the underlying topology for each protocol must be identical. This includes network configuration parameters such as link delays, link bandwidths, the amount and type of traffic, and their source and sink patterns. Satisfying these constraints is not easy, yet it
ensures that the results obtained give a fair representation of the performance comparisons achieved by these protocols.

When designing topologies the benefits brought by IntServ architecture and RSVP to a network are considered. Since RSVP relies on IP routing (i.e., dynamic routing), traffic engineering is not possible, because explicit routing is not implemented. Subsequently, applications are allowed to request bandwidth reservations to guarantee some level of service. These factors have been taken into account when designing the topologies.

The physical layout of the topology also dictates what is possible and what is not in a network. There are numerous types of networks that are generally used within the research community when carrying out simulations. Typically, complete, partial and single path network topologies are considered. Complete network topologies are necessary when a particular problem is scaled across the network and its aim is to identify the impact of the problem on all devices involved. A complete topology is thus useful to monitor link utilisation, router behaviour, and application end-to-end delays across an entire network. Partial network topologies represent key portions of a network in detail, while abstracting other sections. This allows researchers to study the utilisation of backbones and the effect of QoS on the backbone. Single path topologies isolate and model the infrastructure between two devices of interest (e.g., a client and server). It enhances the focus of study to the main objects of interest, minimising errors and making the simulation fairly efficient.

The type of traffic that runs over the network topology must also considered and is an essential element in the success and confidence with which the simulation results are viewed. CBR over UDP and FTP over TCP are selected since they are implemented in ns-2 to demonstrate real applications. This also matches the three application classes that IntServ attempts to service in the previous chapter.

CBR provides a constant stream of data over a network, simulating either a video or audio session. FTP is one of the most common applications that runs over TCP on the Internet and is used to transfer files between nodes. UDP and TCP are the main transport protocols used on the Internet to carry traffic from source to sink. Examining the properties of UDP data shows how IP and ISPN networks react to constant flows of data while TCP is required to show how the effects of packet retransmission and the slow-start algorithm impact overall
performance. In current networks, the proportion of TCP traffic is typically higher than UDP traffic. However, as more real-time traffic is used in the future, the amount of UDP data will increase.

Due to the randomness introduced into the traffic flows (i.e., ns-2 uses random number generators to determine at what times packets are sent) each experiment is replicated five times and the results averaged. This provides a fair representation of the performance achieved by each protocol as well as minimising the chance of rogue values or errors. This is common practice in simulation forms the basis for providing confidence intervals.

5.3 Chosen Topology

A single path topology has been designed to illustrate the advantages offered by ISPN networks. There are two versions of the same topology shown in Fig 5.1, one of which is IP-enabled, and the other RSVP-enabled. The simulation is expected not to show too much difference in the overall performance between IP and ISPN networks. Some minor improvements, especially to individual flows are visible. The fact that traffic cannot take different routes limits the extent to which performance can be improved on since the routing protocols are identical in both cases.

Nodes 0, 1, and 2, on the left are the traffic sources. They send data to nodes 5, 6, and 7, respectively. Each link has a bandwidth of 10 Mbps and a delay of 1 ms except for the link
between nodes 3 and 4 that has a bandwidth of 1 Mbps and a delay of 10 ms. The design allows for a congestion situation to develop. In the IP-enabled experiments, each router uses a Droptail queue with a maximum queue size of 50 packets. The RSVP-enabled topology uses a WFQ scheduler that provides a queue for each traffic flow. NOTE: RSVP/ns does not allow the user to specify a maximum queue size for each queue in the WFQ scheduler. The default queue size is not known either.

Three traffic sources allow for the comparison of congestion control and how IP and ISP networks deal with varying volumes of data. The traffic sources are activated and deactivated at different times throughout the simulation. The topology has three different scenarios. The scenarios are outlined below and are identical for both the IP-enabled and RSVP-enabled experiments.

- Nodes 0,1, and 2 all exhibit CBR traffic over UDP.
- Nodes 0,1, and 2 all exhibit FTP traffic over TCP.
- Node 0 and node 1 exhibit CBR traffic over UDP and node 2 exhibits FTP traffic over TCP.

When CBR traffic is used, 210 byte packets are generated every 0.00375 seconds. This amounts to a sending rate of 448 Kbps. When TCP is used, packets are 1000 bytes in size except for ACK packets which are 40 bytes. The slow-start algorithm is also enabled when TCP traffic is used. These parameters are representative of the traffic found on the Internet today and are the default values supplied by ns-2.

Each experiment is run for 60 seconds, an adequate amount of time from which trace files are post-processed to measure the networks performance. Traffic from node 0 begins transmitting at 10 seconds, followed by node 1 at 13 seconds, and finally node 2 at 16 seconds. The initial 10 seconds is used for factoring out initial transient effect and is common practice in simulation modelling. This is known as the warm-up time and makes sure that there is a consistent environment in which to begin simulation (Dobson, 2002). In the RSVP-enabled topology, the 10 seconds are used to set up and stabilise PATH and RESV states.
The reservations are identical in all the RSVP-enabled experiments. They are specified using a token bucket policy. Node 5 reserves 0.5Mbps, node 6 reserves 0.3Mbps, and node 7 reserves 0.15Mbps. These reservations closely match the weightings used in the analytical modelling portion of this work. A small percentage of the link (i.e., 1000 bps) has also been reserved to avoid the loss of PATH and RESV messages as refresh intervals are every 5 seconds. This reduces the scalability problem mentioned earlier.

5.4 Performance Metrics

Performance metrics are important when carrying out a comparison of different protocols. They highlight the differences between protocols and show how effective the protocols are. A number of common properties are considered important in most networks regardless of the protocol used. The properties chosen to compare the performance of IP and RSVP are outlined below.

- **End-to-end delay** – the average time taken for a packet to travel from source to sink.
- **Throughput** – the amount of data (i.e., in bits) that arrives at the sink from a particular source.
- **Number of packets lost** – the number of packets that are sent but never received at the sink (or are delivered in an unusable, corrupted form).

Monitoring the number of packets lost gives a good indication of the reliability associated with the protocol. The packet delay shows how well a protocol deals with congestion situations. Throughput measures how effective a protocol is in using and allocating resources. Various other metrics such as jitter (i.e., variance in delay), round trip time and power could also have been considered but the above metrics highlight the primary enhancements that RSVP provides IP networks.

5.5 Summary

This chapter illustrates the topologies, experiments and configuration parameters used in the simulations. The results are presented and discussed in the following chapter.
Chapter 6

Results

6.1 Introduction

This chapter presents and discusses the results achieved in both the analytical models and the simulation experiments. Conclusions for each methodology and a summary of the overall findings are also given.

6.2 Analytical Models

For each performance metric, the IP and ISPN system as a whole are compared. This is followed by a set of graphs giving the results obtained for the individual traffic classes. A discussion of the results has also been included to explain the results.

6.2.1 Response Time

Overall response times (i.e., averaged over all traffic classes) for both models are identical as shown in Fig 6.1. This is expected since IP or ISPN model route packets at the same service rate. Response time increases at a constant rate as the workload intensity (i.e., the MPL) increases. However, the response times of the individual service classes (i.e.,...
The guaranteed traffic class provides applications with guarantees including minimal delay and packet loss. To maintain these service commitments a higher weight is assigned to the queue that deals with these types of packets. The higher weight ensures that these packets are allowed more time at the router, allowing packets to be forwarded to their next hop quicker. This reflects the lower response times depicted in Fig 6.2. The controlled load service exhibits similar results to IP, since the queue weight is in between the weights for the guaranteed and best-effort classes, and is closer to the ‘average’ IP traffic. The high response time experienced by best effort packets is due to the lower queue weight and the trade-off experienced in allowing guaranteed service packets faster response times. Intuitively, as the number of packets in the system increases the response times increase exponentially.

**6.2.2 Queue Length**

The queue lengths for both the IP and ISPN models seen in Fig 6.3 are identical and increase at a constant rate. As in the average response time, this is expected. Although the ISPN model has three queues, one for each traffic class, the sum of the number of packets in each queue is identical to that of the IP model. This is because the number of packets serviced remains the same, regardless of class. In the ISPN model, however, priority is given to guaranteed service packets rather than equal priority being given to all packets. This is depicted in Fig 6.4.
The difference in queue length experienced is expected. Queues with higher weights (i.e., guaranteed traffic) are allowed more time at the router and, in turn, will cause more packets from that queue to be routed. The guaranteed service queue experiences lower queue lengths at the expense of the best effort service queue in the ISPN model. The controlled load queue exhibits similar queue lengths to the IP model and depicts the increase in queue length as constant. This is because the average arrival rate is one packet per second per traffic class. All other classes have the same arrival rate and exhibit a constant increase in queue length throughout.

6.2.3 Throughput

Fig 6.5 illustrates that the overall throughput achieved by the IP model is the same as that of the ISPN model. This is as expected since it is physically impossible to push more packets
through the network than it can handle (i.e., total bits sent is equal to the link capacity). Having a relatively low service rate accounts for the increase in throughput as the offered load (i.e., the MPL) increases. It can also be seen that both models are 100% utilised once six packets (i.e., MPL = 2 in each of the three classes) are in the system as the workload is relatively high.

When looking at the throughput for each individual traffic class, as shown in Fig 6.6, it can be seen that RSVP in this case has given individual traffic classes a particular bandwidth reservation according to the queues weight. The guaranteed service as expected, has a higher throughput than the controlled service since the guaranteed service queue has the greater weight. The controlled load service class shows a similar throughput to that of IP. It is interesting to note that the IP packet class and the controlled load service class in IntServ reach their peak throughput relatively quickly. The increased throughput seen by guaranteed
service packets is achieved at the expense of the IntServ best-effort service class. This is understandable, as the weight assigned to the best effort queue is considerably lower than that of the guaranteed service queue. Furthermore, as the number of packets in the system increases, the best-effort class throughput continues to drop until its minimum threshold is met. Consequently, the guaranteed service class continues to experience improving throughput.

### 6.2.4 Power

![Fig 6.7 Power for the IP and ISPN Models](image)

Fig 6.7 shows that the IP and ISPN models have equal values of power. Considering that power is calculated from throughput and response time, this is understandable, as the IP and ISPN models have shown identical results for all metrics discussed previously. Peak power is achieved when the number of packets in the system is between two and three. This indicates the peak operating region of the network, where response times are still relatively low and high throughput is maintained. Consequently as the number of packets in the system increase, power continues to drop as throughput increases only minimally and response times increase dramatically.

The power exhibited by the individual traffic classes is displayed in Fig 6.8. The power of each traffic class follows a similar pattern to that of the IP and ISPN system as a whole. As expected, the power of the guaranteed traffic class exceeds those of the controlled load and best-effort traffic classes. The excess power awarded to the guaranteed traffic class comes because of the poor performance attained by the best-effort traffic class. The controlled load
traffic class exhibits similar properties to IP, as has been the case for the other performance metrics.

\[\text{Fig 6.8 Power per Traffic Class}\]

### 6.3 Simulation

For each scenario, the IP and ISPN topologies as a whole are compared. This is followed by a set of graphs giving the results obtained for each individual traffic flow in the IP or ISPN topology. A discussion of the results is included to explain the results.

#### 6.3.1 Three UDP sources

#### 6.3.1.1 Average end-to-end delay

Figure 6.9 shows that the end-to-end delay (i.e., averaged over all the traffic flows) of packets in the ISPN network is higher than the end-to-end delay exhibited by packets in the IP network. In fact, the delay is up to 1.5 seconds slower at times. This is because RSVP nodes have a queue allocated for each individual flow. Each queue is subsequently allowed a fraction of the routers capacity resulting in longer queuing delays. The constant arrival rate of packets adds to the congestion.
Fig 6.9 Average End-to-End Delay for the IP and ISPN Topology

Fig 6.10 shows the average end-to-end delay experienced by IP packets. IP routers treat all packets equally. This equal treatment of packets explains why the same end-to-end delay is recorded for each flow.

As the first source begins transmitting after 10 seconds, the delay is recorded at approximately 0.013 seconds. This is the minimum end-to-end delay a packet of this size can experience in this network. When the second source starts transmission after 13 seconds delay increases slightly to approximately 0.018 seconds. A small increase in delay is experienced because of the considerable amount of processing being performed at the
routers. Queuing is not necessary since both sources take occupy the entire bandwidth of the bottleneck link.

Once source 3 begins transmission on 16 seconds, the combined traffic rate is 1.5 Mbps. This rate exceeds the available bandwidth causing the buffers to overflow. This introduces large queuing delays and accounts for the increase in delay of packets.

The RSVP-enabled topology reserves bandwidth for each application (i.e., 500 Kbps for source 0, 300 Kbps for source 1, and 150 Kbps for source 2). The bandwidth reserved for source 0 equals the applications sending rate. Packets therefore traverse the network experiencing the minimum delay possible. This is illustrated in Fig 6.11. As bandwidth reservation drops, packets are required to queue at the routers. The large delays experienced by packets from flow 1 and flow 2 exhibit this pattern. As a result, RSVP improves the delay of individual flows at the expense of other less demanding traffic.

![Graph](Image)

*Fig 6.11 Average End-To-End Delay per ISPN Flow*

6.3.1.2 Throughput

Fig 6.12 shows that the overall throughput achieved by the IP and ISPN networks are identical. This is as expected since it is physically impossible to push more packets through the network than it can handle (i.e., total bits sent is equal to the link capacity). The manner
in which this bandwidth is allocated for each flow is shown in Fig 6.13 for the IP network and Fig 6.14 for the ISPN network.

**Fig 6.12 Throughput for the IP and ISPN Topology**

As sources begin transmitting, the bandwidth occupied matches the sending rate of each application. However, once all sources are transmitting packets (i.e., after 16 seconds of simulation time) in the IP network, each flow occupies the same amount of bandwidth. This is as expected since IP treats all packets equally. As sources stop transmitting packets, remaining flows take up the freed bandwidth allowing the buffer to empty and the throughput to be increased.

**Fig 6.13 Throughput per IP Flow**
In the ISPN network, the bandwidth reserved for each flow dictates the throughput achieved. This is as expected and remains so throughout the simulation. It is interesting to note the throughput in Fig 6.14 for flow 2 is recorded as 190 Kbps rather than the 150 Kbps reserved by RSVP. This is greater than the bandwidth reserved originally for flow 2 due to the greedy nature of the UDP data model. As in the IP model once sources stop transmitting packets, remaining flows take up the freed bandwidth easing congestion at the routers.

![X Graph](image)

*Fig 6.14 Throughput per ISPN Flow*

### 6.3.1.3 Packet Loss

Figure 6.15 shows that ISPN networks drop fewer packets than IP networks. The IP network starts dropping packets at approximately 17 seconds (i.e., just after the third and final source begins transmitting packets). This is as expected since the combined sending rate of the three flows exceeds the links capacity. The ISPN network records the first packet loss at 19 seconds. The reason for this difference in time before a packet drop is recorded is that packets in the ISPN network need to fill each queue in the WFQ scheduler while in the IP network only one buffer exists.
The number of packets lost for each individual traffic flow in IP networks is similar for all flows (see Fig 6.16). The total packets lost for each flow is identical as would be expected of a best-effort service protocol. The individual traffic flows in the RSVP-enabled topology yield much more interesting results (see Fig 6.17).

Flow 0 exhibits no packet loss as the bandwidth reserved by RSVP matches the sending rate of the application. As bandwidth reservations drop, packet loss is increased. Consequently,
flow 2 records losses of approximately 2500 packets during the course of the simulation while flow 3 drops approximately 5000 packets.

6.3.2 Three TCP sources

6.3.2.1 Average end-to-end delay

The average end-to-end delay of packets for three TCP sources shows a similar pattern of results to the average end-to-end delay of packets experienced with three UDP sources. TCP traffic over IP does not give a constant average delay like UDP traffic. It increases gradually and then produces the jagged edge exhibited in Fig 6.18. This is due to the slow-start algorithm of TCP. The variation in delay is called jitter.

The source sends one TCP packet and waits for an acknowledgement (ACK) before sending any further packets. When acknowledged the source transmits the next two packets of data. One ACK results in two further packets being sent. This causes an exponential increase in packets sent per round trip time. The source continues to increase the number of packets it sends until the maximum specified rate is met or an ACK is not received. When an ACK is misplaced because of congestion or packet loss, the TCP sender retransmits the packet, starting the process again.
In the ISPN network, (see Fig 6.19), the jagged edge is only seen as new sources begin transmitting packets (i.e., at 13 and 16 seconds respectively). Once the traffic flows occupy the reserved bandwidth, the delay remains constant throughout the simulation. RSVP reduces and in some cases, eliminates jitter (e.g., flow 0). Low jitter is a useful property transmitting audio and video packets.

The average end-to-end delay experienced by packets in the IP network is similar for all flows. This is as expected from a best-effort service such as IP. Flow 0 in the ISPN network
experiences a maximum delay of 0.3 seconds throughout the simulation. As the reserved bandwidth diminishes, the remaining two flows show higher delays. The overall effect is that the end-to-end delay experienced by all packets (i.e., averaged over all traffic flows) in the ISPN network is greater than the IP network (see Fig 6.20). This is because large queuing delays are experienced by flows queuing at low priority queues in the WFQ scheduler. The low jitter provided by ISPN networks is clearly exhibited in Fig 6.19 and Fig 6.20 even though the overall end-to-end delay (i.e., averaged across all flows) is greater than the delay shown for IP networks.

![Graph showing average delay and throughput comparison between IP and ISPN networks.]

**Fig 6.20 Average End-To-End Delay for IP and ISPN Networks**

### 6.3.2.2 Throughput

The results as expected, show a similar pattern to the throughput seen for UDP sources. In the IP network, it is interesting to note that the last flow to begin transmitting (i.e., flow 2 in Fig 6.21) is not treated equally as the other flows. Lower throughput is exhibited for flow 2 because a congestion situation has already developed and the slow-start algorithm restricts the number of packets being sent.
The ISP network throughput for each flow reflects the bandwidth reserved by RSVP as expected. The throughput for each flow remains constant throughout the simulation. As flows terminate, the throughput for the remaining flows increases since packets at the buffers are processed (see Fig 6.22).

6.3.2.3 Packet Loss
Due to the reliable data transfer model provided by TCP, packet loss in the IP and ISPN networks is minimal (see Fig 6.23). The ISPN network exhibits no packet loss providing an improvement on plain IP networks. Most packets lost in the IP network occur at approximately 16 seconds. Source 2 starts transmitting at 16 seconds creating congestion in the network resulting in the subsequent packet loss.

![X Graph](image)

*Fig 6.23 Packet Loss for IP and ISPN Networks*

Packets are transmitted and simply queued if they cannot reach their destination immediately. Because acknowledgements are only produced at quite large time intervals, the amount of traffic is small enough such that queue limits are never exceeded.

### 6.3.3 Two UDP sources and one TCP source

#### 6.3.3.1 Average end-to-end delay

The average end-to-end delay shown by packets when a mixture of TCP and UDP sources is considered exhibits a similar pattern of results as for previous experiments. As expected from a best-effort service such as IP, in the IP network the delay is approximately identical for all traffic flows (see Fig 6.24). The variance in delay is considerable with delays varying between 0.05 and 0.1 seconds. The jitter is caused by the constant production of UDP data and the intermittent production of TCP data.
In the ISPN network, delay for each individual traffic flow is governed by the weight assigned to each queue and the bandwidth reserved by RSVP (see Fig 6.25). As expected from previous results, flow 0 yields the minimal end-to-end delay (i.e., approximately 0.014 seconds) possible. Queuing delays increase the average end-to-end delay of flow 1 packets because of the lower queue priority.

![Fig 6.24 Average End-To-End Delay per IP Flow](image1)

![Fig 6.25 Average End-To-End Delay per ISPN Flow](image2)

As the TCP source begins transmitting packets (i.e., flow 2), the slow-start algorithm shows a small variance in delay. Jitter is reduced after this initial start-up period. The low jitter
provided by RSVP in ISPN networks is also clearly exhibited in Fig 6.25 and Fig 6.26 even though the overall end-to-end delay (i.e., averaged across all flows) is greater than the delay shown for IP networks.

6.3.3.2 Throughput
As expected, the throughput for each flow in the ISPN network exhibits similar results those discussed previously (see Fig 6.27). The bandwidth reserved by RSVP for each flow is the reason for this similarity in results (i.e., 500 Kbps for flow 0, 300 Kbps for flow 1, and 150 Kbps for flow 2).

![Image](image_url)

**Fig 6.28 Throughput per IP Flows**

Fig 6.28 produces interesting results for the throughput of each flow in the IP network. The throughput of each flow is not equally distributed across all flows. This is expected since the constant traffic rate coming from both UDP sources effectively starves the TCP packets. As the TCP data transfer model provides some congestion control, it simply relaxes its sending rate to relieve the network of congestion. The throughput exhibited by the TCP source (i.e., flow 2) varies between 100 and 120 Kbps. UDP source (i.e., flow 0 and flow 1 respectively) have a throughput of approximately 450 Kbps.

The variation in throughput in the IP-enabled is greater because of the interleaving of TCP and UDP packets at the routers buffers. As discussed previously in 6.3.1.2 the overall throughput of both networks show identical results. This is expected (see Fig 6.29), as the number of packets transmitted is restricted by the link capacity (1 Mbps).
6.3.3.3 Packet Loss

Fig 6.30 records some packet loss for all flows in the IP network. As expected, flow 0, and flow 1, exhibit more packet loss because of the continuous transmission of packets. Flow 2 (i.e., the TCP source) exhibits a smaller packet loss as TCP implements a reliable data model. The rate of packet loss is constant for all flows. The amount of packets lost by both UDP sources is similar. This is expected, as the sending rates are identical, and packets are treated equally by IP.
The ISPN network only shows packet loss for a single flow (see Fig 6.31). Flow 1 has a sending rate greater than the bandwidth RSVP reserved. As traffic is transmitted at a constant rate, and UDP does not provide any methods for congestion control, buffers quickly overflow forcing excess packets to be dropped. As discussed previously in 6.3.1.3, no packet loss is shown for flow 0. The congestion control commonly associated with TCP data reduces the packet loss for flow 2.

Fig 6.32 Packet Loss for IP and ISPN Networks
Fig 6.32 shows that the overall number of packets lost in the ISPN network is greater than the in the IP network. The packets lost in the ISPN network come from a single flow. The reason ISPN network exhibiting more packet loss is because the UDP source in the IP network effectively starve the TCP flow from any network resources. Meanwhile the combined sending rate of the UDP sources occupies the entire bandwidth with no little need for queuing. This is also the reason for TCP packets being lost or corrupted in the IP network but not in the ISPN network.

6.4 Conclusions

The results of both the analytical models and the simulation experiment consistently show that RSVP provides enhanced performance to individual traffic flows. The bandwidth reserved by RSVP reduces the number of packets lost, reduces delay, minimises jitter, and enables networks to successfully deliver more data. RSVP not only allows a greater throughput in traffic but also results in the throughput being more consistent. This is most notable when dealing with TCP traffic flows. Providing flows of traffic with consistent throughput, low average end-to-end delays, and minimal jitter, are ideal properties desired when using audio or video applications.

The enhanced performance exhibited by high priority traffic flows is achieved at the expense of lower priority traffic. The performance shown by these lower priority traffic flows is very poor. Large delays, low throughput, and large number of packet losses are commonplace. This accounts for the poor overall performance given by ISPN networks in some of the simulation experiments discussed in this chapter.

The improvements to traffic flows noted above provide QoS to networks that plain IP networks are not capable of achieving. The increase performance and consistency demonstrate that RSVP can be used to establish QoS for any application. As an example, consider the third scenario discussed in 6.3.3.1. If UDP sources (i.e., inelastic applications), starve a TCP source (i.e., an elastic application) of access to the network RSVP can reserve bandwidth for the TCP source so that at least some service is received.
Chapter 7

Evaluation

7.1 Introduction

This chapter introduces the criteria for evaluating the project. The criteria consists of an evaluation of both the analytical and simulation models as well as an evaluation of the project as a whole. An evaluation of the methodology is required so that the results achieved are presented with confidence. Meanwhile, the evaluation of the project will refer back to the minimum requirements and objectives set almost 6 months ago to identify whether or not the project has indeed been a success.

7.2 Analytical Models

Validating the correctness of both models is drawn from the knowledge attained when conducting the literature review. This served to highlight the WFQ scheduler as crucial and instrumental in managing and maintaining the bandwidth reservations for each traffic class in ISPN networks. The operation of traditional IP routers is simple and intuitive when compared to its ISPN counterpart.

The models are built on techniques widely used within the research community to illustrate and compare the performance of systems much more complex than those discussed and presented in this report. Many examples to identify how these principles are put into practice were also investigated to add further weight to the correctness and development of the models.

The development of the models was conducted in a step by step fashion. Microsoft Excel, a spreadsheet software tool, was used to cross-check that the matrices input into Matlab were correct. Until these were confirmed as correct, development of the models were delayed. The performance of both the analytical models and the simulations show similar results and adds further support for the correctness of both modelling techniques.
7.3 Simulation

When evaluating the results achieved from the simulations, it is necessary to validate the correctness of the ns-2 simulator and the RSVP/ns patch. The choice of topology and experimental design must also be checked for consistency.

7.3.1 ns-2

Ns-2 has proved to be a very useful piece of software and is capable of generating accurate and confident results. It is assumed for this project that the correctness and validation for ns-2 has been carried out by more qualified researchers. Its reputation and extensive use for simulations throughout the research community adds further weight to its correctness and robustness.

One major shortcoming of this software is the ns-2’s documentation. As mentioned previously, the documentation is technically sound yet daunting for newcomers. A lot of information included within the documentation discusses how the software actually works and how objects are implemented rather than how to use it and what commands, functions, or protocols are available. Its on-going developmental status also adds to the complexity of the software. Users are required to consult examples and tutorials available on the Internet and in the example directory in order to properly use the various function calls and commands.

Ns-2 appears to lack important functionality expected when using the tool for the analysis and collection of results. Modifying or adding to the underlying C++ code rectifies this, however, the required time to do this would render the project infeasible. Other methods such as post processing of trace files were chosen.

7.3.2 RSVP/ns
As mentioned earlier the version of RSVP/ns used in this project is an updated version of the original developed by Marc Greis. Sean Murphy updated the software in order to fix some minor bugs and to make RSVP/ns compatible with ns2.1b8, the latest version of ns-2.

RSVP/ns has not been formally proved correct but some element of validation has taken place. According to the documentation, expert intuition is used to validate the correctness of RSVP/ns because no complex RSVP test-bed has been available. Creating such a test-bed requires extensive effort. For this reason, a set of simulation experiments to cover several important cases for RSVP was designed to test its correctness.

The experiments used to validate the software are at level well beyond those presented in this project. They give correct results and thus it is assumed that the software is correct and the results achieved can be viewed with confidence.

The supporting documentation for RSVP/ns was hard to get due to its discontinued support. The manual however, proved a valuable source of information. Its main objectives were to help users intending to use RSVP/ns in their simulations and to help those who wanted to understand and extend the underlying code.

The quality associated with the documentation of the patch itself is weak since it only implements a subset of the full functionality detailed in the relevant RFCs for RSVP. The examples included with the patch formed part of the validation tests but also served as good demonstration of the capabilities of the software.

The most notable restriction is that only the controlled load service class has been implemented in the software. This obviously limits the extent to which RSVP can be tested. The manner in which reservations are made and the amount of bandwidth that is reserved for specific flows were chosen to simulate the three possible service classes as closely as possible.

### 7.3.3 Topology Design
The correctness of ns-2 and RSVP/ns was assumed correct leaving only the simulation scripts to be validated. This ensures that the results are a true reflection of the situation or scenario being simulated. The easiest and most intuitive method of checking the correctness of the simulation scripts was to follow packets from individual traffic flows across the network.

This allows for the source and sink of each flow to be paired correctly, to ensure that PATH and RESV states are set up correctly and to verify the times at which traffic was started and stopped. As an extra precaution, RSVP/ns allows upcalls when PATH and RESV states take place. These upcalls display warning messages to the computer screen. These messages give where, when, and what soft states are updated or set up.

Evaluation of each script was carried out as and when each script was produced, making the process an efficient one. As the simulations were monitored from the early stages of their development until their completion, it was easy to keep track of the extra nodes and traffic flows continuously being added. In every simulation or scenario, the data travelling across the network behaved as expected.

It is important to note that the topology used in this project does not reflect a real network or its behaviour. It is simply a single path, known in the research community as the ‘dumbbell’ topology. It would have been more desirable to increase the size and complexity of the topology as well as the amount of traffic flowing across the network to model a more realistic network. Simulating a simple topology however enhances the focus of study to the main objects of interest (i.e., how RSVP provides and maintains bandwidth reservation) and makes the simulation both efficient and accurate.

### 7.4 Project Evaluation

The project has been quite successful since all the objectives were carried out fully. The objectives were structured in such a way that completing them would also complete the minimum requirements.
The first objective was to research and understand about network and Internet protocols. A significant amount of time was spent conducting a literature review. This included textbooks, RFC’s, and technical papers made available on the Internet by other researchers. A good level of understanding was attained in IP, IntServ, DiffServ, MPLS, and RSVP protocols. This built upon knowledge already acquired from previous network related modules taken in the School of Computing, University of Leeds. The first objective was completed and exceeded. In retrospect, maybe too much time was spent reading.

Analytical modelling techniques were covered by Professor Larry Dowdy over a series of lectures throughout the first and second semester. Some of the techniques covered included Markov Modelling and Mean Value Analysis. A considerable number of examples were also investigated so that the principles could be seen in practice. A suitable level of understanding was thus developed in the subject.

The techniques learned were then applied to produce a suite of models from which the performance of IP and ISPN networks could be achieved. This completed the second, and part of the fourth objective. It also served as an indication as to what the results for the simulation would be. One criticism that could be made of the models is that only a small set of results were produced as not enough time was available to investigate the effect of varying arrival rates, service rates and queue weights. This turned out to be one of the most enjoyable and interesting parts of the project.

Ns-2 and RSVP/ns, were the network simulation tools used in this project. Using the various software tools to implement what had been researched meets the third objective (i.e., to learn and apply network simulation tools). On reflection, the amount of time spent learning about the simulator and its operation was limited. If given the chance to repeat the project, this would be an area that the author would improve on. The level of competence attained in the use of simulation tools allowed for the design of a set of simulation experiments to evaluate the performance of IP and ISPN networks. This completed the fourth objective successfully but to a level less than originally envisaged.

The final objective, to evaluate and compare results of simulations and analytical models to determine which protocol performs best under various workloads has been presented and
discussed in this report. The discussion is a substantial piece of this report that meets and exceeds the original objective.

7.5 Summary

This chapter presents and discusses some of the issues, problems, and shortcomings experienced throughout the project. Time management and amendments to the project schedule were strictly adhered to throughout the life of the project. This has contributed greatly to the overall success of the project as all the minimum requirements have been completed.
Chapter 8

Conclusion

8.1 Summary

The results arrived at from both the simulations and analytical models have demonstrated that the IntServ architecture and RSVP protocol offer flows of packets some element of QoS in computer networks. This is an improvement on plain IP networks, be it though at the expense of other, not so demanding, network traffic.

Furthermore, this QoS (reserving of resources, bandwidth) can be explicitly requested by applications before transmission, guaranteeing at least a minimum level of service (i.e., in terms of delay, packet loss, and throughput). Allowing only applications to request QoS results in little or no control being exercised by the administrators or ISP when re-routing traffic around bottlenecks in the network. This problem is addressed, however, by other QoS protocols such as MPLS.

8.2 Future Work

Having carried out experiments in both simulation and analytical modelling to demonstrate the performance of both protocols, the depth to which each was carried out was less than originally planned. This would therefore be an area for important future work. Results, such as, predicting by how much better one protocol is over another, and what are the optimal settings or workload for these protocols could also be accomplished.

A variety of scenarios regularly found in computer networks could also be simulated. How RSVP reacts to link and node failures in a network (i.e., how long does RSVP take to set up new PATH and RESV states) is an interesting scenario. This could be simulated with RSVP messages being treated as best effort packets or as a separate traffic class with its own queue and weight in the WFQ scheduler to measure RSVP scalability. How the RSVP protocol copes with multicasting scenarios is another useful experiment that could be designed.
Investigating other performance metrics, such as jitter, would also be a good way of showing the improvements offered by RSVP and the IntServ architecture. The results obtained in this project imply that RSVP does indeed reduce jitter in networks. This was not one of the chosen metrics in this project but could be considered and conducted independently.

Allowing for the modelling and simulation of real networks such as the University of Leeds Campus Network would also yield better and more confident results. Real world scenarios and applications could also be studied and applied to the simulation topology to give a clear insight as to the differences exhibited by IP and ISPN networks. Recommendations for improvement of the university network can also be put forward because of this research.

Performance evaluation of other QoS network protocols, including ATM, DiffServ, and MPLS could also be carried out using both simulation tools and analytical models. This would allow protocols that have been designed with similar goals and criteria to RSVP to be evaluated and compared against each other.

Such extensions would help in identifying problem areas that these protocols address or under which situations they perform poorly. Finally, recommendations for a hybrid of these protocols that would further improve network performance could also be investigated, modelled and validated.
Bibliography


Dobson S, (2002), *Performance Study of MPLS (Multi-Protocol Label Switching)*, Final Year Project, School of Computing, University of Leeds


Greis M, *RSVP/ns: An Implementation of RSVP for the Network Simulator ns-2*, Computer Science Department IV, University of Bonn


Millar J, (2001), *Performance Evaluation of Active Networks*, Final Year Project, School of Computing, University of Leeds


nam: Network Simulator Animator[Internet],

Nortel Networks, (1998), *IP QoS – A Bold New Network* [Internet], New York

ns-2: Network Simulator [Internet],

OPNET Modeler: Network Simulator [Internet],


Postel J, (1980), *RFC768 - User Datagram Protocol* [Internet].


University of Southern California, (1981), *RFC791 – Internet Protocol* [Internet],


Wroclawski J, (1997), *RFC2210 – Use of RSVP with IETF Integrated Services* [Internet],

xgraph: Network Simulator Graph Utility [Internet],

YATS – Yet Another Tiny Simulator: ATM Simulator [Internet],
Appendix A

Personal Reflection

This project is without a doubt the largest piece of work I have undertaken. As is customary, new tasks and challenges provide the opportunity to learn new skills and further develop those that one originally possesses. This was no different in this project where I feel that I have developed both personally and professionally in the understanding of the subject area as well as more general administrative tasks such as project management. In fact, having carried out a project of this sort has given me the confidence to advise other students with similar projects on a suitable action plan if the opportunity arises.

Time management, organisational skills, and self-motivation are essential skills that are required and are built upon throughout the life of a project. Choosing a project in a subject area that one finds interesting is paramount. It helps in producing both a good report and a successful project as a whole. Having enjoyed previous network-related modules within the School of Computing at the University of Leeds and having a personal interest in computer networks myself meant that this project was ideally suited to my interests.

Also, the recommendations given by past students such as Millar (2001) and Dobson (2002) to take more modules in the first semester than in the second (i.e., 50 and 30 credits respectively) was a great help. This left enough time to complete all the objectives and write up the report during the second semester. Consequently, the first semester was quite stressful at times due to the increased workload.

Installing ns-2 and the RSVP/ns patch was carried out by the support staff of the School of Computing at the University of Leeds. This was a relatively simple task as help was available from the ns-2 website. This software had also previously been installed for other members of staff.

Problems were encountered when acquiring the RSVP/ns patch and its supporting documentation. This led to the simulation experiments being delayed for approximately two
weeks after the original date set in the project schedule. For students who intend on using software tools as part of the methodology it is advisable that they make sure that the software and supporting documentation is readily available. However, time was used effectively as the analytical models were completed fully and parts of the final report were written up.

The delay was caused as RSVP/ns is no longer being developed and supported by its original author Marc Greis. His change of occupation from researcher at the University of Bonn in Germany to Nokia Telecommunications meant that the documentation was hard to get hold of and as his personal website was no longer available. In this respect, the ns-2 mailing list and archive proved to be a valuable resource. I would therefore recommend that all students subscribe to related newsgroups when undertaking a project of any kind.

The development status of ns-2 and the limited functionality of RSVP/ns, meant that restrictions were imposed on the simulation experiments. For this reason I would advice students who intend on carrying out projects using ns-2 or the like to install the software for use during the initial stages of the project. This allows students to become familiar with the software and to be aware of its limitations and capabilities.

A considerable amount of time should also be set aside to learn about the tools and how to use them. This is because, although ns-2 is a very capable piece of software its development status means that the documentation, although technically concise, is not known to be user friendly.

The analytical models presented some minor problems, mainly during their development stages. During this stage of the project the help and advice given by Professor Larry Dowdy was invaluable. His advice was not only instrumental in overcoming these problems but all others that I was faced with throughout the project. Students are therefore encouraged to keep in touch with their supervisor regularly and take their advice seriously.

A lot of the work done with the analytical models was time consuming. Calculations were done manually, as software tools were not readily available leaving less time for the completion of the simulation phase of the project. The schedule was constantly revised throughout the project to cater for these shortfalls. I would therefore advise students to
choose only one methodology (i.e., simulation or analytical modelling). This would give them ample time to master one of the methods rather than work under pressure to attain a suitable standard in both areas.

Writing up the report itself also takes significant time. It is therefore recommended that students become confident with using a word processing package such as Microsoft Word or LaTex. Even when the project is finished, time is required for proof reading so learning how to format documents effectively and efficiently simplifies this task.
Appendix B

Solving the IP Router Markov Model

B.1 Introduction

This appendix illustrates how the Markov model for the IP router presented in chapter 4 is solved through the formation of a set of linear balance equations. It goes on to explain how the solution of this model can be used to predict various performance metrics.

B.2 Model Solution

The model solution refers to the probability of taking a random snapshot of the system and finding the system in a particular state (i.e., the long-term ‘steady state’ probability of being in a particular state). This is found by forming a set of linear balance equations for each state. Therefore, given a Markov model with N states there are N desired unknowns (i.e., the steady state probabilities of being in each state) along with N linear equations (Dowdy, 2002). The solution is a straightforward linear algebra math problem.

The balance equation for each system state represents the fact that overall ‘flow’ into each state must equal the overall flow out of each state. If $p_{xy}$ is the probability of being in state $X, Y$, then the set of linear balance equations for this model is

\[
\begin{align*}
(P20) \times (2\alpha) &= (P11) \times (\mu) \\
((P11) \times (\alpha)) + ((P11) \times (\mu)) &= ((P20) \times (2\alpha)) + ((P02) \times (\mu)) \\
(P02) \times (\mu) &= (P11) \times (\alpha)
\end{align*}
\]

There are three equations and three unknowns. However, any two equations can be used to infer the third equation. One equation is therefore redundant and can be eliminated. To find the unique solution, the redundant equation is replaced by the ‘conservation of total probability equation’ that states that the system must be in one of the system states. That is:
\[ P_{20} + P_{11} + P_{02} = 1 \]

The set of equations now looks like

\((P_{20}) \times (2\alpha) = (P_{11}) \times (\mu)\)

\(((P_{11}) \times (\alpha)) + ((P_{11}) \times (\mu)) = ((P_{20}) \times (2\alpha)) + ((P_{02}) \times (\mu))\)

\[ P_{20} + P_{11} + P_{02} = 1 \]

If \(\alpha = 1\) and \(\mu = 1\) then the solution to this set of linear balance equations is

\[ P_{20} = \frac{1}{5} \]
\[ P_{11} = \frac{2}{5} \]
\[ P_{02} = \frac{2}{5} \]

**B.3 Performance Predictions**

Having arrived at a solution, these probabilities are then interpreted to answer more practical questions such as:

- How often is the router utilised?
  (i.e., this is equivalent to percentage of time the router is busy).
  \[ 1 - P_{20} = \frac{4}{5} = 80\% \text{ of the time} \]
  (This router is therefore idle for 20\% of the time)

- What is the throughput of the router (i.e., average number of packets processed per second)?
  service rate (i.e., \(\mu\)) \times router utilisation
  Throughput \[ \mu \times \frac{4}{5} \]
  \[ = \frac{4}{5} \text{ packets/second} \]

- What is the average number of packets at the queue?
  Queue length \[ \text{weighted average} \]
  \[ = 0P_{20} + 1P_{11} + 2P_{02} \]
\[ = 0 + 2/5 + 2(2/5) \]
\[ = 6/5 \]
\[ = 1.2 \text{ packets} \]

- What is the response time of the system?

Little’s Result (or Little’s Law) states that \( N = XR \). \( N \) is the average number of customers in the system (or queue in this case), \( X \) is the average throughput of the system and \( R \) is the average response time.

\[ \Rightarrow R = \frac{N}{X} \]
\[ R = \frac{6/5}{4/5} \]
\[ = 3/2 \text{ seconds} \]

- Finally what is the power of the system?

(i.e., power = average throughput/average response time).

Power \[ = \frac{4/5}{3/2} \]
\[ = 8/15 \text{ packets/second} \]

**B.4 Summary**

The model, although simple, concise, and intuitive, provides a very powerful method of evaluating the performance of queuing systems such as routers. Furthermore, varying the parameters \( \alpha \) and \( \mu \) allows for analysis of this system under various arrival or service rates. Optimal settings for this router can also be determined. It also highlights possible problem areas in the system and helps the user in understanding how individual system components interact with each other to achieve its intended task (in this context to route packets along the network).
Appendix C

Solving the ISPN Router Markov Model

C.1 Introduction

This appendix illustrates how the Markov model for the ISPN router presented in chapter 4 is solved through the formation of a set of linear balance equations. It highlights the advantages of using a software tool such as Matlab to arrive at the solution to the set of equations and goes on to explain how this solution is used to predict various performance metrics.

C.2 Model Solution

Given that the above Markov model has eight states, there are eight linear equations and eight desired unknowns. Solving a relatively large set of equations is rather tedious, time consuming, and error prone. A software tool such as Matlab provides a faster and more reliable method.

As was the case for the IP system the linear balance equations for each system state represents the fact that overall ‘flow’ into each state must equal the overall flow out of each state. If \( P_{abcxyz} \) is the probability of being in state A, B, C, X, Y, Z, then the set of linear balance equations for this model are:

\[
\begin{align*}
P_{111000} (\alpha_1 + \alpha_2 + \alpha_3) & = \mu_1 P_{011100} + \mu_2 P_{101010} + \mu_3 P_{110001} \\
P_{011100} (\mu_1 + \alpha_2 + \alpha_3) & = \alpha_1 P_{111000} + 2/5 \mu_2 P_{001110} + 1/4 \mu_3 P_{010101} \\
P_{101010} (\mu_2 + \alpha_1 + \alpha_3) & = \alpha_2 P_{111000} + 3/5 \mu_1 P_{001110} + 1/3 \mu_3 P_{100011} \\
P_{110001} (\mu_3 + \alpha_1 + \alpha_2) & = \alpha_3 P_{111000} + 3/4 \mu_1 P_{010101} + 2/3 \mu_2 P_{100011} \\
P_{001110} (2/5 \mu_2 + 3/5 \mu_1 + \alpha_3) & = \alpha_3 P_{011110} + \alpha_1 P_{101010} + 1/6 \mu_3 P_{000111} \\
P_{010101} (1/4 \mu_3 + 3/4 \mu_1 + \alpha_2) & = \alpha_3 P_{011100} + \alpha_1 P_{110001} + 1/3 \mu_2 P_{000111} \\
P_{100011} (1/3 \mu_3 + 2/3 \mu_2 + \alpha_1) & = \alpha_3 P_{101010} + \alpha_2 P_{110001} + 1/2 \mu_1 P_{000111} \\
P_{000111} (1/6 \mu_3 + 1/3 \mu_2 + 1/2 \mu_1) & = \alpha_3 P_{001110} + \alpha_2 P_{010101} + \alpha_1 P_{100011}
\end{align*}
\]
As noted in appendix B, one of the equations is redundant. Arbitrarily replacing one of the above equations, say, the last equation, with:

\[ P_{111000} + P_{011100} + P_{101010} + P_{110001} + P_{001110} + P_{010101} + P_{100011} + P_{000111} = 1 \]

the ‘conservation of total probability equation’, the set of equations now looks like:

\[-(\alpha_1 + \alpha_2 + \alpha_3)P_{111000} + \mu_1P_{011100} + \mu_2P_{101010} + \mu_3P_{110001} + 0 + 0 + 0 = 0 \]
\[ \alpha_1 P_{111000} - (\mu_1 + \alpha_2 + \alpha_3)P_{011100} + 0 + 0 + 2/5\mu_2P_{001110} + 1/4\mu_3P_{010101} + 0 + 0 = 0 \]
\[ \alpha_2 P_{111000} + 0 - (\mu_2 + \alpha_1 + \alpha_3)P_{101010} + 0 + 3/5\mu_1P_{001110} + 0 + 1/3\mu_3P_{100011} + 0 = 0 \]
\[ \alpha_3 P_{111000} + 0 + 0 - (\mu_3 + \alpha_1 + \alpha_2)P_{110001} + 0 + 3/4\mu_1P_{010101} + 2/3\mu_2P_{100011} + 0 = 0 \]
\[ 0 + \alpha_2P_{011100} + \alpha_1P_{101010} + 0 - (2/5\mu_2 + 3/5\mu_1 + \alpha_3)P_{001110} + 0 + 0 + 1/6\mu_3P_{000111} = 0 \]
\[ 0 + \alpha_3P_{011100} + 0 + \alpha_1P_{110001} + 0 - (1/4\mu_1 + 3/4\mu_2 + \alpha_2)P_{010101} + 0 + 1/3\mu_2P_{000111} = 0 \]
\[ 0 + 0 + 0 + \alpha_2P_{110001} + 0 + \alpha_3P_{101010} - (1/3\mu_3 + 2/3\mu_2 + \alpha_1)P_{100001} + 1/2\mu_1P_{000111} = 0 \]
\[ P_{111000} + P_{011100} + P_{101010} + P_{110001} + P_{001110} + P_{010101} + P_{100011} + P_{000111} = 1 \]

Given that

- \( \alpha_1, \alpha_2 \) and \( \alpha_3 = 1 \) (i.e., they all arrive at the same rate)
- \( \mu_1, \mu_2 \) and \( \mu_3 = 1 \) (i.e., they are all serviced at the same rate)

the equations can be written in the matrix form \( AX=B \), where

\[
A = \begin{bmatrix}
-3 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
1 & -3 & 0 & 0 & 2/5 & 1/4 & 0 & 0 \\
1 & 0 & -3 & 0 & 3/5 & 0 & 1/3 & 0 \\
0 & 1 & 1 & 1 & 0 & -2 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & -2 & 0 & 1/3 \\
0 & 0 & 1 & 1 & 0 & 0 & -2 & 1/2 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\end{bmatrix}
\]
\[
X = \begin{bmatrix}
P_{111000} \\
P_{011100} \\
P_{101010} \\
P_{110001} \\
P_{001110} \\
P_{010101} \\
P_{100011} \\
P_{000111} \end{bmatrix}
\]
\[
B = \begin{bmatrix}
0 \\
0 \\
0 \\
1/6 \\
0 \\
0 \\
0 \\
1 \end{bmatrix}
\]

Solving this problem (i.e., finding the unknown vector \( X \), and the steady state probabilities) simply requires that the matrices \( A \) and \( B \) be entered into Matlab. The solution to this set of linear balance equations is
P111000 = 0.0625
P011100 = 0.0422
P101010 = 0.0553
P110001 = 0.09
P001110 = 0.08
P010101 = 0.1286
P100011 = 0.1664
P000111 = 0.375

C.3 Performance Predictions

Having arrived at the solution, the probabilities can be interpreted to answer more practical questions such as:

• How often is the router utilised?
  (i.e., this is equivalent to percentage of time the router is busy).
  \[1 - P111000 = 0.9375\%\] of the time
  (The router is therefore idle for 0.0625\% of the time)

The following performance metrics are calculated on a per flow basis (i.e., for an individual class of traffic). Only the results for class 1 packets are shown here. They demonstrate how the results are achieved. The same methodology can be used to deduce the performance for the remaining two classes. Summing the results of the performance metrics for each traffic class allows the performance of the ISPN router (as a whole) to be compared to the IP router.

• What is the throughput of class 1 packets the router (i.e., average number of class 1 packets processed per second)?

  (i.e., state probability of class 1 packet at queue * probability of being served)
  \[
  \text{Throughput} = P011100 + P001110 \times 3/5 + P010101 \times 3/4 + P000111 \times 1/2
  \]
  \[= 0.3742 \text{ packets/second}\]
• What is the average number of class 1 packets at the queue?

Queue length = weighted average of states with class 1 packets at the queue

\[
= 0P1110000 + 1P011100 + 1P001110 + 1P010101 + 1P000111
= 0 + 0.0422 + 0.08 + 0.1286 + 0.375
= 0.6258 \text{ packets}
\]

• What is the response time of the system?

Given that Little’s Result (or Little’s Law) states that \( N = XR \)

\[ R = \frac{N}{X} \]

\[
R = \frac{0.6258}{0.3742}
= 1.6724 \text{ seconds}
\]

• Finally, what is the power associated with queue1?

(i.e., average throughput/average response time).

\[
\text{Power} = \frac{0.3742}{1.6724}
= 0.2238 \text{ packets/second}
\]

C.4 Summary

The model although simple, concise, and intuitive, provides a very powerful method of evaluating the performance of routers. Varying the parameters \( \alpha \) and \( \mu \) for each traffic class allows analysis of this system under varying arrival or service rates. Optimal settings for this router can also be derived. It also highlights possible problem areas in the system and helps the user in understanding how individual system components interact with each other to achieve its intended task (in this context, to route packets along the network).
Appendix D

Tabulated Results for Analytical Models

D.1 Introduction

This appendix presents both the steady state probabilities and the performance results for the analytical models developed in chapter 4. The results are shown in tabular form. The parameters used are outlined in Table 4.3. Table D.1 displays the results for the router as a whole (i.e., over all traffic classes) while Table D.2 summarises the results for each traffic class.

D.2 IP Model Steady State Probabilities

<table>
<thead>
<tr>
<th>States</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.6667</td>
</tr>
<tr>
<td>01</td>
<td>0.3333</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MPL = 1</th>
<th>MPL = 2</th>
<th>MPL = 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>States</td>
<td>Probability</td>
<td>States</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
<td>-------</td>
</tr>
<tr>
<td>10</td>
<td>0.6667</td>
<td>20</td>
</tr>
<tr>
<td>11</td>
<td>0.4</td>
<td>21</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>States</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>02</td>
<td>0.2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>States</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>03</td>
<td>0.1579</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MPL = 6</th>
<th>MPL = 9</th>
</tr>
</thead>
<tbody>
<tr>
<td>States</td>
<td>Probability</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>60</td>
<td>0.0121</td>
</tr>
<tr>
<td>51</td>
<td>0.0363</td>
</tr>
<tr>
<td>42</td>
<td>0.0906</td>
</tr>
<tr>
<td>33</td>
<td>0.1813</td>
</tr>
<tr>
<td>24</td>
<td>0.2719</td>
</tr>
<tr>
<td>15</td>
<td>0.2719</td>
</tr>
<tr>
<td>06</td>
<td>0.136</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### D.3 ISPN Model Steady State Probabilities

<table>
<thead>
<tr>
<th>States</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>111000</td>
<td>0.2105</td>
</tr>
<tr>
<td>011100</td>
<td>0.0816</td>
</tr>
<tr>
<td>101010</td>
<td>0.0976</td>
</tr>
<tr>
<td>110001</td>
<td>0.1366</td>
</tr>
<tr>
<td>001110</td>
<td>0.0773</td>
</tr>
<tr>
<td>010101</td>
<td>0.1078</td>
</tr>
<tr>
<td>100011</td>
<td>0.1307</td>
</tr>
<tr>
<td>000111</td>
<td>0.1579</td>
</tr>
</tbody>
</table>

**MPL = 1 for traffic classes**

**MPL = 3 for system**

<table>
<thead>
<tr>
<th>States</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>333000</td>
<td>0.0002</td>
</tr>
<tr>
<td>233100</td>
<td>0.0001</td>
</tr>
<tr>
<td>323010</td>
<td>0.0002</td>
</tr>
<tr>
<td>332001</td>
<td>0.0005</td>
</tr>
<tr>
<td>133200</td>
<td>0.0001</td>
</tr>
<tr>
<td>223110</td>
<td>0.0003</td>
</tr>
<tr>
<td>232101</td>
<td>0.0007</td>
</tr>
<tr>
<td>313020</td>
<td>0.0002</td>
</tr>
<tr>
<td>322011</td>
<td>0.0012</td>
</tr>
<tr>
<td>331002</td>
<td>0.001</td>
</tr>
<tr>
<td>033300</td>
<td>0.0001</td>
</tr>
<tr>
<td>123210</td>
<td>0.0003</td>
</tr>
<tr>
<td>132201</td>
<td>0.0006</td>
</tr>
<tr>
<td>213120</td>
<td>0.0004</td>
</tr>
<tr>
<td>222111</td>
<td>0.0028</td>
</tr>
</tbody>
</table>

**MPL = 2 for traffic classes**

**MPL = 6 for system**

<table>
<thead>
<tr>
<th>States</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>222000</td>
<td>0.0121</td>
</tr>
<tr>
<td>122100</td>
<td>0.0072</td>
</tr>
<tr>
<td>212010</td>
<td>0.01</td>
</tr>
<tr>
<td>221001</td>
<td>0.0191</td>
</tr>
<tr>
<td>022200</td>
<td>0.0023</td>
</tr>
<tr>
<td>112110</td>
<td>0.0125</td>
</tr>
<tr>
<td>121101</td>
<td>0.0228</td>
</tr>
<tr>
<td>202020</td>
<td>0.0044</td>
</tr>
<tr>
<td>211011</td>
<td>0.0329</td>
</tr>
<tr>
<td>220002</td>
<td>0.0158</td>
</tr>
<tr>
<td>012210</td>
<td>0.0063</td>
</tr>
<tr>
<td>021201</td>
<td>0.0113</td>
</tr>
<tr>
<td>102120</td>
<td>0.0087</td>
</tr>
<tr>
<td>111111</td>
<td>0.063</td>
</tr>
<tr>
<td>120102</td>
<td>0.0288</td>
</tr>
<tr>
<td>201021</td>
<td>0.0222</td>
</tr>
<tr>
<td>210012</td>
<td>0.0409</td>
</tr>
<tr>
<td>002220</td>
<td>0.0062</td>
</tr>
<tr>
<td>011211</td>
<td>0.0436</td>
</tr>
<tr>
<td>020202</td>
<td>0.0193</td>
</tr>
<tr>
<td>101121</td>
<td>0.0598</td>
</tr>
<tr>
<td>110112</td>
<td>0.1063</td>
</tr>
<tr>
<td>200022</td>
<td>0.0366</td>
</tr>
<tr>
<td>001221</td>
<td>0.0537</td>
</tr>
<tr>
<td>010212</td>
<td>0.0931</td>
</tr>
<tr>
<td>100122</td>
<td>0.1251</td>
</tr>
<tr>
<td>000222</td>
<td>0.136</td>
</tr>
</tbody>
</table>

**MPL = 9 for system**
### D.4 Performance Results

<table>
<thead>
<tr>
<th>Metric</th>
<th>IP Model</th>
<th>ISPN Model</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MPL</td>
<td>MPL</td>
</tr>
<tr>
<td></td>
<td>0,0,1</td>
<td>0,1,1</td>
</tr>
<tr>
<td>IDLE</td>
<td>0.6667</td>
<td>0.4</td>
</tr>
<tr>
<td>UTIL</td>
<td>0.3333</td>
<td>0.6</td>
</tr>
<tr>
<td>TPUT</td>
<td>0.6667</td>
<td>1.2</td>
</tr>
<tr>
<td>QLEN</td>
<td>0.3333</td>
<td>0.8</td>
</tr>
<tr>
<td>RESPT</td>
<td>0.4999</td>
<td>0.6667</td>
</tr>
<tr>
<td>POWER</td>
<td>1.3337</td>
<td>1.7999</td>
</tr>
</tbody>
</table>

*Table D.1 Results for IP and ISPN Routers as a Whole*

<table>
<thead>
<tr>
<th>Metric</th>
<th>IP Model</th>
<th>ISPN Model</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MPL</td>
<td>MPL</td>
</tr>
<tr>
<td></td>
<td>1,1,1</td>
<td>2,2,2</td>
</tr>
<tr>
<td>TPUT</td>
<td>0.5263</td>
<td>0.6586</td>
</tr>
<tr>
<td>QLEN</td>
<td>0.4737</td>
<td>1.3415</td>
</tr>
<tr>
<td>RESPT</td>
<td>0.9</td>
<td>2.0369</td>
</tr>
<tr>
<td>POWER</td>
<td>0.5848</td>
<td>0.3233</td>
</tr>
</tbody>
</table>

*Table D.2 Results for IP and ISPN Routers per Packet Class*
Appendix E

ns-2 Tcl Scripts

E.1 Introduction

This appendix contains a subset of the OTcl scripts written for this project. Only one topology is used to test the performance of IP and ISPN networks. Not all scripts have been included as changes from one simulation to another where minor in terms of code.

The first script is the IP topology with two UDP sources simulating audio and video streams (i.e., CBR traffic) and a TCP source exhibiting FTP traffic. The second script shows the ISPN topology with the same number of sources producing the same traffic but with each flow reserving a portion of the links bandwidth as detailed in chapter 5.

E.2 IP Topology

# Jonathan Galliano
# University of Leeds
# Performance Evaluation of IntServ & RSVP
# 21/4/03
#
# Simple topology with IP nodes
#
$defaultRNG seed 1

# create a simulator object
set ns [new Simulator]

# colours for traffic flows in nam
$ns color 0 red
$ns color 1 green
$ns color 2 blue

# open trace file for gathering statistics
set tf [open /tmp/simple_ip_both.tr w]
$ns trace-all $tf
# nam trace generation
set nf [open /tmp/out.nam w]
$ns namtrace-all $nf

# create nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
set n6 [$ns node]
set n7 [$ns node]

# link nodes with 10Mb links and a delay of 1ms
$ns duplex-link $n0 $n3 10Mb 1ms DropTail
$ns duplex-link $n1 $n3 10Mb 1ms DropTail
$ns duplex-link $n2 $n3 10Mb 1ms DropTail
$ns duplex-link $n3 $n4 1Mb 10ms DropTail
$ns duplex-link $n4 $n5 10Mb 1ms DropTail
$ns duplex-link $n4 $n6 10Mb 1ms DropTail
$ns duplex-link $n4 $n7 10Mb 1ms DropTail

# node position for nam
$ns duplex-link-op $n0 $n3 orient right-down
$ns duplex-link-op $n1 $n3 orient right
$ns duplex-link-op $n2 $n3 orient right-up
$ns duplex-link-op $n3 $n4 orient right
$ns duplex-link-op $n4 $n5 orient right-up
$ns duplex-link-op $n4 $n6 orient right
$ns duplex-link-op $n4 $n7 orient right-down

# monitor queue in link n3 – n4 (for nam)
$ns duplex-link-op $n3 $n4 queuePos 0.5

# create FTP & CBR traffic generators over TCP & UDP agents respectively
set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0
$udp0 set fid_ 0

set cbr0 [new Application/Traffic/CBR]
# $cbr0 set interval_ 0.00375
$cbr0 set random_ 1
$cbr0 attach-agent $udp0

set udp1 [new Agent/UDP]
$ns attach-agent $n1 $udp1
$udp1 set fid_ 1
set cbr1 [new Application/Traffic/CBR]
# $cbr1 set interval_ 0.00375
$cbr1 set random_ 1
$cbr1 attach-agent $udp1

set tcp2 [new Agent/TCP]
$ns attach-agent $n2 $tcp2
$tcp2 set fid_ 2

set ftp2 [new Application/FTP]
$ftp2 attach-agent $tcp2

# create traffic sinks
set sink0 [new Agent/Null]
$ns attach-agent $n5 $sink0

set sink1 [new Agent/Null]
$ns attach-agent $n6 $sink1

set sink2 [new Agent/TCPSink]
$ns attach-agent $n7 $sink2

# connect source and sink
$ns connect $udp0 $sink0
$ns connect $udp1 $sink1
$ns connect $tcp2 $sink2

# start/stop traffic
$ns at 10.0 "$cbr0 start"
$ns at 44.0 "$cbr0 stop"
$ns at 13.0 "$cbr1 start"
$ns at 47.0 "$cbr1 stop"
$ns at 16.0 "$ftp2 start"
$ns at 50.0 "$ftp2 stop"

# Call the finish procedure after traffic stops
$ns at 60.0 "finish"

# define a 'finish' procedure
proc finish {} {
    global ns nf tf
    $ns flush-trace
    close $nf
    close $tf
    puts "Done."
    exec nam /tmp/out.nam &
    exit 0
}
# start simulation
$ns run

E.3 RSVP Topology

# Jonathan Galliano
# University of Leeds
# Performance Evaluation of IntServ & RSVP
# 21/4/03
#
# Simple topology with RSVP nodes
#
$defaultRNG seed 1

# create simulator object
set ns [new Simulator]

# colours for traffic flows in nam
$ns color 0 red
$ns color 1 green
$ns color 2 blue
$ns color 46 purple

# open trace file for gathering statistics
set tf [open /tmp/simple_rsvp_both.tr w]
$ns trace-all $tf

# trace files for PATH and RESV states on bottleneck router
set ff [open /tmp/flow.tr w]
set pf [open /tmp/path.tr w]

# nam trace generation
set nf [open /tmp/out.nam w]
$ns namtrace-all $nf

# create nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
set n6 [$ns node]
set n7 [$ns node]

# link nodes with 10Mb links and a delay of 1ms
# 1000 bit/sec is reserved for RSVP PATH and RESV messages
$\text{n} \text{s} \\text{d} \text{u} \text{p} \text{l} \text{e} \text{x}-\text{r} \text{s} \text{v} \text{p}-\text{lin} \text{k} $ \text{n} 0 $ \text{n} 3 10\text{Mb} 1\text{ms} 0.99 1000 50000 \text{Par} \text{m} \text{Null}

$\text{n} \text{s} \\text{d} \text{u} \text{p} \text{l} \text{e} \text{x}-\text{r} \text{s} \text{v} \text{p}-\text{lin} \text{k} $ \text{n} 1 $ \text{n} 3 10\text{Mb} 1\text{ms} 0.99 1000 50000 \text{Par} \text{m} \text{Null}

$\text{n} \text{s} \\text{d} \text{u} \text{p} \text{l} \text{e} \text{x}-\text{r} \text{s} \text{v} \text{p}-\text{lin} \text{k} $ \text{n} 2 $ \text{n} 3 10\text{Mb} 1\text{ms} 0.99 1000 50000 \text{Par} \text{m} \text{Null}

$\text{n} \text{s} \\text{d} \text{u} \text{p} \text{l} \text{e} \text{x}-\text{r} \text{s} \text{v} \text{p}-\text{lin} \text{k} $ \text{n} 3 $ \text{n} 4 1\text{Mb} 10\text{ms} 0.99 1000 50000 \text{Par} \text{m} \text{Null}

$\text{n} \text{s} \\text{d} \text{u} \text{p} \text{l} \text{e} \text{x}-\text{r} \text{s} \text{v} \text{p}-\text{lin} \text{k} $ \text{n} 4 $ \text{n} 5 10\text{Mb} 1\text{ms} 0.99 1000 50000 \text{Par} \text{m} \text{Null}

$\text{n} \text{s} \\text{d} \text{u} \text{p} \text{l} \text{e} \text{x}-\text{r} \text{s} \text{v} \text{p}-\text{lin} \text{k} $ \text{n} 4 $ \text{n} 6 10\text{Mb} 1\text{ms} 0.99 1000 50000 \text{Par} \text{m} \text{Null}

$\text{n} \text{s} \\text{d} \text{u} \text{p} \text{l} \text{e} \text{x}-\text{r} \text{s} \text{v} \text{p}-\text{lin} \text{k} $ \text{n} 4 $ \text{n} 7 10\text{Mb} 1\text{ms} 0.99 1000 50000 \text{Par} \text{m} \text{Null}

# node positions for nam
$\text{n} \text{s} \\text{d} \text{u} \text{p} \text{l} \text{e} \text{x}-\text{lin} \text{k}-\text{o} \text{p} $ \text{n} 0 $ \text{n} 3 \text{orient right-down}
$\text{n} \text{s} \\text{d} \text{u} \text{p} \text{l} \text{e} \text{x}-\text{lin} \text{k}-\text{o} \text{p} $ \text{n} 1 $ \text{n} 3 \text{orient right}
$\text{n} \text{s} \\text{d} \text{u} \text{p} \text{l} \text{e} \text{x}-\text{lin} \text{k}-\text{o} \text{p} $ \text{n} 2 $ \text{n} 3 \text{orient right-up}
$\text{n} \text{s} \\text{d} \text{u} \text{p} \text{l} \text{e} \text{x}-\text{lin} \text{k}-\text{o} \text{p} $ \text{n} 3 $ \text{n} 4 \text{orient right}
$\text{n} \text{s} \\text{d} \text{u} \text{p} \text{l} \text{e} \text{x}-\text{lin} \text{k}-\text{o} \text{p} $ \text{n} 4 $ \text{n} 5 \text{orient right-up}
$\text{n} \text{s} \\text{d} \text{u} \text{p} \text{l} \text{e} \text{x}-\text{lin} \text{k}-\text{o} \text{p} $ \text{n} 4 $ \text{n} 6 \text{orient right}
$\text{n} \text{s} \\text{d} \text{u} \text{p} \text{l} \text{e} \text{x}-\text{lin} \text{k}-\text{o} \text{p} $ \text{n} 4 $ \text{n} 7 \text{orient right-down}

# monitor queue in link n3 – n4 (for nam)
$\text{n} \text{s} \\text{d} \text{u} \text{p} \text{l} \text{e} \text{x}-\text{lin} \text{k}-\text{o} \text{p} $ \text{n} 3 $ \text{n} 4 \text{queuePos} 0.5

# enable upcalls on all nodes
Agent/RSVP set noisy_ 255

# set refresh intervals of soft state at nodes to 5 secs
Agent/RSVP set refresh_ 5

set rsvp0 [$n0 add-rsvp-agent]
set rsvp1 [$n1 add-rsvp-agent]
set rsvp2 [$n2 add-rsvp-agent]
set rsvp3 [$n3 add-rsvp-agent]
set rsvp4 [$n4 add-rsvp-agent]
set rsvp5 [$n5 add-rsvp-agent]
set rsvp6 [$n6 add-rsvp-agent]
set rsvp7 [$n7 add-rsvp-agent]

# create FTP & CBR traffic over TCP & UDP agents respectively
set udp0 [new Agent/UDP]
$\text{n} \text{s} \text{a} \text{t} \text{t} \text{a} \text{c} \text{h}-\text{a} \text{g} \text{e} \text{nt} $ $\text{n} \text{o} $ $\text{u} \text{d} \text{p} \text{0}$
$\text{u} \text{d} \text{p} \text{0} \text{s} \text{e} \text{t} \text{f} \text{i} \text{d}_ \text{0}$

set cbr0 [new Application/Traffic/CBR]
# $\text{c} \text{b} \text{r} \text{0}$ $\text{s} \text{e} \text{t}$ $\text{i} \text{n} \text{t} \text{e} \text{r} \text{a} \text{l}_ \text{0.00375}$
$\text{c} \text{b} \text{r} \text{0}$ $\text{s} \text{e} \text{t}$ $\text{r} \text{a} \text{t} \text{e}_ \text{500k}$
$\text{c} \text{b} \text{r} \text{0}$ $\text{s} \text{e} \text{t}$ $\text{r} \text{a} \text{n} \text{d} \text{om}_ \text{1}$
$\text{c} \text{b} \text{r} \text{0}$ $\text{a} \text{t} \text{t} \text{a} \text{c} \text{h} \text{-e} \text{g} \text{e} \text{n} \text{t} \text{u} \text{d} \text{p} \text{0}$

set udp1 [new Agent/UDP]
$\text{n} \text{s} \text{a} \text{t} \text{t} \text{a} \text{c} \text{h}-\text{a} \text{g} \text{e} \text{nt} $ $\text{n} \text{1} $ $\text{u} \text{d} \text{p} \text{1}$
$\text{u} \text{d} \text{p} \text{1} \text{s} \text{e} \text{t} \text{f} \text{i} \text{d}_ \text{1}$
set cbr1 [new Application/Traffic/CBR]
# $cbr1 set interval_ 0.00375
$cbr1 set rate_ 500k
$cbr1 set random_ 1
$cbr1 attach-agent $udp1

set tcp2 [new Agent/TCP]
$ns attach-agent $n2 $tcp2
$tcp2 set fid_ 2

set ftp2 [new Application/FTP]
$ftp2 attach-agent $tcp2

# create traffic sinks
set sink0 [new Agent/Null]
$ns attach-agent $n5 $sink0

set sink1 [new Agent/Null]
$ns attach-agent $n6 $sink1

set sink2 [new Agent/TCPSink]
$ns attach-agent $n7 $sink2

# connect source and sink
$ns connect $udp0 $sink0
$ns connect $udp1 $sink1
$ns connect $tcp2 $sink2

# create sessions on all sender nodes
$rsvp0 session $n5 0
$rsvp1 session $n6 1
$rsvp2 session $n7 2

set counter 0
$ns at 0.0 "record"

# generate PATH messages
$ns at 5.0 "$rsvp0 sender 0 +1000000 100000 32"
$ns at 8.0 "$rsvp1 sender 0 +1000000 100000 32"
$ns at 11.0 "$rsvp2 sender 0 +1000000 100000 32"

# perform reservations
$ns at 7.0 "$rsvp5 reserve 0 ff +500000 100000 $n0"
$ns at 11.0 "$rsvp6 reserve 0 ff +300000 100000 $n1"
$ns at 13.0 "$rsvp7 reserve 0 ff +150000 100000 $n2"

# start/stop traffic
$ns at 10.0 "$cbr0 start"
$\texttt{ns at 44.0 "$cbr0 stop"}$
$\texttt{ns at 13.0 "$cbr1 start"}$
$\texttt{ns at 47.0 "$cbr1 stop"}$
$\texttt{ns at 16.0 "$ftp2 start"}$
$\texttt{ns at 50.0 "$ftp2 stop"}$

# call finish procedure after traffic stops
$\texttt{ns at 60.0 "finish"}$

# procedure that counts how many PATH and RESV states
proc record {} {
    \texttt{global rsvp3 ff pf ns counter}
    \texttt{set time 1.0}
    \texttt{set path [rsvp3 set num_psb_]}
    \texttt{set resv [rsvp3 set num_flows_]}
    \texttt{set now [ns now]}
    \texttt{puts pf "$now $path"}
    \texttt{puts ff "$now $resv"}
    \texttt{\$ns at [expr $now + $time] "record"}
    \texttt{if { $counter == 10 } {}
        \texttt{puts "Time: $now"}
        \texttt{set counter 0}
    }
    \texttt{incr counter}
}

# finish procedure that closes trace files and opens nam
proc finish {} {
    \texttt{global ns nf tf ff pf}
    \texttt{\$ns flush-trace}
    \texttt{close $nf}
    \texttt{close $tf}
    \texttt{close $ff}
    \texttt{close $pf}
    \texttt{puts "Done."}
    \texttt{exec nam /tmp/out.nam &}
    \texttt{exit 0}
}

# start simulation
\texttt{\$ns run}
Appendix F

AWK Scripts

F.1 Introduction

This appendix includes the scripts needed to post process the trace files produced by ns-2 to obtain the performance metrics.

F.2 Delay

# awk script to generate packet delay
#
BEGIN {
    highest_packet_id = 0;
}
{
    action = $1;
    time = $2;
    from_node = $3;
    to_node = $4;
    traffic_type = $5;
    packet_size = $6;
    flow_id = $8;
    source = $9;
    destination = $10;
    seq_no = $11;
    packet_id = $12;

    if (packet_id > highest_packet_id) {
        highest_packet_id = packet_id;
    }

    if (start_time[packet_id] == 0) {
        start_time[packet_id] = time;
    }

    if (action != "d") {
        if (action == "r" && (to_node == "5" || to_node == "6" || to_node == "7")) {
            # if (action == "r" && traffic_type == "cbr" && to_node == "5") {
                end_time[packet_id] = time;
        }
    }
}
} else {
    end_time[packet_id] = -1;
}
}
}

END {
    for (packet_id = 0; packet_id <= highest_packet_id; packet_id++) {
        start = start_time[packet_id];
        end = end_time[packet_id];
        packet_duration = end - start;
        if (start < end) {
            printf("%d %f\n", start, packet_duration);
        }
    }
}

# awk script to calculate average packet delay
#
BEGIN {
    average_delay = 0;
}

{
    time = $1;
    delay = $2;

    sum_of_delay[time] += delay;
    number[time] += 1;
}

END {
    for (i = 0; i < 60; i+=1) {
        if (number[i] != 0) {
            average_delay = sum_of_delay[i]/number[i];
            printf("%d %f\n", i, average_delay);
        }
    }
}

F.3 Packet Loss

# awk script to generate total packet loss
#
BEGIN {


highest_packet_id = 0;
current_id = 0;
highest_id = 0;
}
{
 action = $1;
time = $2;
from_node = $3;
to_node = $4;
traffic_type = $5;
packet_size = $6;
flow_id = $8;
source = $9;
destination = $10;
seq_no = $11;
packet_id = $12;

if (action == "d" && (flow_id == "0" || flow_id == "1" || flow_id == "2")) {

 # if (action == "d" && flow_id == "0") {
    if (packet_id > highest_packet_id) {
      highest_packet_id += 1;
      current_id += 1;
      highest_id = packet_id;
      time_dropped[current_id] = time;
    }
  }
}
END {

for (current_id = 0; current_id <= highest_id; current_id++) {
  if (current_id <= highest_packet_id) {
    drop_time = time_dropped[current_id];
    printf("%d %d\n", drop_time, current_id);
  }
}
}

F.4 Throughput

# awk script to generate packet throughput in bytes
#
BEGIN {
  highest_packet_id = 0;
  current_id = 0;
}
{ 
    action = $1;
    time = $2;
    from_node = $3;
    to_node = $4;
    traffic_type = $5;
    packet_size = $6;
    flow_id = $8;
    source = $9;
    destination = $10;
    seq_no = $11;
    packet_id = $12;
}

if (action == "r" && (to_node == "5" || to_node == "6" || to_node == "7")) {
    #if (action == "r" && traffic_type == "cbr" && to_node == "5") {
        highest_packet_id += 1;
        current_id += 1;
        time_received[current_id] = time;
        size_of_packet[current_id] = packet_size;
    }
}

END {
    for (current_id = 0; current_id <= highest_packet_id; current_id++) {
        received_time = time_received[current_id];
        bytes = size_of_packet[current_id];
        printf("%d %.2f\n", received_time, bytes);
    }
}

# awk script to calculate total bandwidth at 1 sec intervals in bits
# uses output produced by throughput.awk
#
BEGIN {
    bytes = 0;
}

{
    time = $1;
    size = $2;
    bandwidth[time] += size;
}

END {
    for (i = 0; i < 60; i+=1) {
        bytes = bandwidth[i];
        }
bits = bytes*8;
printf("%d %.2f\n", i, bits);
}