An Investigation into Buffer Management Mechanisms for the Diffserv Assured Forwarding Traffic Class

Prepared by:
Joshua Mentz

Supervised by:
Mr. Neco Ventura

Department of Electrical Engineering
University of Cape Town
September 2003

This dissertation is submitted to the University of Cape Town in fulfilment of the academic requirements for the Degree of Master of Science in Engineering.

Declaration

I declare that this thesis is my own work. Where information from other sources has been used herein the relevant material has been referred to in the references.

This work is being submitted for the Master of Science Degree in Electrical Engineering at the University of Cape Town. It has not been submitted to any other university for any other degree or examination.

Joshua Mentz

Signed: Signed by candidate Date: 02-09-03

Acknowledgements

I would like to thank Siemens and the National Research Foundation for providing financial support to me during this project.

Thank you to Neco Ventura for giving guidance and for providing a conducive learning environment. Thank you to Sven Shepstone for providing invaluable support during the formative stages of my project. Thank you to the folks at The Applied Research laboratory, Washington University for creating the MSR research platform and for taking the time to address my queries. Thanks also to Adam Burke for an initial collaboration that spurred me on.

Thank you to my family for your rich support. Thank you also to Rowena Quinan and to Hendrik Mentz for the hours spent proof reading.

Synopsis

The IETF proposed the Diffserv architecture as a means to provide preferential service for delay and loss sensitive applications. One of the service classes offered by Diffserv is the Assured Forwarding (AF) class. Because of scalability concerns, IETF specifications recommend that microflow and aggregate-unaware active buffer management mechanisms such as RIO (Random early detection with In/Out-of-profile) be used in the core of Diffserv networks implementing AF. Such mechanisms have, however, been shown to provide poor performance with regard to fairness, stability and network control. Furthermore, recent advances in router technology now allow routers to implement more advanced scheduling and buffer management mechanisms on high-speed ports.

This thesis evaluates the performance improvements that may be realized when implementing the Diffserv AF core using a hierarchical microflow and aggregate-aware buffer management mechanism instead of RIO. The author motivates, proposes and specifies such a mechanism. The mechanism, referred to as H-MAQ or Hierarchical multi drop-precedence queue state Microflow-Aware Queuing, is evaluated on a testbed that compares the performance of a RIO network core with an H-MAQ network core. The results of these evaluations are presented.

The results obtained show that H-MAQ provides fairness that is comparable to RIO at the microflow level. At the aggregate level, H-MAQ networks were shown to be effective in implementing precise quantitative allocation of resources to aggregates on all network links. This was not the case for RIO networks.

Table of Contents

List of Figures

Glossary

AF	Assured Forwarding – a Diffserv service class.
ACK	ACKnowledgement packet – a small TCP packet used to acknowledge receipt of a data packet.
Aggregate	A collection of microflows that come from or are bound for a given client and form part of that client's SLA.
APIC	ATM Port Interface Controller – a network interface chip.
ATM	Asynchronous Transfer Mode – refer Appendix A.
Bursty	Data that is transferred or transmitted in short, uneven spurts.
DRR	Deficit Round Robin – a fair packet scheduling mechanism.
DSCP	Diffserv Code Point – a field in the IP header of a packet that specifies the packet's service level in a Diffserv network.
Edge router	A router that lies on the boundary of a network domain. Edge routers are often responsible for admission control.
EF	Expedited Forwarding – a Diffserv service class.
Elastic Flow	A network flow without hard bandwidth or delay requirements.
FIFO	First In First Out – a queuing / dequeuing policy.
FIPL	Fast IP Lookup – an efficient algorithm for finding a packet's next hop.
Flow	A series of packets travelling between a TCP or UDP socket pair. Equivalent to microflow.
FPGA	Field Programmable Gate Array – an integrated circuit that can be programmed in the field after manufacture.
FPX	Field Programmable port eXtender – an FPGA based port processor on the MSR. Designed primarily to perform IP lookups.
FRED	Fair Random Early Detection – a RED based buffering mechanism.
Full Duplex	A link or port that can carry traffic in both directions simultaneously.

Goodput	The amount of data correctly received by a traffic sink in a specified amount of time. Compare with throughput.
H-MAQ	Hierarchical multi drop-precedence queue state Microflow-Aware Queuing. The proposed buffer management and scheduling mechanism for use in core routers.
IETF	Internet Engineering Task Force – an Internet standards body.
IPv4	The fourth version of IP. This IP version currently predominates in the Internet.
IPv6	A new version of IP that includes an increased address space and the addition of a flow label.
ISP	Internet Service Provider – a company selling network resources.
Jitter	Large variations in the transit time for packets travelling through a network. Also called delay variation.
Microflow	Packets that are identical in terms of the 5-tuple: source address; destination address; source port; destination port and protocol. A flow as seen at the packet level.
MPLS	Multi Protocol Label Switching - a label-switching technology in which packets are appended with locally significant MPLS labels at the MPLS network's ingress.
MSR	Multi-Service Router –a router developed for research at The Applied Research Laboratory, Washington University.
MTU	Maximum Transmission Unit – the largest size packet that a network can transmit without fragmentation.
North Bridge	A computer chip that implements the CPU interface and memory interface. Compare with South Bridge.
NTP	Network Time Protocol –a mechanism for synchronising computers.
NTPQ	Network Time Protocol Query – a program that queries NTP servers about their current state.
OSI	Open Systems Interconnect – a layered model describing protocols for communicating systems
Police rate	The maximum allowable data rate as enforced by a policing system.
PVC	Permanent Virtual Circuit – refer Appendix A – ATM
QSDRR	Queue State Deficit Round Robin - a scheduling mechanism and drop policy.

RED	Random Early Detection – a microflow unaware active buffer management mechanism.
RSVP	Resource reSerVation setup Protocol – a protocol for reserving network resources.
RFC	Request For Comments – an IETF standards track document.
RIO	Random early detection with In/Out-of-profile - a microflow-unaware active buffer management mechanism with multi drop precedence levels.
SE	Switching Element – a component of an ATM switch fabric.
SLA	Service Level Agreement - an agreement between a user and a network operator that specifies the level of service the user receives.
Soft-State	A network connection that expires if not periodically refreshed.
South Bridge	A computer chip that runs onboard devices such as IDE controllers and sound. Compare with North Bridge.
SPC	Smart Port Card - a multipurpose MSR port processor.
Target rate	The rate at which a traffic source attempts to send data.
TCP	Transmission Control Protocol – a flow-controlled transport layer protocol.
Throughput	The amount of data transferred across a link in a specified time. Compare with goodput.
TSW	Time Sliding Window – a meter of a microflow's data rate.
UDP	User Datagram Protocol – a non-flow-controlled transport layer protocol.
VCI	Virtual Channel Identifier – refer Appendix A – ATM
VOQ	Virtual Output Queue – a queue on the input port of a router. There is a VOQ associated with each destination port.
WUGS	Washington University Gigabit Switch - the switching component of an MSR.

Chapter 1

Introduction

1.1 Background Information

Initially, the Internet was used as a medium for data communications with predominantly non real-time applications such as email, newsgroups and file transfer. Such traffic is termed elastic as it can tolerate a fair delay before reaching the receiver. With the current convergence of data and telecommunications networks, as well as with the increase in the use of multimedia applications, there is an increasing need for the Internet to carry real-time traffic such as voice and video. Such traffic types have stringent requirements in terms of bandwidth, delay and correct delivery. One of the technologies that were designed to enable the Internet to provide adequate service levels to non-elastic traffic was Diffserv. This mechanism provides improved service levels to non-elastic traffic by prioritising such traffic classes. The advent of this technology is considered important in the Internet's development, as it will equip the Internet to carry real-time media.

The Diffserv model was designed to maximise simplicity and scalability. Diffserv involves the use of a state-aware network edge that polices incoming microflows* on an individual basis. The network edge is also responsible for marking the Diffserv Code Point (DSCP) in the IP headers of incoming packets. The value of a packet's DSCP determines the level of service that it should receive in the network core. The network elements in the core are assumed to be unaware of individual microflows. These elements simply use DSCPs to determine the level of service that each packet should receive. Such microflow-unaware mechanisms are said to be desirable, as they are computationally simple.

The Diffserv model maximises network scalability by pushing the complexity to the edge of the network. The rationale for this is that because data rates are highest in the network core, the data path in core network elements should be as simple and efficient as possible.

One of the service classes offered by Diffserv is the Assured Forwarding (AF) class. This class has multiple drop precedence levels. This is intended to maximise utilization by allowing traffic sources to inject additional traffic into the network at the risk of a higher proportion of packets being dropped. The mechanism works as follows: Should a source exceed its negotiated sending rate, all offending packets are remarked with a higher drop precedence level at the edge of the Diffserv network. Should congestion occur in the core of the Diffserv network, the core network elements will drop the packets with the higher drop precedence levels first. By gambling on there not being any congestion in the network core, traffic sources may receive increased data rates.

It has been recommended that active queuing mechanisms such as RIO (Random early detection with In/Out-of-profile) be used to implement the Diffserv AF class in core routers. Such mechanisms are considered desirable, as

- They are computationally simple,

* The term microflow, as used in this thesis, refers to a correlated series of packets that consist of the five-tuple: source address, destination address, protocol number, source port and destination port. This is the finest granularity with which network elements can identify similarity between packets.

- They limit average queue size,
- They handle multiple drop precedence levels, and
- They provide better fairness than simple drop-tail queues.

1.2 Problem Description

Although the use of microflow-unaware active queuing mechanisms, such as RIO, has been recommended for the Diffserv AF implementation, such mechanisms have been found to have a number of drawbacks. These are listed below.

- There is unfairness between competing microflows as well as competing aggregates* when they differ in terms of:
  - Transport protocol makeup (TCP or UDP),
  - Average round trip time, and
  - Average packet size.

  Because the Internet's traffic makeup is so very diverse, any unfairness relating to traffic type will have very real repercussions.
- There is unfairness between aggregates with different allowable data rates. This is explained as follows: At the edge of the Diffserv network, packets are policed according to their aggregate's Service Level Agreement (SLA). Once in the core of the Diffserv network, identically marked packets from aggregates with higher service levels are treated the same as those from aggregates with lower service levels. This causes unfairness because, for example, excess network capacity is not allocated according to the SLAs of the competing aggregates. This can put the aggregates with higher SLAs at a disadvantage.
- There is unfairness between competing aggregates that have the same SLAs, but differ in terms of the number of microflows that they contain. For example, two companies may have identical SLAs with an Internet Service Provider. These SLAs would specify the company's network connectivity at aggregate level. If one of the companies has more microflows than the other, the company with less microflows will often be disadvantaged

* An aggregate is a collection of microflows that come from or are bound for a given client and form part of that client's SLA.

- The use of mechanisms such as RIO has been shown to result in instability when capacities are high and round trip times are long. This causes jitter, packet losses, under-utilisation and reduced responsiveness.

In addition to the intrinsic unfairness, these factors result in the behaviour of the Diffserv network core not being predictable, but being affected by a number of factors relating to the traffic being carried. This means that although Diffserv does provide relative service differentiation, it is unable to provide quantifiable service levels without being used in conjunction with network over-provisioning. The value of providing relative, rather than quantifiable service levels, is questionable as SLAs are specified in absolute, not relative terms. Furthermore, because the Diffserv core is not aware of competing aggregates, it is not possible to explicitly provision for them. This makes the precise control of Diffserv networks an impossible task.

1.3 Objectives

Recent advances in router technology enable core network elements to provide microflow and aggregate-aware buffer management and scheduling on high-speed network ports. This obviates the need for Diffserv networks to use simple microflow-unaware mechanisms such as RIO in Diffserv network cores. This study explores the degree to which the performance of Diffserv networks implementing the AF class of service may be improved by the use of a microflow, aggregate and DSCP-aware buffer management mechanism in core network elements.

The need for such a buffer management mechanism is motivated and a specification is provided. The specified mechanism, referred to as H-MAQ or Hierarchical multi drop-precedence queue state Microflow-Aware Queuing, was evaluated on a testbed. The results of the evaluations are given. Using these results, this study compares the performance of a Diffserv network where H-MAQ is implemented in the core with that where RIO is implemented. The benefits of using H-MAQ rather than a microflow and aggregate-unaware mechanism are thus demonstrated.

1.4 Scope and Limitations

This study considers the performance of Diffserv networks according to whether the buffer management mechanisms in core network elements are microflow and aggregate-aware or not. A microflow, aggregate and DSCP-aware mechanism is motivated and specified. Thereupon its performance is compared with that of a microflow and aggregate-unaware mechanism. This comparison is made based on a series of comparative evaluations that were performed on a network testbed. The comparisons are made using factors such as fairness and controllability as indices.

Employing H-MAQ instead of RIO would in general require the addition of a number of signalling paths. These would be needed to allocate resources to aggregates on internal links of the network. These signalling paths are, however, not considered in this study. Only the functionality of the network elements with regard to user traffic is considered. Further, only the Diffserv AF class of service is considered.

This study is concerned with IP network functionality only. Little attention is given to lower layer protocols. In particular, no attention is given to the synergies that exist when quality of service-aware protocols such as ATM are used in conjunction with IP.

1.5 Development Plan

The remainder of this document is organised as follows:

- Chapter 2 provides the reader with the necessary background information on a number of topics. It begins with a discussion on Internet protocols. The nature of Internet traffic is investigated next. The chapter continues with a discussion of three mechanisms for implementing improved quality of service. These are over-provisioning, Intserv and Diffserv respectively. The Diffserv AF class of service is examined in greater detail next. A section follows this on the interoperation of quality of service mechanisms to provide end-to-end quality of service.
- Because of the relative complexity of H-MAQ, and the fact that it is run on routers, it is necessary to analyse router architecture in detail. Chapter 3 begins with an overview of the role of the router in IP networks. A description of past, present and future router architectures follows. Thereafter, there is a discussion on

the capabilities of current routers. There is a discussion on router buffer management and scheduling best practice. Finally, the architecture of the gigabit router used in the author's implementation is described in detail.

- Chapter 4 describes the design considerations of H-MAQ. It motivates the use of a microflow and aggregate-aware mechanism by noting the performance limitations of using mechanisms such as RIO to implement the Diffserv AF core. Thereafter, the overall design of H-MAQ is considered. This is followed by a discussion on H-MAQ's feasibility. Finally, the scope of the author's implementation is given.
- Chapter 5 provides a technical specification for the implementation of H-MAQ. This is done in two parts. Firstly, the modifications required on the edge routers are discussed, whereupon the implementation of the H-MAQ core is specified.
- Chapter 6 describes the design and implementation of the testbed used to evaluate H-MAQ and compare its performance with that of RIO. The testbed's topology, hardware implementation, traffic sources, router mechanisms, and data collection mechanism are all described here.
- Chapter 7 describes the evaluations that were performed on the testbed. It also describes the results of the evaluations and gives an analysis of the results.
- Chapter 8 gives the conclusions drawn from the evaluations as applicable to Diffserv AF network performance in general.
- Chapter 9 makes recommendations for future work.
- Finally, two appendices give details on ATM as well as on the development and testing procedure for the author's RIO implementation.

Chapter 2

Literature Review

2.1 Internet Protocols

The following section introduces the Open Systems Interconnect (OSI) model for describing the hierarchical protocol stack of computer networks. Individual Internet protocols are then considered with reference to this model.

2.1.1 The Layered Structure of Internet Protocols

The task of allowing applications that are hosted on separate machines to communicate is a complex one. Because of this inherent complexity, it is useful to use a layered model to describe the process. This approach facilitates the abstraction of detail at many levels. Although the OSI model has been criticised for being too rigid, it will be reviewed here as it provides a context for a description of the Internet's constituent protocols. Figure 1 illustrates the OSI model.

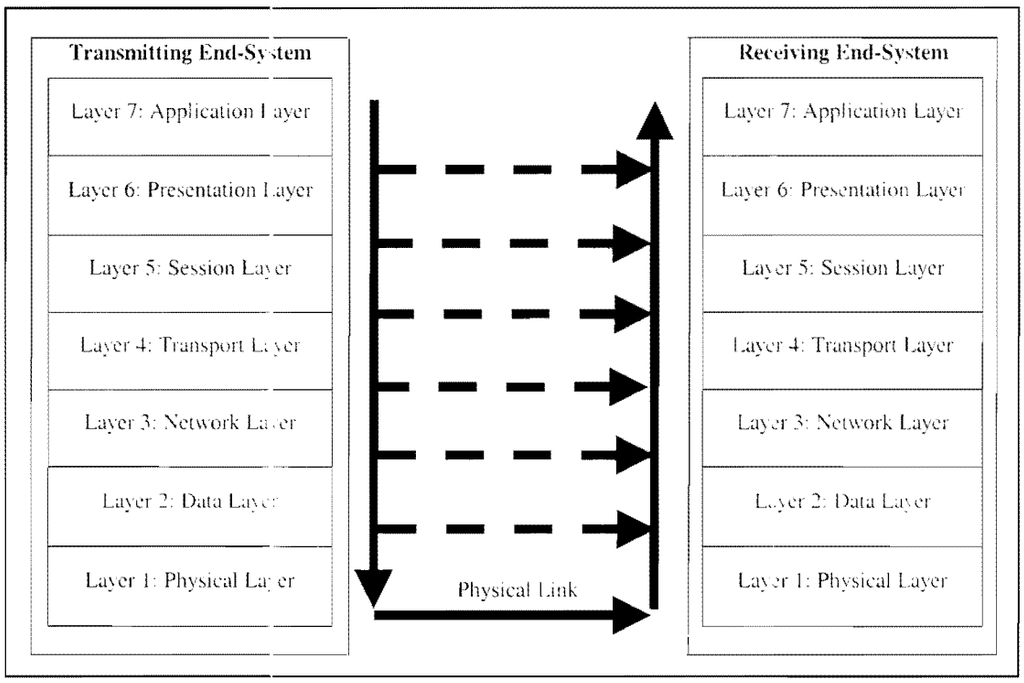

Figure 1: The OSI model

Figure 1 demonstrates that identical layered architectures are duplicated on both the transmitting and the receiving end-systems. The diagram illustrates that when an application on one end system-needs to send data to that on another end-system, the data is initially sent down the stack to the Physical Layer. The data is then sent to the receiving end-system before being translated back up the stack. The following points describe the role of each layer on the OSI stack [1].

- The Application Layer consists of the user applications that need to be connected. This refers to applications such as web browsers and FTP.
- The Presentation Layer translates data from the application's format to a format understood by the network and vice versa.
- The Session Layer is responsible for establishing, managing, and terminating connections between applications
- The Transport Layer provides transparent transfer of data between end systems. This includes error recovery and flow-control.
- The Network Layer addresses the routing and switching issues required to create a logical circuit between the end-systems.
- The Data Layer is concerned with the data being encoded into bits. This layer handles the physical medium as well as frame synchronisation.

- The Physical Layer consists of the physical medium, be it air, fibre or wire.

2.1.2 Network Layer Protocols

The Internet Protocol (IP) operates at the network layer of the OSI protocol stack. When a packet is passed down to the IP Layer, an IP header is appended to the packet. This header contains information relating to the packet such as the source and the destination addresses of the packet. The IPv4 header format is illustrated in Figure 2.

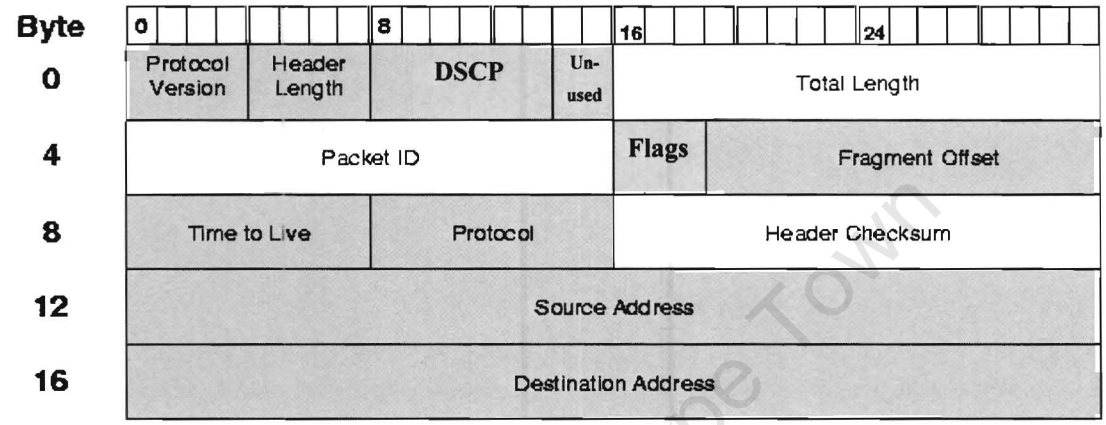

Figure 2: The IPv4 header format (20 bytes total)

A new version of IP, namely IPv6 is currently on the IETF standards track [2]. Two notable differences between IPv4 and IPv6 follow.

- IPv6 has a far larger address space than IPv4.
- The IPv6 packet header includes a flow label. The flow label, together with the source address, may be used by network elements to uniquely identify traffic sources [3]. This is beneficial as it means that higher-layer protocol headers need not be examined at routers for flows to be uniquely identified.

IP provides a datagram service to the protocols above it. By a datagram service, it is meant that the IP service has the following characteristics:

- IP is a connectionless packet delivery service. This means that no IP layer connection set-up procedure occurs before data transmission and that the communicating end-systems don't maintain one another's state.
- IP is unreliable. This means that IP senders have no verification mechanisms to ensure that a transmitted IP packet is ever received.

IP does not include any flow-control mechanisms. These are implemented in the layers above IP.

2.1.3 Transport Layer Protocols

Both Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) are found above the Network Layer. Although the position of these protocols corresponds to the Transport Layer of the OSI Model, only TCP provides error recovery and flow-control as specified by the OSI model. However, because UDP is considered a peer to TCP, it will be included in this section too.

2.1.3.1 Transmission Control Protocol (TCP)

TCP provides an end-to-end connection abstraction. This means that TCP hides transport details such as multiplexing and error recovery from the above layers. When TCP receives data from the layer above it, it splits that data into packets. Each packet gets a TCP header appended to it. The TCP header format is illustrated in Figure 3.

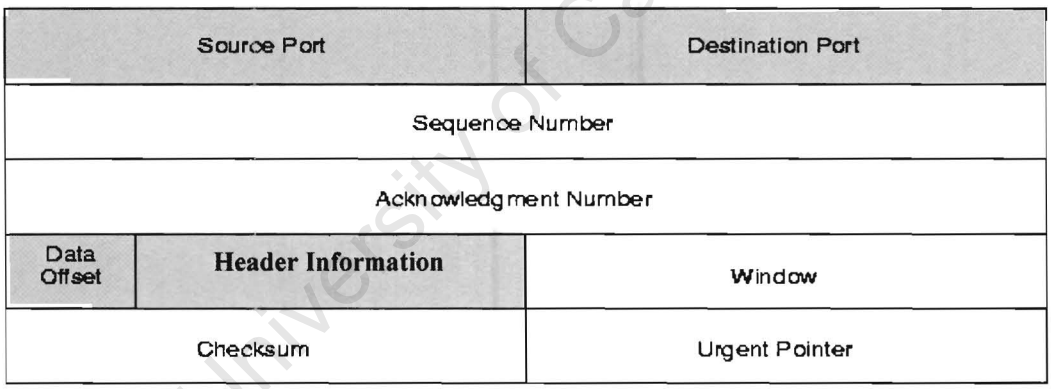

Figure 3: The TCP header format (20 bytes total)

Data packets are transmitted using connection orientated data transfers. A connection must thus be negotiated between two communicating parties before data may be sent.

TCP is said to provide guaranteed delivery. This means that should a TCP packet be lost in the network, the TCP sender will re-send it until the packet is delivered. This is achieved as follows: Before transmission, a TCP sender saves a local copy of all packets. It then transmits them using the unreliable IP datagram service described in Section 2.1.2. If a packet is lost or corrupted in the network, the transmitter re-sends

it. If the receiver acknowledges that it has got a packet, the transmitter may delete its local copy.

Because TCP provides an in-order service, if a packet is lost in a network, no subsequent packets may be delivered to the above layers of the receiver until the lost packet has been retransmitted by the sender and received by the receiver.

2.1.3.2 TCP Flow-control

In a scenario where data is being transferred from a sender to a receiver, there is two-way communication between the two end points. The sender transmits data packets to the receiver. The receiver then signifies receipt of the data packets by sending acknowledgement packets (ACKs) back to the sender.

TCP senders use a congestion window to determine the maximum number of packets that may be in the network at any given time. Whenever a TCP sender receives an acknowledgement it knows the acknowledged packet has been received and is no longer in transit on the network. It is thus safe for the sender to transmit another packet. In a scenario where there are few senders and the network is not congested, senders may safely be allowed a large congestion window. However if the network is approaching congestion, the senders should reduce their congestion windows so that the total number of packets in the network is reduced. TCP senders receive feedback about the amount of congestion in the network by monitoring dropped packets.

TCP's sliding window mechanism will now be considered in detail. When a connection is first established, the congestion window for that connection is set to one packet. TCP then enters an initial phase of multiplicative congestion window increase. First, the one packet is transmitted. Upon the receipt of that packet's ACK, the congestion window is doubled. Two packets are transmitted. Upon receipt of these two ACKs the congestion window is again doubled. This multiplicative increase continues until a packet is lost in the network. When a packet is lost, the receiver indicates this by sending duplicate ACKs to the source upon the receipt of each packet subsequent to the lost packet. When the source receives these, it deduces which packet was lost and re-transmits the lost packet. Because the loss of a packet indicates that there is congestion in the network, the source now halves its congestion window.

From this point on, the source goes into congestion avoidance mode. Here the TCP window size increases additively until another packet is lost. When this happens, the window size is again halved. In the case of multiple packet losses, it is possible that a source will not receive any ACKs. If this happens, the source is forced to wait for a timeout to occur before it can resend the packets. During this wait, the congestion window size is reduced to one packet. This occurrence is very damaging to TCP throughput. Figure 4 shows a possible TCP congestion window profile over time.

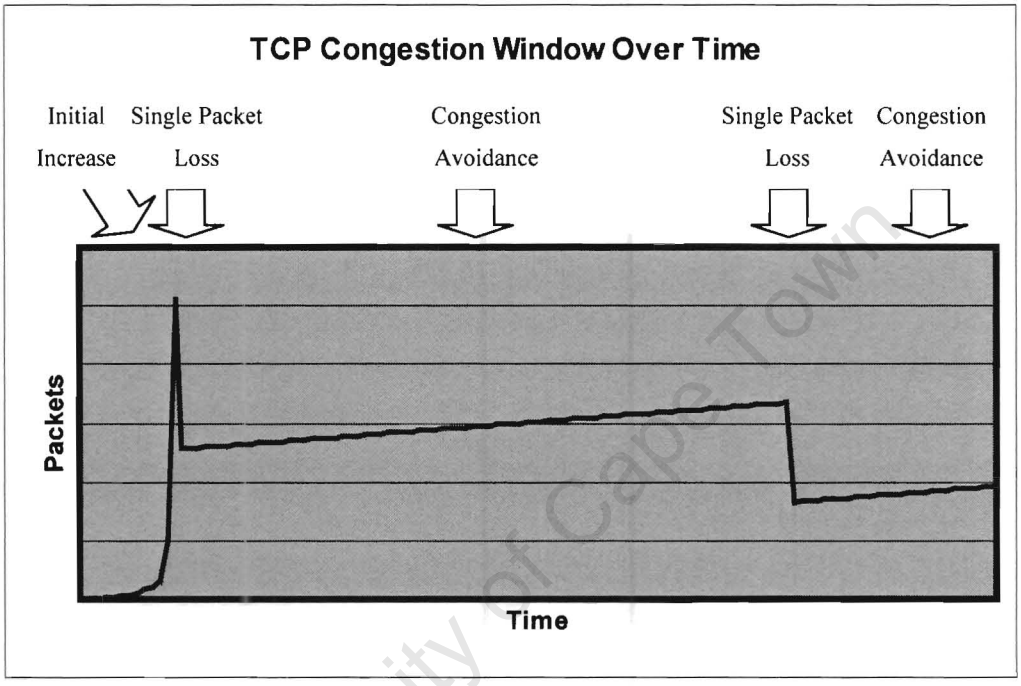

Figure 4: An example TCP congestion window profile over time

This demonstrates that with the exception of the start of a new connection, TCP uses a linear increase and an exponential decrease when manipulating its congestion window. The exponential decrease is necessary because congestion grows exponentially in networks. Figure 4 also demonstrates that TCP congestion control results in bursty traffic profiles.

The increase in TCP sending rates is affected by the connection's round trip time. This is so because TCP sources need to wait for the receipt of ACK packets before sending further data. If the round trip time for a connection is long, the increased delay in receiving the ACKS results in the connection ramping up its sending rates more slowly than it would if the connection had a shorter round trip time.

2.1.3.3 User Datagram Protocol (UDP)

By re-transmitting packets that are lost or corrupted in the network, TCP provides a guaranteed service. There are, however, applications for which this service is not desirable. An example of this is real-time voice transfer. When a real-time voice stream is being sent across an IP network, the stream can tolerate the loss of a certain proportion of the packets without significant degradation of sound quality. If a network transmitting a voice stream loses a packet, it is preferable for the receiver to compensate for the lost packet rather than wait for it to be re-sent by the source. This is because the time required for the packet to be re-sent is often prohibitively high. One can use UDP as an alternative to TCP for time-dependent applications. UDP is a datagram protocol. As such it provides a connectionless, non-guaranteed service. Figure 5 shows the UDP header.

Source Port	Destination Port
Length	Checksum

Figure 5: The UDP header format (8 bytes total)

It is clear from its header that UDP is a very simple protocol. Should a source need to send data, it splits the data into packets, adds the UDP header and sends the packets down to the IP layer. The UDP receiver does not acknowledge the receipt of packets.

UDP does not include any form of flow-control or congestion avoidance. Users are thus able to send as much data into networks as they wish. Sending more packets into a network than the network can handle does, of course, result in high packet losses.

2.1.4 The Effect of Transport Protocols on Fairness

The previous sections describe how although TCP includes a flow-control mechanism, UDP contains none. This section considers link fairness in the light of transport layer flow-control mechanisms.

In general, routers receive packets and send them out on links. In the case where a given link's available bandwidth is limited, the router should share the link's available bandwidth fairly amongst all contending parties. This is done by dividing the traffic

sources into microflows and sharing the output link bandwidth between them. Microflows are packets that are identical in terms of the 5-tuple: source address; destination address; source port; destination port and protocol. This is the finest granularity at which routers are able to identify similarity between packets. Microflows are equivalent to the transport layer flows mentioned in Section 2.1.3.

A theoretically fair link-sharing mechanism may be implemented on a router by providing a separate queue for each microflow present. Arriving packets are placed on their corresponding queue. A deficit round robin scheduler services the queues. The real life fairness of such a mechanism is, however, strongly affected by the transport protocols that are run on the end stations. The resultant unfairness when traffic sources with different transport layer protocols compete for a resource is described below.

Should a TCP packet be dropped due to queue overflow, that packet's source will subsequently reduce its data rate by at least half. This is due to the transport layer flow-control of TCP. Should a UDP packet be dropped because of queue overflow, the sending rate of the source will be unaffected. This is because UDP has no flow-control. On the contrary, many UDP application sources increase their sending rate when packets are dropped to provide better forward error correction.

In the above example, even though both the TCP and the UDP sources were treated fairly at the packet level, their resultant data rates were far from fair. Transport layer protocols thus make providing fine granularity fairness to microflows a near impossible task. This is due to large differences in how TCP and UDP respond to packet losses and TCP's coarse-grained response to packet losses. The fact that TCP flow-control is affected by round trip times also limits microflow fairness.

2.2 The Nature of Internet Traffic

In order to perform relevant evaluations of buffer management and scheduling mechanisms, it is necessary for testbed conditions to model as closely as possible those experienced in real networks. It is thus necessary to consider the nature of Internet traffic. Finding current data on Internet traffic is a difficult task as few

carriers release such data. Using the data that was found, this section describes the volume and composition of Internet traffic. It also discusses how Internet traffic may change in the future.

When characterising Internet traffic, the following attributes should be considered:

- The volume of traffic and the number of concurrent microflows on core links.
- The application types that source the traffic.
- Transport Layer protocols (TCP or UDP).
- Predominant packet sizes.

Each of these topics will be dealt with individually in the sections below.

2.2.1 Internet Core Traffic Volume

The growth in Internet usage has been described as explosive. Some have even claimed that Internet usage has been doubling every three months. Andrew Odlyzko debunked these claims in late 2000 as being nothing more than a pitch aimed at selling network hardware. Odlyzko stated that growth rates have levelled off at approximately 100% per year [4]. This suggests compliance to Moore's law. Overestimations of the Internet's growth rate resulted in companies increasing their network capacities greatly. The realisation that network capacity far outstripped demand was one of the primary factors leading to the Telecom Crash in 2000. Subsequent to the crash, increases in network capacity have slowed dramatically. TeleGeography, Inc. found that total growth of international link bandwidth had shrunk to less than 40% during 2001 and 2002 [5]. This reduction in the growth of capacity suggests that service providers are waiting for demand to catch up with the supply of bandwidth.

In terms of Internet core link capacity, Internet service providers currently tend to use links of up to 2.4 Gigabits per second (OC-48) [6,7]. Such trunks can carry thousands to hundreds of thousands of concurrent microflows [8,9].

2.2.2 Application Sources of Internet Traffic

Figure 6 presents the application byte composition for selected applications using two samples of Internet traffic, one from the year 1997 and the other from 2001/2. The

1997 data was collected by MCI along one of their trunks [10]. The 2001/2 Data was found at the CAIDA (Cooperative Association for Internet Data) web site [11] and provides a breakdown of the traffic composition at the San Diego Network Access Point over a year beginning on 14 March 2001.

Traffic Type	1997 Byte Volume	2001/2 Byte Volume
HTTP	75%	28%
FastTrack	<1%	9%
SSH	<1%	6%
FTP	2-8%	5%
SQUID	<1%	5%
Shoutcast (real-time)	<1%	3%
IRC	<1%	1%
NNTP	1-4%	<1%
DNS	1-2%	<1%
RealPlayer (real-time)	0.5-2.5%	<1%

Figure 6: Application byte volumes in 1997 and 2001/2

Figure 6 demonstrates a marked change in the application byte-composition between the samples taken in 1997 and in 2001/2. This signifies that the distribution of application traffic found on the Internet is dynamic and reflects the public's shifting favour. A notable observation is that a large number of new applications seem to be displacing existing applications such as HTTP. Also noteworthy is that in 2001/2, the amount of real-time traffic remained modest.

Quantitatively predicting future application traffic compositions is not possible. It is, however, plausible that should the Internet's quality of service improve to the extent where real-time traffic can be adequately carried, there will be a rapid uptake of services such as Voice Over IP and video-conferencing. This will markedly increase the byte volume of real-time traffic applications.

2.2.3 Distribution of Internet Traffic according to Transport Protocol

During 1997, MCI collected traffic data from their trunks [10]. The data collected demonstrated that 95% of the byte composition was TCP traffic and that the remaining 5% was UDP. A report of traffic at the NASA Ames Internet exchange between May 1999 and March 2000 had an identical transport protocol byte composition [12].

In summary, the majority of Internet traffic is TCP and the proportion of TCP to UDP traffic remained constant between the years 1997 and 2000. The transport protocol distribution is a governed by the mix of applications. Should there be an increase in the amount of real-time applications in the future, it is likely that the proportion of UDP traffic will increase. This point is explained in Section 2.1.3.3.

2.2.4 Distribution of Internet Traffic according to Packet Size

During March 1998, data collected at FIX West demonstrated that almost half of the packets were less than 44 bytes. 18% were between 552 and 576 bytes. And almost 18% were 1500 bytes. Although there were many 44-byte packets, they only formed a 5% byte volume whereas the 1500-byte packets made up more than half of the byte volume [13]. A report of traffic at the NASA Ames Internet exchange During February 2000 had a similar packet size distribution [12].

The 44-byte packets may be attributed to ACK packets and small packets such as Telnet packets. Packets of approximately 576 bytes come from TCP stacks that do not implement Maximum Transmission Unit (MTU) discovery and 1500-byte packets come from Ethernets [14].

2.3 Over-Provisioning for Quality of Service

One proposed mechanism for providing quality of service in networks is to simply over-provision a best effort network. In this model, Internet Service Providers (ISPs) ensure that the sustainable throughput of a given link or exchange is greater then the maximum expected demand. Such a network would theoretically never have any congestion and thus provide a high quality of service to all traffic.

This model for providing quality of service does, however, have the following shortcomings.

- It is not possible for networks to be transparent at all times. Equipment failures, inaccurate forecasts and unexpected traffic surges can all result in situations where the available capacity exceeds demand [15]. Under such conditions, there is no mechanism for prioritising non-elastic microflows.
- There is a tendency for ISPs to overbook in an effort to improve network utilization and increase profits. That ISPs will overbook by factors of 5 to 10 is to be expected [15]. The extent of overbooking can be expected to increase in future. This is because the global growth in Internet traffic is currently far higher than the growth in network capacity. This is quantified in Section 2.2.1.
- The model of best effort with over-provisioning is an inefficient use of network resources. This is so because all traffic, whether it is elastic or not is treated identically. By rather prioritising the non-elastic traffic, it is possible to realise acceptable network performance for all traffic with far greater utilization of network capacity.

For the reasons stated above, it can be argued that over-provisioning does not provide a desirable solution to the quality of service problem.

2.4 Intserv for Quality of Service

One of the first mechanisms proposed for providing tiered quality of service in IP networks was Intserv. The Internet Engineering Task Force (IETF) Intserv Working Group specified Intserv [16].

2.4.1 The Intserv Mechanism

The Intserv model allows users to make end-to-end reservations. This ensures that the network can provide the required service to the users. The mechanism works hand-in-hand with ReSource reserVation setup Protocol (RSVP) [17]. Should an application require a given end-to-end service, it must first make a reservation using RSVP. In RSVP, a path message is sent from the source to the recipient. The path message primes each Intserv router along the path to expect a reservation message. Upon receipt of a path message, the end-point sends a reservation message back to the

source along the same path. As the message moves upstream, each network element makes a reservation for that particular microflow. If the resources aren't available, the request is denied. Conversely, if the resources are available on all routers along the path, a soft-state connection is created and data may be sent.

Intserv is not considered to be an adequate solution for providing Internet scale quality of service, as there are scalability concerns. Should Intserv be implemented in the Internet, core routers would be required to perform RSVP resource allocation, to store per-microflow state and to perform per-microflow queuing for a large number of microflows. Providing this functionality for a large number of microflows is considered excessively demanding in terms of computational resources.

2.4.2 Intserv Classes of Service

There are two main service classes that are provided by Intserv networks. These are Guaranteed Service and Controlled Load Service. The Guaranteed Service class is suitable for applications with strict real-time delivery requirements. This class of service provides an assured level of bandwidth and a firm maximum end-to-end delay guarantee. The Controlled Load Service class provides no firm quantitative guarantees. The service provided by the network is equivalent to that experienced by a best effort microflow on a lightly loaded network. This service is suitable for applications that can tolerate a small amount of loss and delay.

2.5 Diffserv for Quality of Service

The IETF Diffserv Working Group (concluded in March 2003), specified Diffserv as a mechanism for providing tiered quality of service in IP networks [18].

2.5.1 The Diffserv Mechanism

Diffserv was proposed as a solution to Intserv's scalability problem. Diffserv solves this problem by introducing traffic aggregation. The mechanism works as follows: If the packets of a given microflow are destined for a Diffserv network, then the packets are classified and marked before entering the network. This can happen at the traffic source or at one of the Diffserv network's edge routers. The packets are marked according to the class of service that they should receive. This information is placed

in the packet's Diffserv Code Point (DSCP), which is stored in the IP header. The six bit DSCP, together with two currently unused bits, replaces the eight bit IPv4 Type of Service field [19]. Refer Figure 2 for an illustration of the IP header. Core routers in Diffserv networks only have to consider the DSCP of a packet to decide what service that packet should receive. A router's response to a DSCP is known as its "per hop behaviour." Employing the mechanism described above, relative differential services may be provided. In order to further improve the performance of Diffserv, it is necessary for policing to be performed at the edges of the Diffserv network. By limiting the traffic entering a Diffserv network and providing class-aware scheduling in the core, it is possible to provide bounded relative quality of service across Diffserv networks.

Should a client wish to use the services of a Diffserv network, it negotiates a Service Level Agreement (SLA) with that network. It may then begin sending data into the network. At the border of the network, the traffic coming from that client is policed according to its SLA. This can involve dropping offending packets, or marking them so that they are the first to be dropped in the event of network congestion.

The Diffserv model assumes that aggregate and microflow-aware policing and buffer management are possible on the edge routers of the network but not in the core. This is based on the assumption that the traffic volume in the core is prohibitively high. Because aggregate and microflow-unaware buffer management and scheduling is used in the core, the extent to which the network engineer can control and monitor the network's performance is severely limited. The network's core behaves, in many respects, like a black box. As a result of this, for adequate levels of service to be achieved, it is necessary for an appreciable amount of over-provisioning to take place. Refer Section 2.3 for comments relating to this practice.

2.5.2 Diffserv Classes of Service

The IETF defines the Expedited Forwarding (EF) and the Assured Forwarding (AF) classes of service. EF provides a virtual leased line service. It was designed for traffic that requires strict bounds on delay as well as guaranteed bandwidth. EF packets that exceed their negotiated rate are dropped at the ingress to the Diffserv network. By

contrast, AF was designed to provide a service similar to that which a lightly loaded network provides. It is intended for traffic types that require guaranteed bandwidth with less stringent delay requirements. AF packets that exceed their negotiated rate are re-marked to a higher drop precedence level at the Diffserv ingress.

Whereas EF is allocated a single DSCP, the AF class comprises four independent sub-classes of service, each containing three drop precedence levels [20]. AF thus has a total of 12 DSCP values allocated to it. The four sub-classes are intended to provide tiered levels of service within AF. The three drop precedence levels are intended for situations where traffic sources wish to inject additional traffic into the network at the risk of a higher number of packets being dropped. This mechanism works as follows: Should a source exceed its negotiated sending rate, all offending packets are re-marked with a higher drop precedence level when they enter the Diffserv network. Traffic sources may thus trade off throughput with packet losses. Networks may aggregate the three drop precedence levels into two [20]. It has been suggested that the use of 12 sub-classes and drop precedence levels in AF has not been properly motivated and that a smaller number would have sufficed [21].

The use of strict priority queuing is recommended for EF packets in the core of Diffserv networks, as EF traffic should not be affected by traffic of lower service classes [22]. Priority queuing ensures that no non-EF packet is sent whilst an EF packet is waiting to be sent. The use of priority queuing for EF packets means that EF is an extremely expensive service. This is because EF requires a substantial reservation of network resources and EF traffic can excessively disadvantage traffic classes with lower priorities [23]. It also results in an increase in the burstiness and jitter of EF traffic profiles. This is especially problematic on long paths [24].

Because this study concerns the AF traffic class, AF architectures will be considered in greater detail in the next section.

2.6 Diffserv AF Architectures

This section describes the commonly accepted way of implementing AF and then some of the problems surrounding this implementation. Finally, some of the proposed workarounds to these problems are considered.

2.6.1 The Diffserv AF Mechanism

This description of the accepted mechanism for implementing AF was compiled using the relevant IETF RFC's as well as technical reports on experiments evaluating which mechanisms are preferable. In Section 2.5.1, it was explained that the behaviour of Diffserv edge routers is very different from that of routers in the core of Diffserv networks. As a result, edge and core router behaviour and architecture will be dealt with in separate sections.

2.6.1.1 Diffserv AF Edge Router Behaviour and Architecture

As described in Section 2.5.1, it is desirable for edge routers on Diffserv networks to implement a policing and packet tagging mechanism. This ensures that the amount and type of traffic entering the network does not exceed that which the network can handle. The traffic may also be monitored for billing purposes at the edge of the Diffserv network.

In the scenario where a Diffserv network is to provide an AF connectivity service to an aggregate client such as a business, the client is required to set up a data link leading to an edge router of the Diffserv domain. Further, a service level agreement must be compiled detailing the amount of data that the client may send on each of the AF sub-classes. The client is required to mark the DSCP of packets within its network to indicate which packets should receive what AF service. The router at the edge of the AF network is required to police, re-mark and possibly drop the packets before sending them into the network core. This is necessary as the client network may accidentally or maliciously send packets in excess of its service level agreement.

In the scenario where a client's service level agreement allows it to send X Kbps of AF1 traffic into a Diffserv network, it marks packets of that aggregate as AF1 and sends them to the edge router. The edge router uses a profile meter called the Time

Sliding Window (TSW) to monitor the incoming traffic. If the client uses less than its allowable data rate, all packets are marked at the lowest drop precedence level. If the client sends traffic at a rate higher than X Kbps, the policing algorithm marks the excess packets to a higher drop precedence level. A microflow-aware policing mechanism is considered desirable as it helps to ensure that should a client exceed its service level agreement, all users from that client receive a similarly reduced service [25].

2.6.1.2 Diffserv AF Core Router Behaviour and Architecture

In this section an overview of the architecture of core routers implementing AF is given, followed by a description of some active random drop algorithms.

2.6.1.2.a Diffserv AF Core Router Architecture Overview

Active random droppers such as Random Early Detection (RED) are required for AF to minimise long-term congestion [20,22]. Although no recommendations are made about which specific random dropper mechanisms should be implemented, Random Early Detection with In/Out-of-profile (RIO) has been found to work better than other microflow-unaware active queuing mechanisms such as Weighted RED [26]. RIO is thus a popular choice for AF networks.

Figure 7 shows a router architecture that may be used to achieve the service specified for AF by RFC 2597 [27].

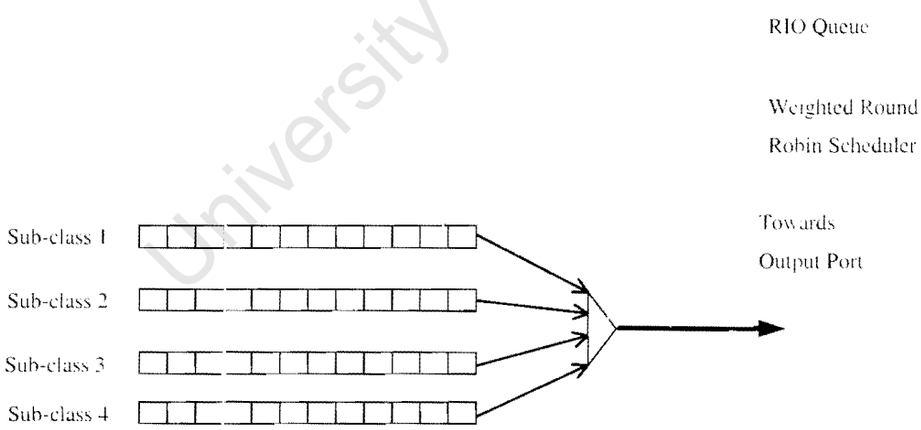

Figure 7: Router architecture with four RIO queues

The AF core router's buffer management and scheduling architecture consists of four RIO queues that lead to a weighted round robin scheduler. There are four queues as each of the four AF sub-classes is assigned its own RIO queue. This provides isolation between the AF sub-classes. The output of the scheduler leads to the output port with the possible inclusion of, for example, a priority scheduler on the data path if EF is implemented on the same network.

Because in-order delivery is specified for microflows of the same AF sub-class [27], packets from the same sub-class with differing drop precedence levels share the same RIO queue. The probability of arriving packets being enqueued versus being dropped, however, depends on the drop priority of the packet. Note that although the IP DSCP makes provision for three drop precedence levels within each AF sub-class, AF compliant networks need only implement two drop precedence levels [27].

RED was first proposed in 1993 by S. Floyd and V. Jacobson [28]. RIO was proposed in 1998 [25]. RIO is similar to RED except that it includes provision for multiple drop precedence levels. RED will be described next. This will enable an easy understanding of RIO, which will be described in the subsequent section.

2.6.1.2.b Random Early Detection (RED)

The RED mechanism uses a single First In First Out (FIFO) queue. The buffer management algorithm stores the average buffer occupancy (*avg*) of this queue, as well as minimum and maximum thresholds for *avg*. These are called *Thresh* and *MaxThresh* respectively. It also stores a variable called *Linterm*, which determines the probability of a packet being dropped when it is between *Thresh* and *MaxThresh*. Precisely how the above variables determine whether an arriving packet will be dropped or enqueued is shown in Figure 8.

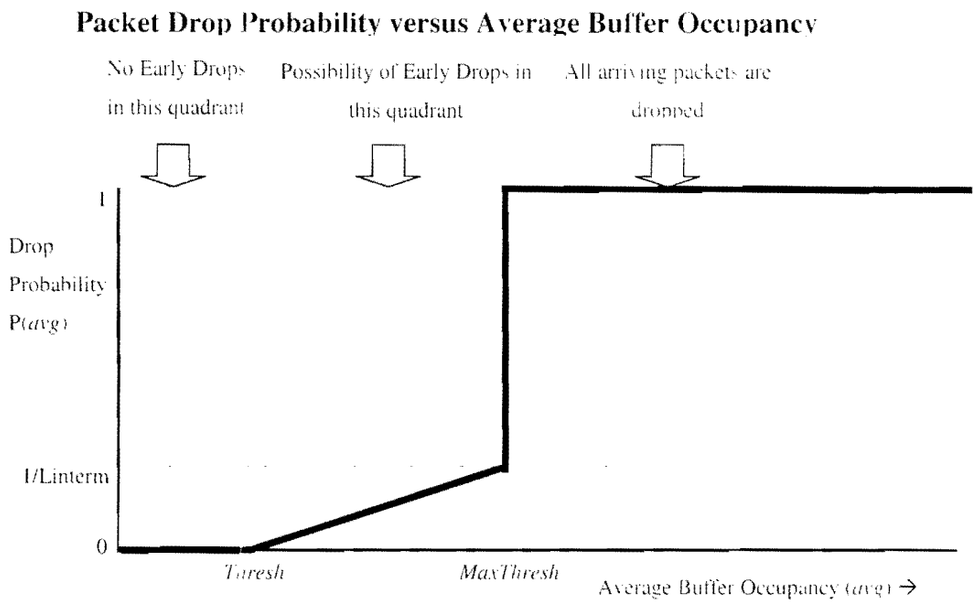

Figure 8: The probability of RED dropping an incoming packet as a function of average buffer occupancy

When a packet arrives, if *avg* is below *Thresh*, the packet is enqueued. If *avg* is above *MaxThresh*, the packet in dropped. If *avg* is between *Thresh* and *MaxThresh*, the packet is dropped with a probability P(*avg*) which increases as *avg* moves from the *Thresh* to MaxThresh. By the appropriate tuning of *Thresh*, *MaxThresh* and *Linterm*, it is possible to specify the desired average buffer occupancy level.

The setting of *Thresh* determines the level of buffer occupancy at which the gateway starts requesting reduced sending rates from senders. Setting this value too high results in bursty traffic sources being penalised. Setting it too low results in reduced link utilization [28]. *Linterm* is used to determine the rate at which the probability of dropping a packet increases as *avg* exceeds *Thresh*. It is recommended that the gradient of the P(*avg*) not be too great otherwise excessive oscillation results [28].

The average queue size (*avg*) is determined by an exponentially weighted moving average. The following equation defines the mechanism:

$$\text{New } avg = (1 - q\_weight)avg + q\_weight \cdot current\_buffer\_occupancy$$

As can be seen, *q_weight* may be used to alter the rate at which *avg* responds to changes in the queue length. By increasing *q_weight*, *avg* is made more responsive to current buffer occupancy levels.

When *avg* is between *thresh* and *MaxThresh*, the function that determines whether a packet should be dropped should perform drops according to a uniform rather than a geometric random function. This minimises the probability of many packets being dropped consecutively.

Packets are dequeued from RED queues in the same way that normal drop-tail FIFO queues are dequeued.

The benefits of using RED buffer management rather than a simple drop-tail queue are as follows [28].

- Through the astute selection of parameters within the router, notably the maximum and minimum thresholds, it is possible to specify the desired average buffer occupancy.
- When packet losses do occur, they tend to occur in isolation rather than consecutively, as often happens with normal drop-tail FIFO queuing. As a result, TCP flows are better able to recover from packet losses.
- Fairness is improved without the need for per-channel-awareness. This is because the probability of a packet from a given microflow being dropped is roughly proportional to that microflow's buffer occupancy level.
- Bursty traffic is not penalised, as is the case with drop-tail FIFO queuing. This is because using the average queue size to determine drop probability amortizes the effect of short bursts.
- The global synchronisation problem is avoided. Because the probability of a drop occurring increases gradually with *avg*, it is unlikely that many flows will simultaneously suffer packet drops with the result that they all reduce their congestion window thus causing the link to be under-utilized.

2.6.1.2.c Random Early Detection with In/Out-of-profile (RIO)

As stated previously, the RIO mechanism is very similar to RED except that there is provision for multiple drop precedence levels. A packet that is marked as being out-

of-profile at the network edge will be treated differently from an in-profile packet once it reaches in the network core. Whereas RED uses *avg* to determine the average buffer occupancy, RIO uses both *avg_in* and *avg_total*. *avg_in* stores the average number of in-profile packets stored in the buffer. *avg_total* stores the average number of in-profile as well as out-of-profile packets.

RIO also has separate "in" and "out" thresholds for each of the RED thresholds: *Thresh*, *MaxThresh* and *Linterm*. Simulation studies have found that these RIO parameters should be staggered to ensure that in-profile packets aren't appreciably disadvantaged by the presence of out-of-profile packets [26]. Staggered parameters are illustrated Figure 9. The top graph in Figure 9 describes the probability of an arriving out-of-profile packet being discarded when it arrives at a router whilst the bottom graph in Figure 9 describes the probability of an arriving in-profile packet being discarded. Note that *avgIn* and *avgTotal* are used respectively on the X-axes of the two graphs.

Packet Drop Probability for In-Profile Packets and for Out-of-Profile Packets

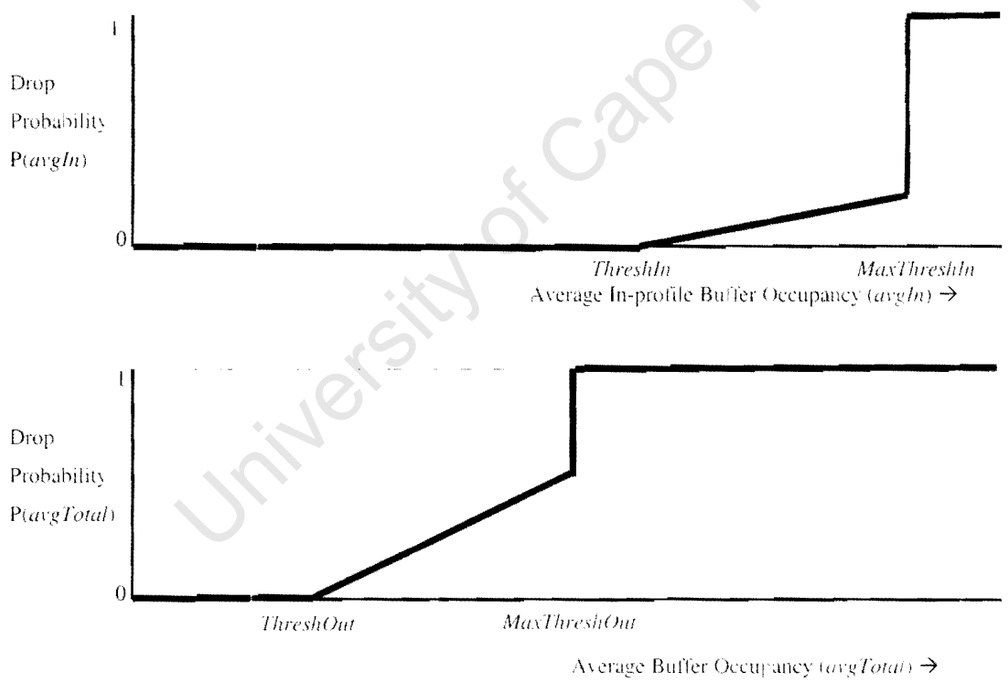

Figure 9: Staggered parameters -A comparison of packet drop probability graphs for in-profile and out-of-profile packets

The parameters for P(*avgIn*) and P(*avgTotal*) in Figure 9 are staggered. This is so because *ThreshIn* is larger than *MaxThreshOut*. In effect this means that before RIO starts dropping in-profile packets, it will be in a state where it drops all arriving out-of-profile packets. This provides greater isolation of in-profile packets from out-of-profile packets.

2.6.2 Performance of the Diffserv AF Mechanism

This section considers the performance of the AF traffic class. It first deals with some of the shortcomings of Diffserv AF and then proposed solutions to those shortcomings are considered.

2.6.2.1 Shortcomings of the Diffserv AF Mechanism

The Diffserv architecture means that core routers need only to consider the DSCP of packets as well as current congestion levels when making buffer management decisions. This results in simple core router implementations. It does, however, introduce a number of problems. The problems, which fall into three categories, are listed below.

1. There is per-microflow unfairness within each AF sub-class [21][3]. In particular, there is unfairness when competing microflows differ in the following areas:
   - Transport protocol: microflows with unresponsive transport layer protocols such as UDP receive higher throughput than microflows with responsive transport layer protocols such as TCP.
   - Round trip times: TCP microflows with longer round trip times are less robust than microflows with shorter round trip times. They thus receive less than their fair share of the available bandwidth. This means that unfairness would result from the following scenario: Two end-users are in contention for a limited resource, but whereas the one user is downloading from a server in the same city, the other user is downloading from a server on the other side of the world. Refer Section 2.1.3.2 for an explanation of this phenomenon.

[3] In the author's opinion, the experiments in this paper were flawed by the fact that the out-of-profile AF packets were placed on the best effort queue. This is contrary to Diffserv specifications. Nonetheless, the author feels that many of the results of this paper remain valid as they correlate with the author's own results.

- Packet size: microflows with larger packet sizes receive higher throughput of data than microflows with smaller packet sizes.

2. There is per-aggregate unfairness when the aggregates, or the typical microflows within them, differ in terms of the following [21,29,30].
   - Transport protocol: aggregates containing predominantly unresponsive transport layer protocols receive higher throughput than aggregates with predominantly responsive transport layer protocols. A client company that has a UDP-heavy traffic profile would thus be advantaged over a company that has predominantly TCP traffic.
   - Round trip times: TCP microflows with longer round trip times are less robust than those with shorter round trip times. Aggregates containing TCP microflows with predominantly longer round trip times thus receive less than their fair share of the available bandwidth. For example, a client company who's data traffic is made up predominantly of TCP flows to a branch at a distant location would be disadvantaged should it be in contention with a company that communicates predominantly with a branch in the same city.
   - Packet size: aggregates containing predominantly larger packet sizes receive higher throughput than aggregates with predominantly smaller packet sizes.
   - Aggregate Service Level: once in the core of the Diffserv network, identically marked packets from aggregates with larger allowable data rates are treated the same as those from aggregates with smaller allowable data rates. This can be unfair to the aggregates with larger allowable data rates. For example, a company with a large contract would be disadvantaged when contending for excess bandwidth against a company with a small contract.
   - Number of microflows: when aggregates containing many TCP microflows compete with aggregates containing fewer TCP microflows, the aggregates containing many microflows often receive more than their fair share of the available bandwidth.
3. Using Random Early Detection or similar mechanisms such as RIO on TCP traffic has been shown to result in instability [31]. This is most problematic in situations where link capacities are high and delays are long. These conditions are exactly what one would expect to find across Diffserv networks in the Internet. The instability can have the following results:

- Increased end-to-end jitter
- Increased packet losses
- Under-utilized network links
- Reduced responsiveness for interactive applications.

The factors mentioned above result in the behaviour of the Diffserv network core not being predictable but depending on a number of factors relating to traffic type. As a result, the AF class cannot, without significant over-provisioning, offer absolute guarantees. It can at best offer relative service differentiation between classes [26,32,33]. The value of providing a relative service to applications is questionable as real applications have requirements that are absolutely quantifiable.

A further implication is that because the Diffserv core is not aggregate-aware, it is not possible to control the allocation of bandwidth to aggregates. This means that providing an adequate service once again necessitates over-provisioning.

2.6.2.2 Workarounds for the Diffserv AF Shortcomings

This section describes some of the mechanisms that have been proposed as an improvement on the classic Diffserv model. These mechanisms were touted as solving one or more of the problems listed in the previous section.

Dynamic Packet State was proposed in 1999 to improve fairness by facilitating communication between core and edge Diffserv routers [34]. This was achieved by placing per-microflow state in the header of each packet that was sent into the Diffserv core. The mechanism is problematic, as it is not compatible with the current IPv4 header. It also incurs a great overhead on the forwarding plane.

Fair Allocation Derivative Estimation (FADE) with feedback control was proposed in 1999 [35]. This mechanism improved fairness by facilitating communication between core and edge Diffserv routers. This was achieved by using dynamic feedback signalling. The mechanism was expanded upon in 2001 with the proposal of an efficient fair allocation estimation technique [23]. The work done on FADE should be considered to be synergistic to this study's proposal because although it proposes the

use of a stateless core its signalling and dynamic bandwidth allocation mechanisms may be integrated into the mechanism presented in this thesis.

In a 2000 proposal [36], it was suggested that edge routers monitor packet losses in the Diffserv network by querying core routers. If the edge routers detect that certain microflows/aggregates aren't responding to packet losses, they throttle them before admitting them into the network. A 2001 paper by A. Habib proposed a similar solution [37].

A 2000 paper suggested the use of Fair RED (FRED) in the core of Diffserv networks [38] FRED keeps state for currently buffered microflows and is thus able to isolate the non-responsive ones. Although this mechanism did improve fairness in simulations, non-responsive microflows still received approximately 40% more than their fair share of the available bandwidth using FRED. Further, a 2002 paper contended that there were no appreciable benefits in using a random drop policy if the buffer management mechanism stores state for backlogged microflows [39].

A 2002 paper reported on simulations aimed at evaluating whether microflow-aware buffer management schemes were able to produce better network performance than microflow-unaware mechanisms such as RED [39]. This paper also evaluated the relative performance of a number of variants of the schemes. It was found that better fairness and microflow isolation may be achieved when using microflow-aware buffer management mechanisms.

A 2002 paper considered the performance consequences of a modification to RED that takes packet size into account when determining whether to enqueued or drop arriving packets. It was found that by performing early drops on large packets with a higher probability than on small packets, it was possible to improve fairness when competing flows differed in terms of their average packet size [40]. A further advantage of this scheme is that ACK packets are less likely to be dropped.

2.7 End-to-End Quality of Service

Previous sections have introduced a number of quality of service mechanisms. It is notable that these mechanisms are not mutually exclusive. On the contrary, an end-to-end connection with a guaranteed service level may involve a number of quality of service mechanisms. Figure 10 illustrates one such scenario.

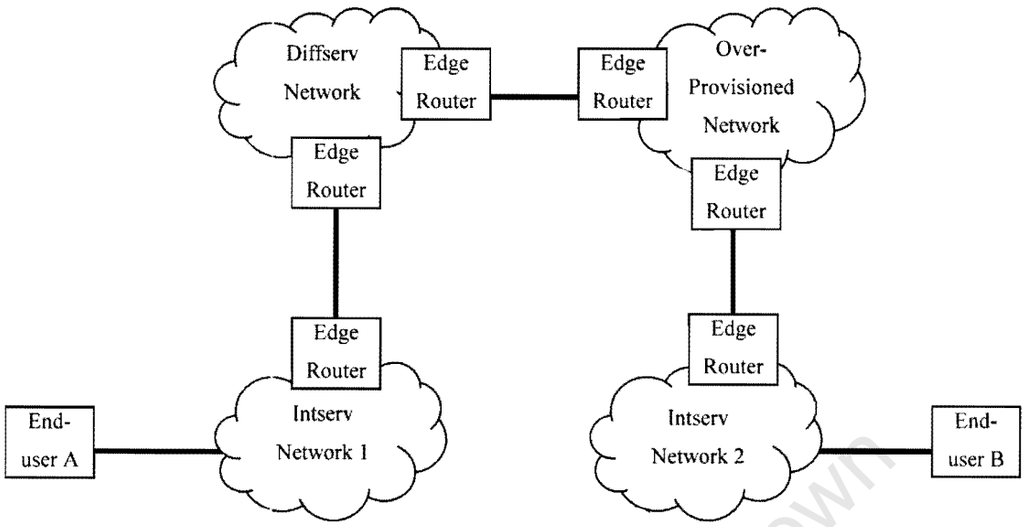

Figure 10: An end-to-end connection with Intserv, Diffserv, and over-provisioned best effort networks

To illustrate the sequence of events needed to establish a connection involving many quality of service mechanisms, we will consider the scenario that End-user A wishes to set up a one-way broadcast to End-user B. This sequence assumes that both end-users are Intserv capable.

- End-user A sends a RSVP path message towards End-user B. This message describes the requirements of the data traffic that it intends to send.
- The RSVP message goes through the routers in Intserv Network 1. This initiates standard RSVP processing.
- The RSPV path message passes transparently through the Diffserv as well as the over-provisioned best effort networks [41].
- The RSVP path message goes through the routers in Intserv Network 2. This initiates standard RSVP processing.
- End-user B receives the RSVP path message and responds by sending an RSVP reservation message.

- The RSVP reservation message is sent back along the path specified by the path message. As it passes through the routers on Intserv Network 2, they perform admission control. If they have the available resources they store the necessary state for a connection. If not, then the request is rejected.
- The RSVP reservation message reaches Intserv Network 2's edge router. If that router deems the necessary resources to be available on the adjacent best effort network then it stores the necessary state for the connection.
- The RSVP reservation message is sent transparently through the over-provisioned best effort network as well as the Diffserv network.
- When the RSVP reservation message reaches the edge router on Intserv Network 1, admission control processing is triggered. It is necessary for this router to establish whether there are sufficient resources available on the virtual link between it and Intserv Network 2. This may involve a comparison between the current service level specification and current reservations. It may also involve a re-negotiation of the service level specification with the Diffserv and best effort networks –although this would not happen with all new RSVP connections. If that router deems the necessary resources to be available, it stores the necessary state for a connection. This could include the appropriate DSCP for the packets.
- The RSVP reservation message goes through the routers of Intserv Network 2. As it passes through, the routers perform admission control. If the resources are available, they store the necessary state for the connection.
- The RSVP reservation message reaches End-user A. This message indicates that the connection has been accepted. It may also specify the DSCP to be used on subsequent packets.

Once a connection has been established, data may begin to flow. The following sequence describes the events as a data packet travels from End-user A to End-user B.

- End-user A sends a data packet. This may or may not have the appropriate DSCP.
- The packet passes through Intserv Network 1 where it follows the path installed by the RSVP path message.
- At Intserv Network 1's border router, the packet's DSCP is marked. The packet is thus subscribed to the appropriate Diffserv service class.

- When the packet reaches the Diffserv network edge router, it is policed. If necessary, it is re-marked to a higher drop precedence level. The packet then passes through the Diffserv network - receiving its corresponding service level.
- The packet passes through the over-provisioned best effort network where it receives the same service as all other packets.
- The packet passes through Intserv Network 2 in the same way it passed through Intserv Network 1.
- End-user B receives the packet.

The above steps demonstrate how it is possible for guaranteed end-to-end connections to be established across networks with disparate quality of service mechanisms. The key factor necessary in such connections is for the various networks to be able to interoperate. For example, when the edge router on Intserv Network 1 was required to assess the availability of resources on the Diffserv network, it had to determine whether the Diffserv network was able to provide then necessary service. Similarly, the Diffserv network had to be aware of the level of service that could be expected from the best effort network. It is in this way that service-level information may be cascaded upstream and end-to-end quality of service is made possible.

Chapter 3

Router Architecture and Operation

3.1 The Role of the Router

The role of the router is primarily to forward IP packets towards their destinations. Further, it is necessary for routers to buffer packets so that bursts and temporary congestion are absorbed. Routers have a number of ports that connect them to other routers or end systems. Routers use routing tables and the destination address stored in the packet header to determine where a packet should be sent next. The next destination of a packet is referred to as its next hop. Because Internet routes and end-systems are always changing, it is necessary that routers frequently update their routing tables. This is done automatically using protocols that enable communication between routers. Common protocols for updating routing tables include Border Gateway Protocol (BGP), Open Shortest Path First (OSPF), Intermediate System to Intermediate System (IS-IS), and Routing Information Protocol (RIP).

3.2 The History of Router Architectures

The evolution of router architectures has been spurred on by ever increasing data rate requirements. This section describes how router architectures have evolved [8].

The first generation of routers consisted of general-purpose single processor computers with multiple network cards. The network cards were connected to the CPU by a shared bus. Arriving packets were forwarded to the CPU. The CPU determined which output port they should be sent to and send them to the appropriate network card. The data path thus required packets to traverse the shared bus twice. In addition to forwarding packets, the CPU was required to perform routing table updates as well as implement control protocols.

The second generation of routers addressed both the CPU and the shared bus bottlenecks by introducing the multiple processor – shared bus architecture. With this architecture, more of the forwarding computation was performed on the incoming port cards. Arriving packets were thus usually sent directly from the input port to the output port. The mechanism worked as follows: The first arriving packet of a flow was sent to the CPU. The CPU determined the output port for the packet. It sent the packet to this port as well as sending a cache entry to the input port. When further packets belonging to this flow arrived at the incoming port, the card was able to use its cache entry to send the packet directly to the output port. This mechanism reduced the load on the shared bus as well as on the CPU. Further enhancements to this architecture involved copying the entire routing table onto each port card. This meant that the initial packet of a connection no longer had to be sent to the CPU.

The third generation of routers had multiple processors on the port cards as before, but used space switching instead of the shared bus. This design alleviated the shared bus bottleneck of second-generation routers by introducing a high speed switching fabric that connected the port cards. This allowed data rates to be increased by a number of orders of magnitude.

The next bottleneck in throughput was packet processing at the port cards. This problem was alleviated with the shared parallel processors – space switching architecture. This optimisation was made possible by the observation that it is unusual for all port cards to be backlogged at the same time. The sharing of forwarding engines was made possible by separating them from the port cards. This architecture is illustrated in Figure 11.

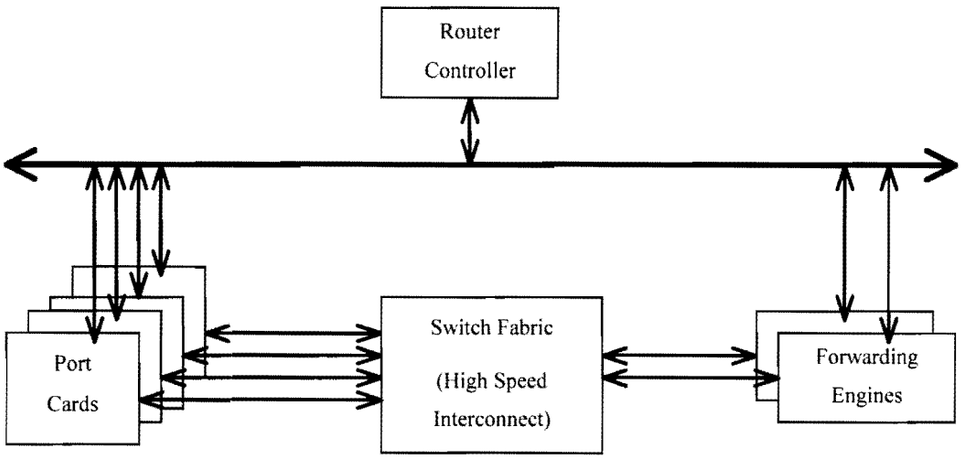

Figure 11: The shared parallel processors – space switching router architecture

The architecture illustrated in Figure 11 worked as follows: When a packet arrived at a port card, its header was sent over the switch fabric to one of many dedicated forwarding engines. The forwarding engine determined the outgoing port and informed the incoming port of this. The packet was then sent to the appropriate output port. Because only packet headers were sent to the forwarding engine, the overhead of sending payloads across the switch fabric was minimised. This architecture made it possible to increase the port density of routers.

The previous architecture was motivated by the assumption that port cards are not able to perform destination lookups at the rate of packet arrivals during a traffic burst. However port processor chips that are able to perform lookups at line speeds of up to 160 Gigabits per second have subsequently been released [42].

3.3 Present and Future Router Architectures

Most modern high-performance routers as well as many routers that are currently in development use a switch fabric architecture with distributed buffer memory and forwarding capabilities [43]. This architecture has a centralised route processor that duplicates its routing table in each of the port cards. The port cards, which are full duplex, each have local processors, input queues, Virtual Output Queues (VOQs), internal buffers, as well as output queues. The input queue stores newly arrived packets. The VOQs store packets whose IP lookup has been performed and are waiting to be sent to the appropriate outgoing port via the switch fabric. The internal buffer stores packets that have been enqueued by an internal buffer management

mechanism. The output queue stores packets that are waiting to be sent to their next hop. A switch fabric connects the port cards to each other as well as to the route processor. Figure 12 shows the data path of a typical high-performance router with three ports.

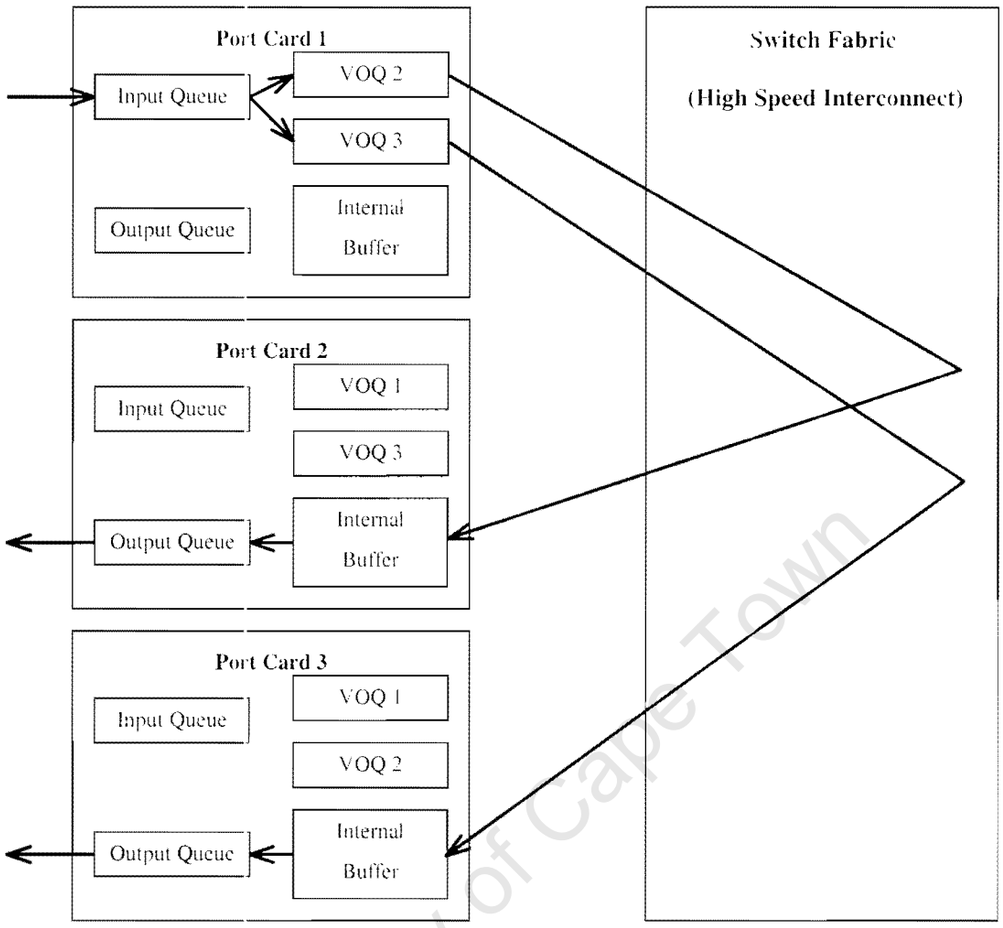

Figure 12: The data path of a router with a switch fabric architecture and distributed buffer memory and forwarding

The following sequence of events occurs when a packet from an external source arrives at Port Card 1:

- The packet is placed on Port Card 1's input queue. The contents of the packet header are read. The port card performs a routing lookup based on the packet header's destination address and determines the packet's next hop.
- The packet is sent to the VOQ corresponding to its destination port.
- The packet is sent through the switch fabric to the appropriate port card's internal buffer. The port cards' internal buffers are the main packet buffers for the router.

It is here that temporary congestion is absorbed and output link sharing is performed. Buffer management and scheduling schemes such as RIO and DRR are implemented here.

- The packet is sent to the output queue whereupon it is sent to its next hop.

The following paragraphs motivate and elaborate on the above data path.

The input buffer's role is to store packets from traffic bursts should the lookup engine become backlogged. However, it is not desirable that packets be stored in this buffer for too long, as the prioritisation of delay sensitive traffic is not possible here due to the packets' headers having not yet been analysed. A further drawback of having long queues here is head-of-line blocking. This will be dealt with in the following paragraph. Because modern port cards are now able to perform packet lookups at line speeds [44], there is no reason why more than one packet should be stored in this buffer at any given time.

After the next hop lookup the packets are sent to one of the port card's VOQs where they wait for the switch fabric to send them to the receiving port card. There is a separate VOQ corresponding to each destination port card. This prevents head-of-line blocking. Head-of-line blocking can occur in a scenario where one output port is backlogged and can't accept any more packets, but another output port is idle. If a single queue were used for packets destined for the switch fabric and the packet at the front of the queue was destined for the backlogged port, it would be necessary to wait for that packet to be dequeued before any packets could be sent to the idle port. Because the input port processor would be blocked from sending packets to the idle port, it would be wasteful of network resources.

As stated previously, the internal buffer of the sending port card is where most of the router's queuing occurs. It is here where quality of service policies are implemented and packet marking or dropping should occur. The reason that the majority of the buffering must be performed on the output port rather than the input port is that all packets here are competing for the same resource, namely the output link.

The output queue must be kept short as, because it is a FIFO, it has no quality of service capabilities. This queue's primary role is to provide a simple interface for the transmission hardware to dequeue.

3.4 Scalability and Microflow-awareness

Although router capacity has increased by a factor of about ten in the past five years [45], the similarly rapid growth of the Internet means that the ability of Internet core routers to process data sufficiently quickly is still of critical concern. This section addresses the ability of routers to scale up to high data rates. The issue of scalability is particularly relevant to this study as H-MAQ is demanding in terms of computational resources. It is thus necessary to consider its viability in this regard. The issue of scalability was touched on previously when considering the relative merits of the Diffserv and Intserv models. There are two aspects to this topic, namely the amount of data or packets that routers can process in a given time, as well as the complexity of operations that the routers are able to perform on the packets given these time constraints. Some examples of complex operations are prioritising packets based on their DSCPs, performing microflow lookups, and performing per-microflow fair queuing. This section considers the performance bottlenecks of modern routers as well as the state of the art technologies that define the maximum capabilities on these bottlenecks. Note that the issue of routing table scalability will not be considered, as it does not relate to this study.

In Section 3.3 it was shown that user traffic passes through two types of elements in modern routers. These are the switch fabric and the port processors. Both of these elements will be dealt with in turn.

3.4.1 Limitations in Switch Fabric Technology

Modern switch fabrics are able to switch data at rates in the order of Terabits per second [46]. Because the switch fabric's role is simply to forward packets, no complex operations need be considered.

3.4.2 Limitations in Port Processor Technology

The second element that will be dealt with is the port processor. The basic operations that are performed here are next hop lookups as well as buffering. Current port processors are able to perform these functions as well as providing service differentiation based on the DSCP of packets at line speeds of 160 Gigabits per second [47].

A more challenging service offering is to provide all of the functions mentioned above as well as to provide per-microflow lookups and queuing for packets in the internal buffers. Providing such a service poses the following challenges.

- The processor needs to perform microflow lookups for all packets before they may be enqueued by the internal buffer.
- The processor needs to be capable of supporting separate queues for all microflows.

These challenges are dealt with separately in the sections below.

3.4.2.1 Performing Microflow Lookups

It has been verified using Verilog that, using currently available hardware components, it is possible to perform per-microflow lookups and queuing on port processors at line speeds of 10 Gigabits per second [6]. Furthermore, the company EZchip has, since October 2001, been producing port processors that are capable of performing per-microflow lookups and buffering at line speeds of 10 Gigabits per second [44]. As described in Section 2.1.2, the adoption of IPv6 will make future microflow lookups easier due to the introduction of the IP flow label.

In Section 2.2.1 it was stated that Internet service providers currently use links of up to 2.4 Gigabits per second. As described above, currently available routers are able to perform microflow-aware packet lookups and buffering at above this rate.

3.4.2.2 Supporting Separate Queues for all Microflows on Port Processors

The number of active microflows generally increases with increased data rates. There is thus a concern that Internet core routers will be unable to store state for all active microflows. This section demonstrates that this is not the case by first discussing the maximum number of microflow queues that core routers would need to support, and

then demonstrating that current routers are able to efficiently support this number of queues.

In Section 2.2.1, it was stated that Internet core links carry thousands to hundreds of thousands of concurrent flows. Microflow-aware core routers do not, however, need to have separate queues for this many microflows. For a router to implement per-microflow queuing, it is not necessary to have a separate queue for all active microflows, but only for backlogged microflows. Because of the bursty nature of TCP traffic, the number of backlogged microflows at any given time is far less than the number of microflows constituting active connections across a given link [8]. In addition to this, the number of microflows for which a router must store state is limited by its internal buffer size. Consider the case of a router buffering for a maximum of 100 ms on a 2.4 Gigabit per second output link. If the mean packet size is 500 bytes, the worst-case scenario in which each buffered packet belongs to a different microflow requires 60 000 separate queues.

The above points give a ceiling to the number of separate queues that port processors need to support. Because River Delta Networks was producing port modules that can support 256 000 queues before 2001 [48], it is clear that this perceived limitation is of little concern. Furthermore, continuing technological advances will increase the number of queues that routers can support.

3.5 Best Practices for Per-Microflow Queuing

This section lists some best practice techniques to be implemented on routers employing per-microflow queuing. These techniques are recommended based on the results of simulation experiments that compared fair queuing mechanisms. The relevance of these practices is that they are all employed in H-MAQ.

- When packets need to be dropped from buffers, they should be dropped from the front of the queues that they are in. This policy, known as drop-from-front, is beneficial as it provides TCP with an early indication of network congestion. It has also been shown to prevent timeouts from occurring on TCP traffic sources by

encouraging fast retransmit [39]. Furthermore, it reduces queuing delay when packet-drop levels are high.

- Using fair schedulers does not alone provide adequate fairness. Fair schedulers should be used in conjunction with appropriate drop policies [39].
- If per-microflow queuing is used, there is little value in persisting to use global random drop mechanisms. Using per-queue drop policies such as longest queue drop helps to provide near perfect microflow isolation [39].
- The Queue State Deficit Round Robin (QSDRR) scheduling and drop mechanism has been found to provide better fairness than conventional DRR [49]. QSDRR is similar to DRR except that whereas when DRR is used, a packet is always dropped from the longest queue, with QSDRR, once a drop queue has been selected, that queue remains the drop queue until there is no queue that is shorter than the drop queue. The QSDRR drop policy is also more computationally efficient because when determining the next drop queue in QSDRR, it is usually not necessary to examine the queue lengths of all microflow queues.

3.6 A High-performance Router Architecture Examined

This section examines the architecture of a high-performance router in detail. The router examined here is the Multi-Service Router (MSR) developed by The Applied Research Laboratory, Washington University, USA [49]. There are three motivations for using the MSR as an example implementation. Firstly, the design of the MSR is such that it reflects proprietary high-performance routers closely. Secondly, because this router is not proprietary, the author had access to details about its inner workings. And finally, because the MSR forms the basis of the author's implementation, it is useful that its architecture be described thoroughly.

The MSR has the following key features.

- Line speeds of up to 2.4 Gigabits per second.
- A switch fabric capable of 160 Gigabits per second.
- Wire speed packet classification and routing.
- Microflow-specific processing of data streams.
- Support for quality of service guarantees.
- Distributed queuing to ensure high throughput, even under extreme overload.

- Support for various signalling protocols such as MPLS and RSVP.
- Active networking functionality.
- Dynamic loading of kernel modules from the control processor.

Firstly the hardware components of the MSR are examined in detail. Then its boot up and configuration procedures are described. Finally its overall operation is described.

3.6.1 The Hardware Components of the MSR

The MSR consists of an ATM switch fabric with embedded programmable processors on each port. (Please refer to Appendix A for an introduction to ATM.) Figure 13 illustrates this architecture.

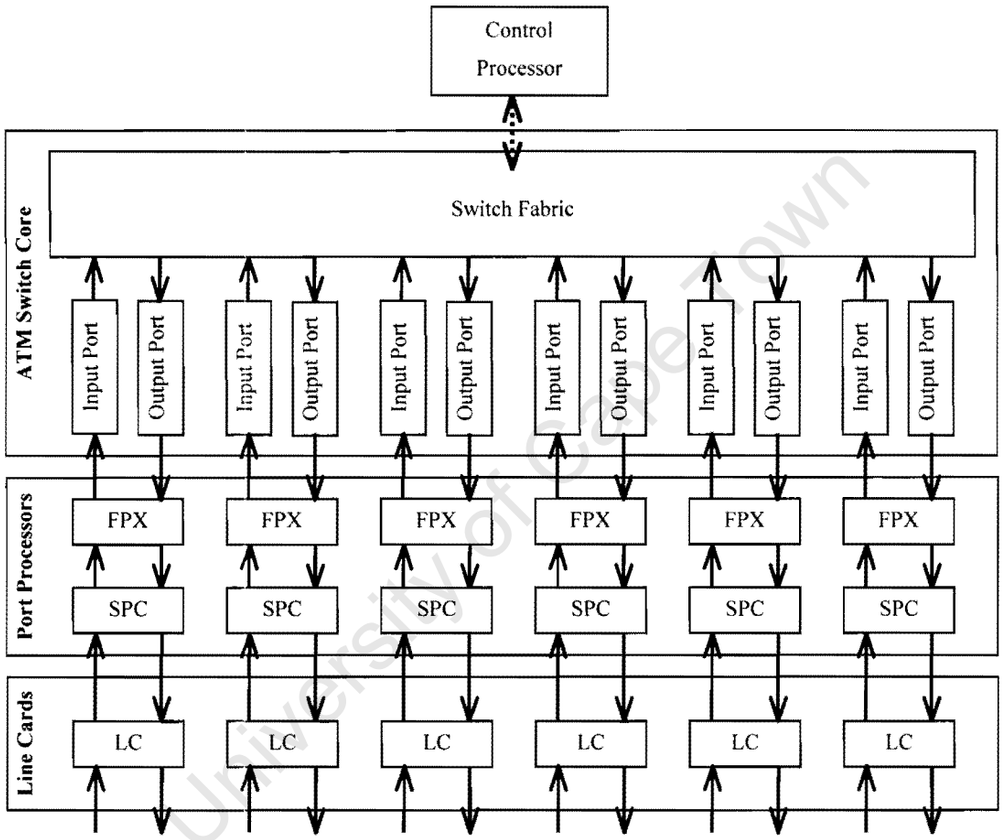

Figure 13: An overview of the MSR hardware showing the switch fabric, switch ports, Field Programmable port eXtenders (FPXs), Smart Port Cards (SPCs) and line cards

The MSR's ATM switch core comprises a Washington University Gigabit Switch (WUGS). The port processors consist of Field Programmable port eXtenders (FPXs)

as well as Smart Port Cards (SPCs). Each element of the MSR will now be dealt with in detail.

3.6.1.1 Control Processor

The control processor is a software program that runs on a stand-alone machine. This machine is attached to a network that is physically connected to one of the MSR line cards. Dedicated ATM Permanent Virtual Circuits (PVCs) logically connect the control processor to the MSR. The control processor uses ATM control cells to communicate explicitly with the switch core as well as with all port processors. The control processor runs software that is responsible for booting, configuring and controlling the MSR as well as monitoring its status.

3.6.1.2 ATM Switch Core

As stated previously, the ATM switch core comprises a WUGS [51]. This consists of an ATM switch fabric with eight input ports and eight output ports. The features of the WUGS are as follows:

- Each port is capable of operating at rates up to 2.4 Gigabits per second.
- The switch fabric is capable of throughputs of 160 Gigabits per second.
- The WUGS handles multicast efficiently.
- The WUGS supports packet level discard, otherwise known as early packet discard. This means that should congestion cause the dropping of ATM cells, the WUGS will drop all cells belonging to a single IP packet rather than dropping cells from many packets. This minimises the number of corrupted IP packets.

The WUGS's ports are connected by four parallel matrices of Switching Elements (SEs). These parallel planes each operate deterministically to distribute the load. The SE interconnect is shown in Figure 14 and is a good example of a Benes Topology with k = 2. Load distribution thus occurs on the first stage of SEs whilst the second and third stages are used to route cells towards their appropriate output ports.

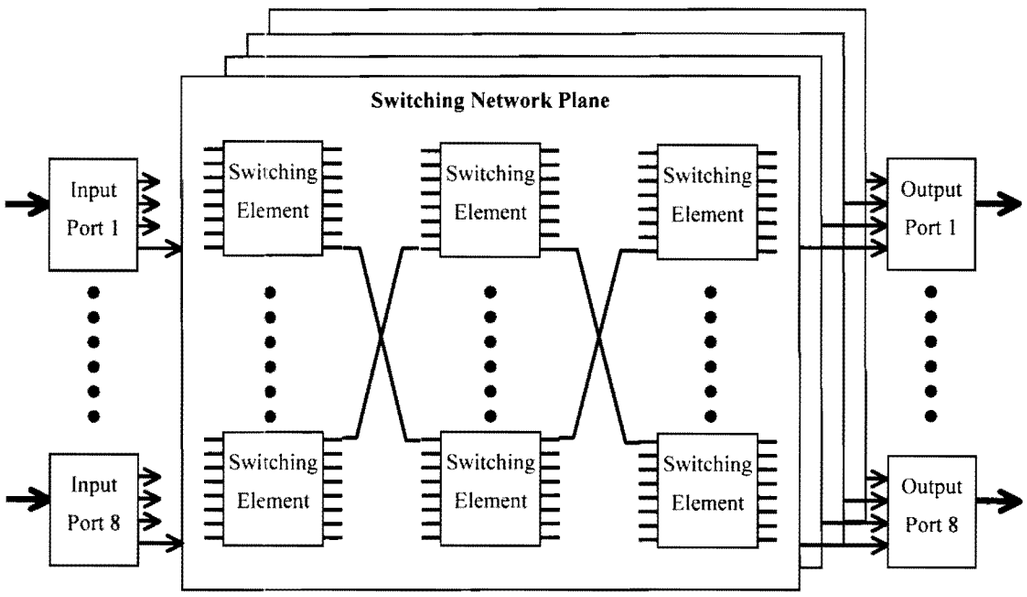

Figure 14: A Benes Topology -the internal structure of a WUGS

When the input port receives a cell from an external link, the cell is sent to a receive buffer. Once the switch is ready to receive the cell, it is sent to the Virtual Path/Circuit Translation table. This provides the necessary ATM-layer routing information. The cell is then ascribed an internal cell header whereupon it is effectively split up into four parts so that each part, together with its internal header, may be sent to one of the four switch element networks.

The cell fragments are sent to the first stage of switch elements where load distribution is performed. Here, adjacent switch elements employ hardware flow-control to distribute the flow of cells to successive stages. This eliminates the possibility of cell loss within the SE network and minimises buffer requirements. Latter switching elements use internal cell headers to switch cell fragments towards their proper output ports.

Once through the SE network, the cell fragments are sent to the appropriate output ports. At the output ports they are reassembled and sent to the output buffers before being transmitted.

The control processor controls the WUGS using a dedicated PVC. This control is performed in real-time using writes to the Virtual Path/Circuit Translation table.

3.6.1.3 Field Programmable Port Extender

The FPX is a Field Programmable Gate Array (FPGA) based system [52,53]. The FPX, which is used as a port processor on the MSR, has the following features.

- The FPX interfaces at line speeds up to of 2.4 Gigabits per second.
- The FPX is guaranteed to perform 9 million IPv4 lookups per second in the worst case. Assuming an average packet size of 256 bytes, this translates to 18.4 Gigabits per second.
- The FPX can support 10 000 forwarding table updates per second with less than 9% degradation in lookup performance.
- The FPX is implemented using dynamic hardware plugins, which combine the re-programmability of a software-based system with the performance of a hardware system.
- The FPX can be reprogrammed over the network by the MSR's control processor.

Figure 15 gives an overview of the FPX architecture. The FPX uses SRAM to store the forwarding table. It also contains two FPGAs. One of these is used to implement the network interface device and the other performs functions related to IP lookups and forwarding table updates. The lookup engines, which implement Eatherton and Dittia's Tree Bitmap Algorithm [54], are time division multiplexed to use all available memory bandwidth on the SRAM interface.

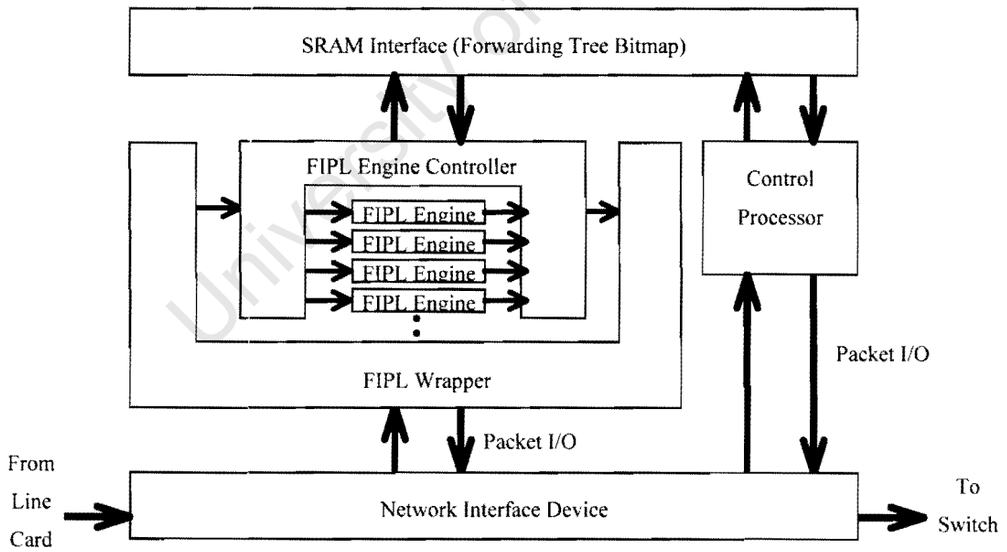

Figure 15: An overview of the FPX architecture showing the data path

The FPX module needs only perform IP lookups on data packets arriving from the line cards. Packets arriving from the switch do not, of course, require lookups as they have already had their lookup performed. These packets may thus pass transparently through the FPX. When a data packet from the line card arrives at the network interface device, it is sent to the Fast IP Lookup (FIPL) wrapper. The FIPL wrapper extracts the packet's destination address and sends this to the FIPL engine controller. The FIPL engine controller enqueues the address and sends it to the next free FIPL engine. It also arbitrates the various FIPL engines' forwarding tree accesses on the SRAM. The FIPL lookup engine performs a longest-prefix map and returns the next hop value for the given packet to the FIPL engine controller. The controller passes this on to the IP Wrapper. Based on the next hop value, the IP wrapper sets the ATM VCI that the packet should be sent to the switch on. This VCI determines which output port the packet will be switched to. The packet is then sent to the switch via the network interface device.

Packets that contain forwarding table updates are forwarded to the control processor. The control processor then updates the forwarding tree on the FPXs' SRAM.

3.6.1.4 Smart Port Card

The SPC is effectively a compact Intel computer that serves as a port processor when sandwiched between the line card and the FPX on an MSR [55]. Figure 16 illustrates the SPC architecture.

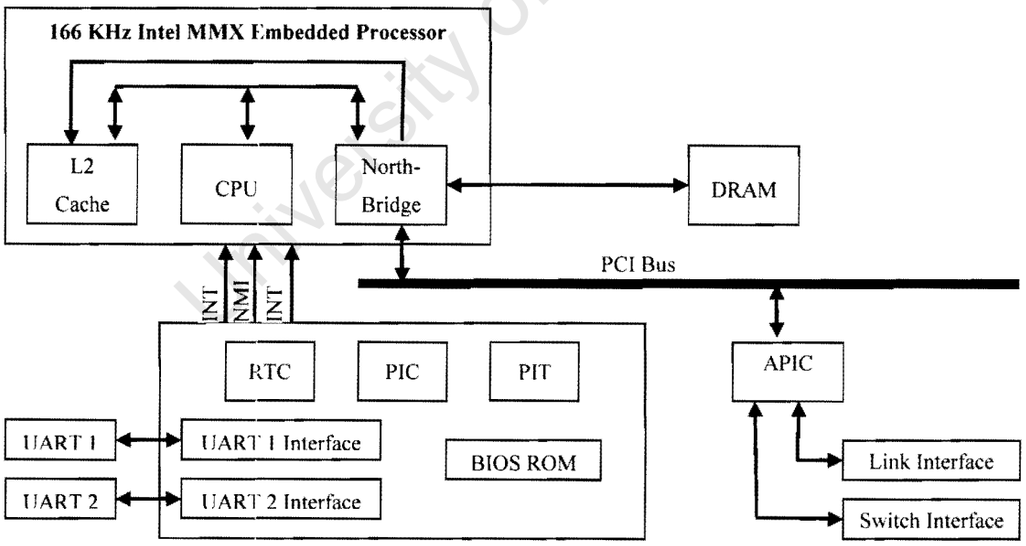

Figure 16: The SPC architecture

A 166 MHz MMX Intel chip constitutes the North Bridge processor of the SPC whilst a system FPGA provides the South Bridge functionality. The SPC has 64 MBytes of DRAM, as well as a.Washington University ATM Port Interface Controller (APIC) host-network interface [56]. The APIC is a network interface chip that allows the SPC to selectively intercept data travelling between the FPX and the switch. Each port of the APIC is capable of full duplex data rates up to 1.2 Gigabits per second.

The SPC runs a modified version of NetBSD and can be booted over the network by the control processor. The control processor first separately downloads the NetBSD kernel and filesystem to the SPC before initiating the booting of the SPC. The SPC contains a serial port that can be connected to a standalone workstation. This allows a user to open an SPC terminal window from the workstation.

The SPC's role in the MSR is to perform buffering as well as to perform any non-standard operations such as:

- Providing differential quality of service
- Packet monitoring and marking
- IP lookups in the absence of the FPX.

Figure 17 illustrates a simplified IP processing data path for the SPC.

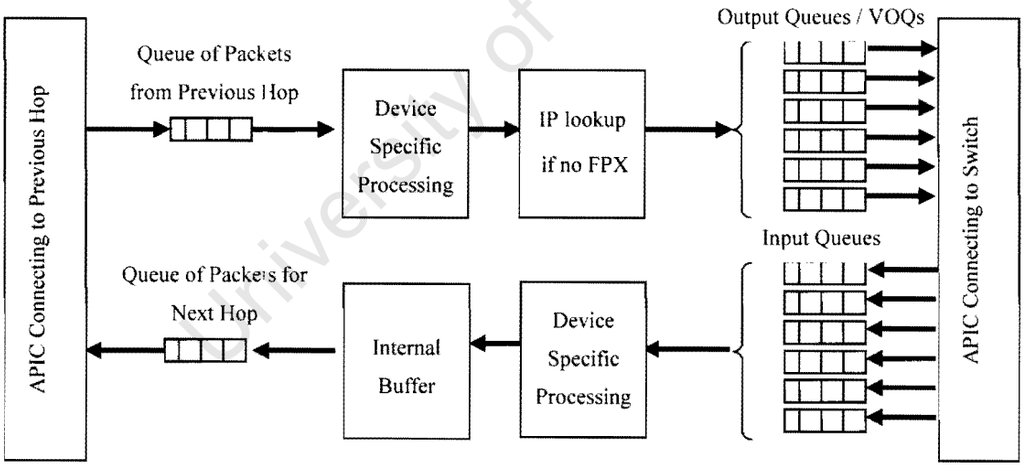

Figure 17: A simplified SPC data path

When data packets arrive at the SPC APIC from the previous hop, they are sent to a receive buffer. The SPC then performs device specific processing on the packets. This could include operations such as monitoring for the purpose of billing, policing, or marking packets. If no FPX is present on that port, then the SPC must perform IP lookups. The SPC is, however, not as fast as the FPX at performing this function. Once all local processing is completed, the SPC sends the packets to the output queues. There is a separate output queue for every destination port on the router. This is designed to prevent head-of-line blocking. The packets are dequeued from these output buffers according to a distributed queueing algorithm. This will be explained further in Section 3.6.3.

When data packets arrive at the SPC APIC from the switch side, they are enqueued. Device specific processing is performed on these packets as before. The packets are then sent to the internal buffer. This is the primary output buffer for each port of the router. It is here that buffer management mechanisms such as RED or fair queuing are implemented. Packets are removed from this buffer as per the scheduling mechanism and sent to the FIFO output queue. This queue, which is always kept short, is dequeued by the APIC.

The SPC II, which was released in 2002, is an upgrade to the SPC. This port card contains a 700MHz Pentium-III processor as well as 256 MB of SDRAM [57] and is intended for use by port applications requiring intensive computational power as well as high data rates.

3.6.1.5 Line Card

The role of the line card is simply to provide physical layer translation between the MSR and the connected network(s). The following media are supported: dual 155 Mbps OC-3 SONET, 622 Mbps OC-12 SONET, 1.2 Gbps HP G-Link as well as dual 1.2 Gbps HP G-Link. A Gigabit Ethernet line card is currently under development.

3.6.2 Booting and Configuration of the MSR

As stated previously, the control processor uses dedicated PVCs to communicate with the MSR. There are VCIs that are dedicated to communication with the switch as well as VCIs for communication with each port processor. For example, port processors

send data to the control processor on VCI number 40 + i where i is their port number. Two-way communication is thus possible.

MSR system configuration begins with ATM switch configuration. This sets up the communication paths between the control processor and all port processors. Following this, the switch discovery phase can begin. Here, the control processor ascertains which port processors and line cards are on each port. This is done by sending control cells to each potential port processor. The receiving port processors in turn report their presence as well as that of the adjacent cards. The control processor then begins downloading a NetBSD kernel and a memory-resident file system to each SPC. This is done using a multicast VC. The SPCs are then booted whereupon each one is sent its port location using MSR control messages. The FPXs are then initialised in a similar manner. In the case of the FPX, a program and configuration is loaded into the application FPGA's reprogram memory. The control processor then sends a control cell, which initiates the reprogramming of the FPGA using the contents of the reprogram memory.

The MSR then runs Zebra [58], which is a route-finding program. The routing data received is used to populate the control processor's routing table. Based on this, the control processor computes the forwarding table for each port. If there is an FPX on a given port, the forwarding table is sent there. If no FPX is present then the forwarding table must be sent to the SPC, which will perform the lookups instead. Any changes made to the routing table are propagated down to the forwarding tables.

3.6.3 The Operation of the MSR

The general mechanism whereby the MSR processes packets is as follows: When a packet arrives at an input port, an IP lookup is performed. This lookup determines which output port the packet should be switched to. The port processor sends the packet to the switch using the ATM VCI that corresponds to this port. The switch then forwards the packet to its appropriate output port based on the VCI that it was received on. The output port buffers the packet if necessary before sending it on to its next hop.

It has been stated previously that although packets should primarily be buffered at the output port, there are a number of packet queues throughout the MSR. In order to ensure that the MSR behaves as an output queuing router, a distributed queuing mechanism is employed [59]. This mechanism prevents sustained congestion on the internal links of the MSR and in so doing prevents reductions in throughput. Put differently: the MSR uses virtual output queuing to prevent internal congestion and head-of-line blocking.

The mechanism employs a coarse scheduling approach whereby each input port periodically (every 100us) broadcasts its backlog to each output port. Output ports similarly broadcast information about their output rates and queue lengths. Input ports are thus able to calculate the rate at which they should send data to each output port. Because a separate VCI is used to send data to each output port, the input ports are easily able to adjust their VC's pacing on the APIC to control the flow of data towards the output port. It is thus possible to keep data moving to the output ports in a timely manner without causing internal congestion.

Chapter 4 will describe the design issues taken into consideration when the proposed mechanism was created, with reference to the architecture of modern routers.

Chapter 4

Design Considerations

4.1 Motivation for H-MAQ

Diffserv is a popular technology for implementing tiered service levels in IP networks. One of the main reasons for this is its ability to scale to large network sizes. The conventional wisdom is that a microflow-unaware active buffer management mechanism such as RIO should be used in the core of Diffserv networks. The drawbacks of using such mechanisms were discussed in detail in Section 2.6.2.1 and can be summarised as being limited fairness, unpredictability, instability and uncontrollability.

Not being able to predict the behaviour of a network quantitatively means that it is only possible to offer relative service differentiation between classes and not absolute guarantees [32]. The value of providing a relative service is questionable as SLAs are defined in absolute, not relative terms. This means that network operators once again have to resort to the extensive use of over-provisioning to guarantee services. This results in the shortcomings given in Section 2.3. A further drawback it that because the core's behaviour is unpredictable and highly dependent on traffic type, a large and unexpected change in traffic profile could result in a network suddenly not being able

to provide a contracted service to a client. Such an outage can result in large penalties being incurred.

As a solution to these problems, this study proposes the employment of core network elements that are aware of competing microflows and aggregates, as well as being aware of competing packets' DSCPs like RIO. Using such a mechanism has the following benefits:

- Network managers would be able to explicitly allocate network resources to aggregates making it possible to provide service guarantees on an aggregate basis.
- Fairness would be improved within aggregates. This would provide a more predictable service to end-users.
- Access control could be improved to make better use of network resources. One reason is that core routers would have access to explicit information about the reservation and utilization of resources by aggregates along all internal network links. Another reason is that resources would be explicitly allocated to aggregates on all links of the network.
- Network utilization would be increased because a more predictable, controllable and stable network core would reduce the amount of over-provisioning needed.
- Should a Diffserv network migrate to the proposed mechanism, external interfaces with other networks or clients would not need to be changed. This is because the mechanism concerns only the inner workings of the Diffserv network.
- Because the proposed mechanism uses a drop-from-front packet drop policy, TCP responsiveness would be improved, as described in Section 3.5.
- Because the proposed mechanism provides the means to allocate different maximum buffer sizes to different aggregates, it would be simple to support specialized services such as a low delay service for individual aggregates. This is described in Section 5.2.1.

Although it is possible to implement the proposed routers incrementally on a Diffserv network, it is only once all core routers in the network implement H-MAQ and effective admission control exists that the behaviour of traffic would become predictable and controllable. Only once this predictability and controllability have been achieved, does it become possible to guarantee quantitatively the service levels

offered by a Diffserv network without significant over-provisioning. As was mentioned above, should a network migrate to H-MAQ, it is only the network's internal interfaces that would have to be modified. The network would still offer a Diffserv service to the outside world and, as a result, existing external interfaces and service level agreements would not need to be changed.

4.2 Overview of H-MAQ Networks

This section gives an overview of the operation of H-MAQ networks. This is done in three parts: first the behaviour required of the edge network elements is described, and then the behaviour of the H-MAQ core network elements. Finally, the overall network design is described.

4.2.1 Edge Network Element Behaviour

With regards to user traffic, the behaviour of edge routers implementing Diffserv AF where H-MAQ is implemented in the core is identical to that when standard RIO routers exist in the core. These routers perform policing of aggregates to limit load on the core. Aggregates are policed on a per-microflow basis. Excess is re-marked to a higher drop precedence level or, in the extreme case, dropped.

The behaviour required of edge routers when H-MAQ is implemented in the core does, however, differ from the standard case with regards signalling. In addition to standard Diffserv signalling, the proposed routers are required to reserve resources for aggregates on core routers within the Diffserv network explicitly.

4.2.2 H-MAQ Core Network Element Behaviour

The behaviour of core routers implementing H-MAQ differs markedly from that of classic Diffserv core routers. The proposed routers implement a hierarchical scheduling mechanism that provides guarantees to aggregates by explicitly allocating link capacity to them. Within each aggregate there is a microflow-aware fair queuing mechanism that allocates a separate FIFO queue to each microflow. These queues are serviced using Deficit Round Robin (DRR) [60]. A drop precedence-aware variation of QSDRR's drop policy is implemented on each aggregate. The benefits of this drop-from-front policy are described in Section 3.5.

4.2.3 Overall Network Design

The overall design of H-MAQ enabled Diffserv AF networks allows the control of resource allocation at all network elements both on the edge and the core of the network. This model differs from the standard Diffserv model where only the edge of the network is controllable, and the core is both largely uncontrollable and unpredictable. The design of H-MAQ will be specified more clearly in Chapter 5.

4.3 Feasibility of Proposed Mechanism

The proposed design does raise the issue of scalability. This is because it has a microflow-aware core and because it performs resource allocation at a finer granularity than conventional Diffserv networks. There are two areas where the scalability of the proposed mechanism could be brought into question. These are in network signalling and in the processing of user traffic. These areas will be considered next.

4.3.1 Signalling Feasibility of Proposed Mechanism

The proposed mechanism employs signalling paths whereby edge routers explicitly allocate resources to aggregates on core links. Because this resource allocation is performed on an aggregate basis, rather then on a per-microflow basis, as is the case with Intserv, there is no reason why any scalability problems should be incurred. In the extreme case, static provisioning would eliminate the need for such signalling paths. Using dynamic mechanisms such as aggregate RSVP [61] would, however, result in more flexible networks. Furthermore, the use of label-switching technologies such as Multi Protocol Label Switching (MPLS) would make aggregate detection simpler in Diffserv networks.

4.3.2 User Traffic Feasibility of Proposed Mechanism

It has been suggested that per-microflow fair link sharing is prohibitively expensive to perform on router port processors because of its computational overhead. This issue was considered in detail in Section 3.4.2 with the conclusion being that current port processors are able to perform these functions at line speeds above those of current Internet core routers.

In Section 2.2.1, it was stated that the usage of the Internet is doubling on a yearly basis. Section 3.4 stated that router capacity increased by only a factor of ten in the last five years. It is thus clear that router capacity is not currently matching the Internet's growth. This situation is more sustainable than it may at first appear as not only are router capacities increasing, but so too are the number of routers in the Internet. Nonetheless, if the Internet of the future is to provide guaranteed service levels together with high levels of resource utilization, router manufacturers will be required to rise to the challenge of producing increasingly powerful microflow and aggregate-aware routers.

4.4 Scope of Implementation

This project aims to address the shortcomings related to Diffserv AF that were described in Section 2.6.2.1. This is done by the proposal and evaluation of an alternative buffer management mechanism (H-MAQ) to be used in core network routers. The evaluations will consider all of the shortcomings discussed in Section 2.6.2.1 excepting the following two issues.

- The issue of the effect of packet size on fairness will be omitted as it can be remedied simply by employing a mechanism that takes packet size into account when determining whether to enqueue or drop arriving packets. This mechanism was described in Section 2.6.2.2.
- The issue of instability will not be considered, as the author was unable to demonstrate that H-MAQ is more stable than RIO. This was due to limitations in the ability of the traffic sources to generate sufficient traffic volumes. It is, however, the author's opinion that H-MAQ does provide better stability than mechanisms such as RIO. This is because H-MAQ, unlike RIO, includes no low-pass filtering component.

The following chapter gives a technical specification for H-MAQ. This specification forms the blueprint of the author's implementation. It is notable that this specification considers buffer management and scheduling only. No signalling is considered. It can, however, be argued that the required signalling functions are trivial due to the fact that should reservations be performed, this would happen on a per-aggregate basis. Such reservations would thus be infrequent. The specification considers only one AF

sub-class of service but the results apply to all four AF classes. This is because the AF classes are specified to behave independently. Only two drop precedence levels will be considered rather than the maximum permissible number of three. This conforms to IETF specifications as described in Section 2.5.2.

Chapter 5

Technical Specification for Implementation

5.1 Edge Router Mechanism

The edge router is responsible for policing traffic entering the Diffserv AF network. The policing should occur on an aggregate basis as well as on a microflow basis. This is implemented as follows: The router periodically divides the data rate for each aggregate between its competing microflows. This yields the police rate (*Rp*) for each microflow. If a microflow's sending rate exceeds its police rate, the edge router begins marking that microflow's packets as being out-of-profile. This is done using the Time Sliding Window (TSW) tagger [25].

The TSW tagger consists of two distinct parts: a rate estimator and a tagging algorithm. The rate estimator estimates each microflow's sending rate upon each packet arrival. It maintains three state variables: *Win_length*, which is a pre-configured window length measured in units of time; *Avg_rate*, which is the current rate estimation; and *T_front*, which is the time of the last packet arrival. Their initial values are:

Win_length = *a constant*;

Avg_rate = *Rp;*

T_front = *0;*

Upon each packet arrival, the state variables are updated as follows:

Bytes_in_TSW = *Avg_rate * Win_length;*

New_bytes = *Bytes_in_TSW + pkt_size;*

Avg_rate = *New_bytes / (current_time –T_front + Win_length);*

T_front = *current_packet_arrival_time;*

This decaying rate estimator produces the *Avg_rate* variable that is used by the tagging algorithm. The tagging algorithm looks for the point where a microflow rate exceeds 1.33 * *Rp*. At this point the tagger starts tagging packets as out-of-profile.

Both the in- and out-of-profile packets are sent into the core of the Diffserv network. In the extreme case where the link connecting the edge router to the rest of the Diffserv network is congested, the edge router may drop packets.

5.2 Core Router Mechanism

The proposed H-MAQ core router mechanism is described it this section. The section begins with a discussion on the allocation of internal buffers. Thereafter, the data flow is described. Finally, H-MAQ's drop policy is described.

5.2.1 H-MAQ Internal Buffer Allocation

As described in Section 3.3, the internal buffer of a router port is the primary buffer for packets that are to exit on that port. In general, ports with higher data output rates can tolerate larger internal buffers without incurring excessive delay. This is because they are dequeued more frequently. The amount of buffer space that is allocated to the internal buffers of output links is governed by the trade-off between link utilization and permissible delay. Allowing longer queues results in increased delays whilst shorter queues can result in underflow leading to reduced link utilization.

H-MAQ uses a per-aggregate buffer allocation policy. This means that each aggregate is allocated a certain amount of internal buffer space. In general, the amount of buffer

space that is allocated to an aggregate should be proportional to its share of the output link. This would provide similar delay and utilization characteristics for all aggregates. Nonetheless, this need not be the case. Should a given client require a customised service, such as a low-delay service, this could be provided simply by reducing the amount of internal buffer space allocated to that aggregate. This point highlights the high degree of flexibility that the proposed solution provides.

5.2.2 H-MAQ Data Flow

H-MAQ routers implement a hierarchical scheduling mechanism that provides guarantees to aggregates by explicitly allocating link capacity to them. The core routers are required to perform aggregate-aware and microflow-aware queuing. In addition, the mechanism takes into account the drop precedence of packets. Figure 18 shows an example of a logical data path that may be found on the output port of a core router implementing H-MAQ.

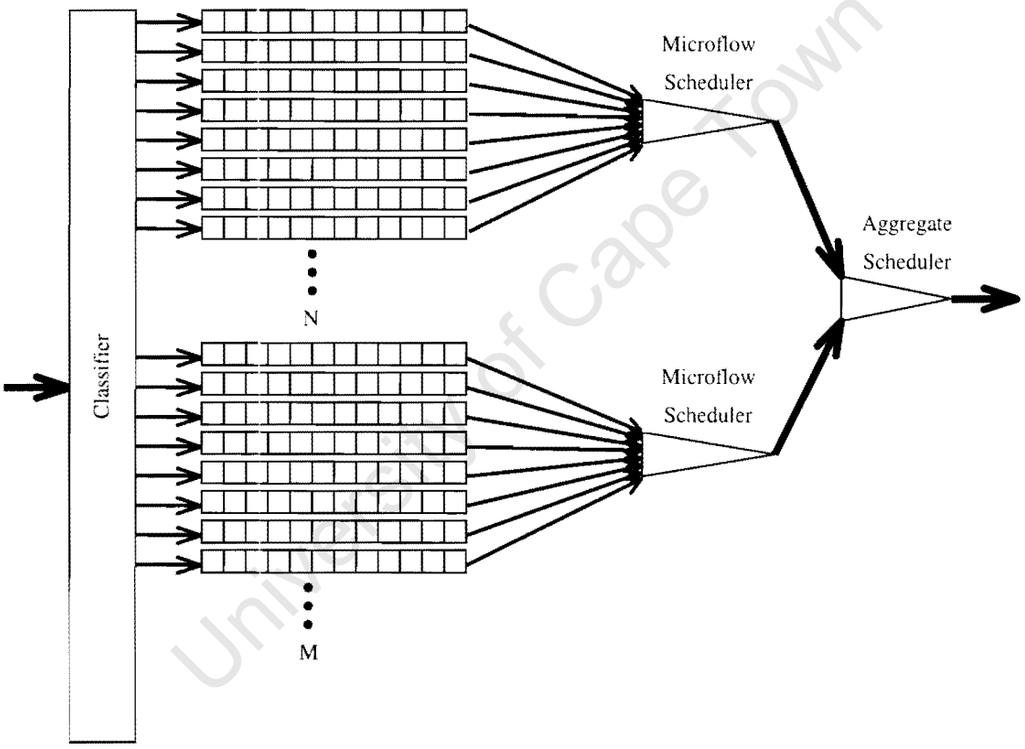

Figure 18: An example of an H-MAQ logical data path

This is an example of a data path for the case where there are two aggregates sharing an output link. In practice, the number of aggregates could be far greater. The data path includes a packet classifier, and two banks of queues. Each bank of queues is

allocated to a separate aggregate. Within each bank of queues, there is a separate queue for each microflow. There is a microflow scheduler allocated to each aggregate. This scheduler services each microflow queue belonging to that aggregate in turn using Deficit Round Robin (DRR). In the final stage, the aggregate scheduler puts the traffic from each aggregate onto the output link according to the aggregates' service level agreements. Any excess available bandwidth is shared out amongst backlogged aggregates in proportion to their service level agreements.

5.2.3 H-MAQ Drop Policy

H-MAQ uses a drop policy that is an enhancement of the QSDRR drop policy. This enhancement extends QSDRR to be able to handle multiple drop precedence levels as well as multiple aggregates. The drop policy maintains a separate drop queue pointer for each aggregate. This improves isolation between aggregates. The drop policy for a single aggregate is explained below. This mechanism is duplicated for each aggregate.

If the arrival of a packet means that an aggregate has exceeded its allowable buffer space, a search is performed to find the microflow queue with the highest number of bytes belonging to out-of-profile packets. This queue becomes the drop queue. The out-of-profile packet that is closest to the front of the drop queue is discarded. Assuming that the aggregate remains in excess of its allowable buffer space, packets will continue to be removed from the drop queue until it has fewer out-of-profile bytes than any other queue. Once this happens, the queue with the most out-of-profile bytes becomes the new drop queue. This process continues until the aggregate is no longer in excess of its allowable buffer space.

Should the network be ill provisioned, it is possible that an aggregate could be in excess of its allowable buffer space whilst that aggregate has no out-of-profile packets enqueued. If this happens, a search is performed to find the queue with the most bytes in it. This queue becomes the drop queue. The packet that is at the front of the drop queue is discarded. Assuming that the aggregate remains in excess of its allowable buffer space and that no out-of-profile packets arrive, packets will continue to be removed from the drop queue until it has fewer bytes than any other queue. Once this happens, the queue with the most bytes becomes the new drop queue. If, at any stage,

an out-of-profile packet arrives, the drop queue will be changed to the queue with the out-of-profile packet in it, so that the newly arrived out-of-profile packet is removed.

Chapter 6

Design and Implementation of Testbed

6.1 Choice of Platform

Both discrete event simulations and network emulation can be used to evaluate network architectures. Network emulation is considered to be more desirable as it models real networks more closely. The following points relate to these models:

- Emulated networks run real protocols. This improves their accuracy in modelling real networks.
- The processing overhead in emulated networks is real. This, once again, improves their accuracy.
- The accuracy of simulators as well as the modelling assumptions that they use are questionable.

For these reasons the author chose network emulation to evaluate the proposed network architecture.

The author considered using The Click Modular Router as his testbed router implementation [62]. Click routers run on multipurpose Linux machines and are capable of supporting advanced quality of service mechanisms. The author decided

against using The Click Modular Router because multipurpose machines are not representative of the physical architectures of real Internet routers.

The Washington University MSR was chosen as the platform for the author's implementation. This router's architecture is described in detail in Section 3.6 and is representative of current high-performance routers. Because the MSR is a development router, the code running on its SPCs is open source. This gave the author full control of the router's behaviour. It also meant that the author had access to many programming tools and macros that had been written for MSR developers.

6.2 Hardware Components

The testbed was implemented using a single MSR and two end stations. The MSR consisted of a WUGS with an SPC and a line card on two of its ports. No FPXs were used in the testbed as performing fast IP lookups was not considered to be of critical importance.

The end stations were each implemented on 1GHz Pentium machines running NetBSD 1.4.1 as their operating systems. Each end station had an APIC card that was connected to one of the two MSR line cards using fibre-optic cables. The MSR's control processor was run on one of the end stations.

6.3 Network Topology

Figure 19 illustrates the conceptual topology that forms the basis of the author's evaluations.

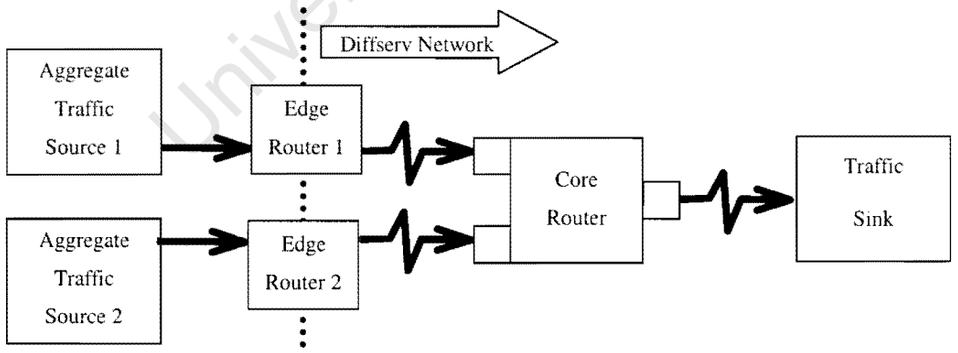

Figure 19: Conceptual topology of testbed

This topology includes two aggregate traffic sources that send data to a Diffserv network. When the packets reach the Diffserv network edge routers, policing and remarking takes place. This occurs on a microflow basis according to the aggregates' SLAs. The packets are then sent to the core router and from there to the traffic sink. The link from the core router to the traffic sink is the primary resource that traffic sources compete for. The ACKs return to the traffic sources along the same path. There is no network congestion along the ACK path. A delay is incurred on all links and network elements.

The previous conceptual topology was emulated using the hardware components described in Section 6.2. Figure 20 shows how the topology was implemented using these components.

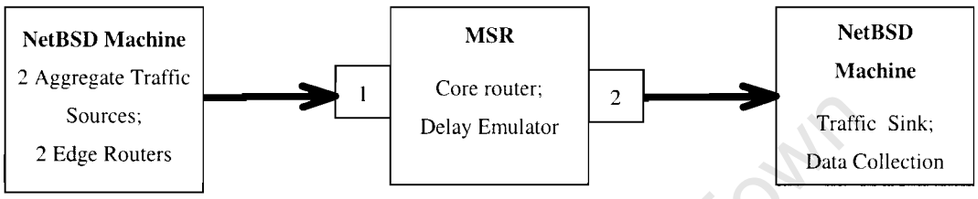

Figure 20: Implementation of testbed

Figure 20 shows that both aggregate traffic sources and both edge routers were implemented on a single NetBSD machine. The traffic sink for both aggregate traffic sources was also implemented on a single NetBSD machine. The delay emulator was implemented on Port 1 of the MSR and affected packets on the return path only. The delay emulator's implementation allowed the author to select which traffic flows or aggregates should incur a delay, and how long that delay should be. Both RIO and H-MAQ were implemented alternately on both ports of the MSR. The following sections describe the components of the testbed in detail.

6.4 Traffic Sources

It is important that the traffic sources are able to respond to feedback from the network. As a result, the use of traces as traffic sources is not appropriate. A number of existing software packages simulate network traffic sources. The author considered using GenSyn [63] and NetPerf [64], for example, but was unable to do so as they did

not run on NetBSD. The author thus decided to write his own traffic generator program.

When designing a network traffic generator, it is necessary to consider the nature of Internet traffic. Not only is the current makeup of Internet traffic applications divergent, as shown in Section 2.2.2, but this is changing continually. Furthermore, one needs to consider what the effect on application traffic makeup will be when technologies such as Diffserv make the transport of real-time traffic on IP networks possible. Presumably the best effort data traffic will continue to increase as it has been doing, but there will be a great increase in real-time traffic such as streaming voice and video. Because such traffic is likely to be carried using Diffserv, the application makeup of future Diffserv traffic may be very different from what it is currently.

The author considered the viability of simulating streaming GSM encoded voice as well as streaming MPEG 2 encoded video. These would require data rates of 13Kbps and 6Mbps respectively. Because the author's evaluations required hundreds of traffic sources to be interleaved, it was clear that that the modelling of all of these traffic sources would not be possible on the testbed due to hardware limitations.

The author thus decided to implement a generic traffic source that did not attempt to mimic the data flow of any application in particular. An advantage of this approach is that using a single generic traffic source instead of simulating a number of applications eliminated an extraneous variable from the experiments. Conclusions could thus be drawn more clearly from testbed data.

The aggregate traffic source consisted of between 50 and 300 independent processes. Each process represented an application level traffic source. Each application traffic source opened a single TCP or UDP socket that was used to send packets into the network. The packets were sent into the network with an exponentially distributed random inter-arrival time. This minimized correlation between the packet send times of each application traffic source. In times of extreme network congestion, the exponential traffic sources would decay into persistent sources. The TCP sources were, of course, also limited by TCP flow-control. The application traffic source processes went to sleep between packet sends to minimise their use of processor

resources. The packet size used was 500 bytes, which is close to the mean value for Internet traffic. The average data rate of the traffic and the transport protocol were determined by the experiment.

6.5 Edge Router

As stated in Section 5.1, the role of the edge router is to police and, where necessary, re-mark packets. These functions are performed on a per-microflow basis using the algorithm described in Section 5.1. The policing and re-marking was performed on the same NetBSD machines as the traffic sources. Because the policing and re-marking is performed on a per-microflow basis, it was decided that each application traffic source process would also be responsible for policing that microflow. The DSCP of each departing packet thus only needed to be set once. The police rate was determined by the experiment.

6.6 Core Router

The core router was implemented using an MSR. The MSR's external link rates were set to 4 Mbps and its internal link rates were set to 6 Mbps. To ensure that the router's backlogged packets were queued primarily at the internal buffers of SPCs, the rate at which packets were dequeued from here was set to 1.5 Mbps. The size of the internal buffers was limited to 200 packets. Because of the low load on the input ports of the router and the fact that only two ports were used, it was not necessary to use distributed queuing on the MSR.

The MSR uses a hashing table to identify which microflow the packets that are to be enqueued in the internal buffer belong to. Because the microflows in the experiments differed only in destination port and protocol, it was necessary to modify the hashing function to give greater weight to these elements. It was also necessary to increase the total size of the hash table to 10 000. Only once these modifications had been made did each microflow tend to hash to a unique hash value.

In order to evaluate the performance of H-MAQ, it was necessary to test its performance as well as the performance of the best performing microflow-unaware active queuing mechanism, namely RIO (refer Section 2.6.1.2.c). Both of these

mechanisms were thus implemented on the router. The mechanisms were responsible for buffering packets before sending them towards their destination as well as for providing fair link utilization.

The mechanisms were implemented as kernel modules on the SPCs. The RIO implementation was written following the relevant specifications [25,28]. A simple FIFO queue that was written by F. Kuhns of Washington University was used as the basis for the author's RIO implementation. Refer Appendix B for a description of the RIO development and testing procedure. As stated in Section 2.6.1.2, RIO includes a number of tuneable parameters. The author chose values for the in- and out-of-profile *Linterm* (*LintermIn* and *LintermOut*) as recommended by the RIO designers. The choice of *ThreshIn, MaxThreshIn, ThreshOut* and *MaxThreshOut* were determined firstly by the acceptable buffer occupancy levels for experiments; as well as by the designer's recommendation of having staggered parameters with maximum thresholds of at least twice the minimum thresholds. The *q_weight* parameter was chosen so as to fall within the range set by the RED designers. The RED designers specify that *q_weight* should be above 0.001 and below the value that is determined by the following equation:

$$L + 1 + \frac{(1 - q\_weight)^{L+1} - 1}{q\_weight} < Thresh$$

Where L = the maximum burst size. Letting $Thresh$ = $MaxThreshIn$ = 30 and allowing a maximum burst size of 40 packets, the following constraint was arrived at: $q\_weight < 0.09$. Thus,

$$0.09 > q\_weight > 0.001.$$

For this simulation it was found that a value of 0.04 for *q_weight* produced good results by allowing bursty traffic whilst converging to equilibrium in an acceptable time.

The chosen RIO parameters were as follows:

q_weight	=	0.04
LintermIn	=	50
LintermOut	=	20

MaxThreshIn = 160
ThreshIn = 80
MaxThreshOut = 60
ThreshOut = 30

H-MAQ was implemented according to the specification given in Section 5.2. The author's implementation used static provisioning of resources to the two aggregates as well as static drop thresholds on each queue. If ever an aggregate was to be dequeued, but it had no backlogged packets, the excess capacity would be given to the other aggregate. The author's implementation was created by extending a QSDRR buffer management mechanism written by A. Kantawala and F. Kuhns of Washington University.

6.7 Traffic Sink

The traffic sink was implemented on a single NetBSD machine. This machine, which was connected to Port 2 of the MSR, was responsible for listening for incoming UDP and TCP connections. Once a connection had been made, the sink would receive all incoming packets. Upon receiving a packet, the sink would write the data pertaining to that packet to a file. Once a packet's relevant data had been stored the packet was discarded.

6.8 Delay Emulator

The author used a loadable kernel module, written by A. Kantawala, on the MSR to emulate network delay. This module buffered arriving packets for a fixed duration as determined by the experiment. Because it is possible to specify which microflows the delay emulator affects, the author was able to emulate different round trip times for different microflows or aggregates. The delay emulator was run on Port 1 of the MSR and affected packets on the return path.

6.9 Data Collection

Data collected during experiments needed to include the data rate of microflows and aggregates as well as the transit delay for packets. Experimental data collection was performed as follows.

All end stations were synchronised using the Network Time Protocol (NTP), which is an open source program used to synchronise computers over networks [65]. NTP is said to provide sub-millisecond precision when used to synchronise computers that are on the same LAN [66]. The NTPQ query program was used to inspect the offsets between two of the end stations. The results of this command, run at 11:40 on 05 March 2003, confirmed an offset of only 7 microseconds. This is illustrated in Figure 21.

```
[root@maya ~]#ntpq -p
     remote           refid      st t when poll reach   delay   offset    disp
==============================================================================
*yucatan.uct.exp LOCAL(0)        11 u    1   64  377     0.42   -0.007    0.93
```

Figure 21: A sample ntpq –p command's output

Before sending data packets, traffic sources would check the current time using the gettimeofday function and place this in the packet payload. The packets would pass through the testbed network until being received by the traffic sink. Upon receipt, the sink checked the current time and wrote this, together with the time that the packet was sent, to a log file. Using the send time and the arrival time, it was possible to determine the transit delay of the packet. In addition to the send and receive times, other information pertaining to the packets was stored. This information included the DSCP of the packet, the source address of the packet, the source port of the packet and the packet size. A log file was thus produced that detailed the network behaviour during experiments. This log file could later be analysed to determine the experimental findings.

Chapter 7

Evaluations, Results and Analysis

This chapter describes the evaluations that were performed on the testbed, the results obtained and an analysis of the results. The evaluations compare the performance of a Diffserv AF network with RIO implemented in its core routers to a Diffserv AF network with H-MAQ implemented in its core routers. For each round of evaluations, the set-up of the testbed is described. The results of the evaluation and then an analysis of the results follow this. Note that because all data was collected at the traffic sink, all statistics for throughput and fairness quote the effective throughput values – also known as goodput.

7.1 Microflow Fairness – Transport Protocol

Following is a comparison between the performance of a Diffserv network that has RIO in its core and that of a Diffserv network with H-MAQ implemented in the core, using microflow fairness according to transport protocol as its performance metric. The scenario of competing individual end-users within an aggregate using different transport layer protocols is being emulated here. An example could be the case where one user is browsing the web (TCP), and their colleague is simultaneously using a UDP voice over IP service.

7.1.1 Evaluation

In this round of evaluations, 100 microflow traffic sources were started. Of these, 90 were TCP sources and the remaining 10 were UDP sources. The TCP to UDP ratio was thus similar to what would be found in the Internet. The sources all started between 0 and 2 seconds. They sent packets through the network for 100 seconds. Data was collected between 10 and 90 seconds. The target rate of the sources and the police rate were as described in the Results section.

7.1.2 Results

Figure 22 gives the results of the evaluation that considers microflow fairness according to transport protocol.

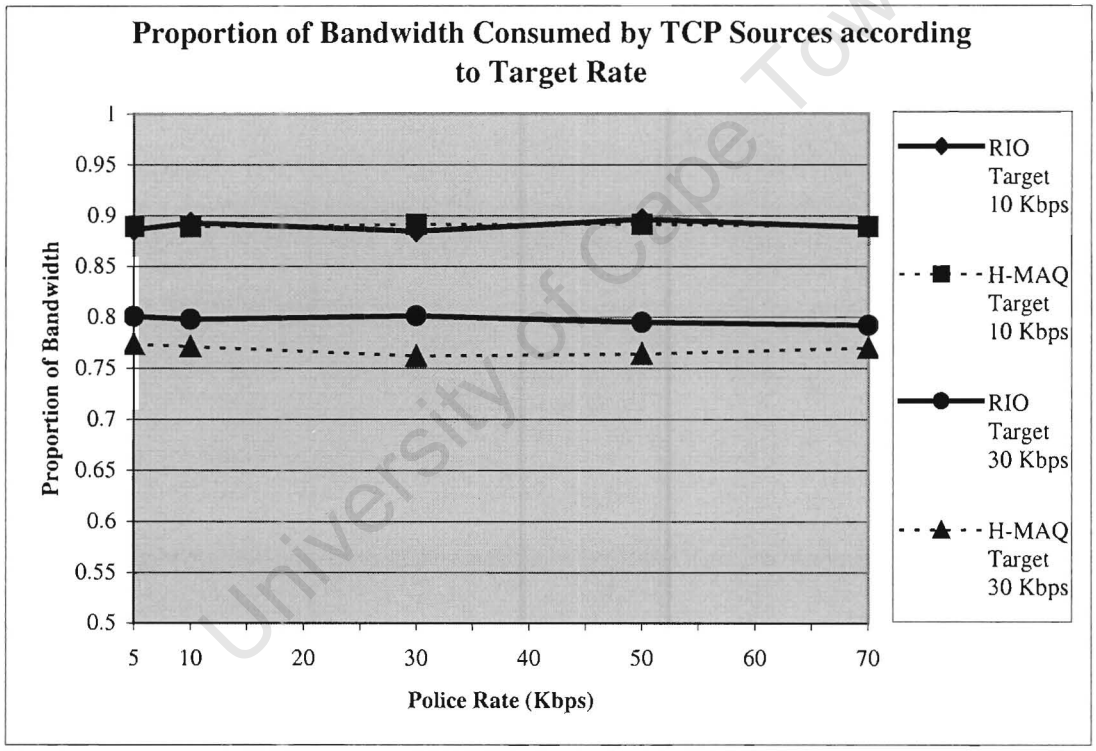

Figure 22: The proportion of bandwidth consumed by the TCP sources when they make up 90% of the competing sources

Figure 22 shows the results of 20 experiments in which the source target rates, the police rates, as well as the core router mechanism were all varied. The graph shows that when the sources' target rates were set to 10 Kbps, the TCP sources consumed between 89 and 90% of the used bandwidth. This was irrespective of whether RIO or H-MAQ was used in the core. When the source target rates were set to 30 Kbps and RIO was used in the core, the TCP sources consumed approximately 80% of the used

bandwidth. When H-MAQ was used in the core, the TCP sources consumed approximately 77% of the used bandwidth. Variations in the police rate had little effect on the proportion of bandwidth consumed by the TCP sources.

7.1.3 Analysis of Results

Because the TCP sources make up 90% of the total number of sources, they should ideally consume that proportion of the bandwidth. However, because only TCP has transport level flow-control, it receives significantly less than 90% as congestion levels increase. This is demonstrated in Figure 22. A comparison between the performance of RIO and H-MAQ reveals that when the target rates were 10 Kbps and there was thus no congestion, both the RIO and H-MAQ networks showed good results. When the target rates were 30 Kbps, both the RIO and the H-MAQ network cores showed reduced fairness. The RIO core was, however, 4% fairer to the TCP sources than the H-MAQ core.

The RIO core performed better than the H-MAQ core because of differences between the traffic profiles of TCP and UDP. Whereas UDP has a smooth traffic profile, TCP has a more bursty traffic profile due to its flow-control mechanism. This is explained in Section 2.1.3.2. Because the RIO drop mechanism monitors the average buffer occupancy level rather than the current buffer occupancy level as H-MAQ does, TCP's bursts are more easily absorbed by the RIO buffer. This puts TCP at an advantage.

The performance comparison between H-MAQ and RIO may be summarised as follows: In times of low congestion, H-MAQ and RIO had similar performance. When the congestion levels were high, RIO performed 4% better than H-MAQ.

7.2 Microflow Fairness – Round Trip Times

Following is a comparison between the performance of a Diffserv network that has RIO in its core and that of a Diffserv network with H-MAQ implemented in the core, using microflow fairness according to round trip time as its performance metric. The scenario being emulated here is that of competing individual TCP end-users within an aggregate being subject to different round trip times. An example could be the case

where one user is performing an FTP transfer to a nearby branch, and their colleague is simultaneously doing an FTP transfer to a branch in a distant city. Only TCP is considered because the concept of round trip time is meaningless in the case of UDP.

7.2.1 Evaluation

In this round of evaluations, 120 TCP microflow traffic sources were started between 0 and 2 seconds. They sent packets through the network for 100 seconds. Data was collected between 10 and 90 seconds. The target rate of the sources and the police rate were as described in the Results section. On the return path, half of the microflow traffic sources were subject to a delay with approximate values of 50, 200 or 400ms.

7.2.2 Results

Figure 23 gives the results of the evaluation that considers TCP microflow fairness according to round trip time.

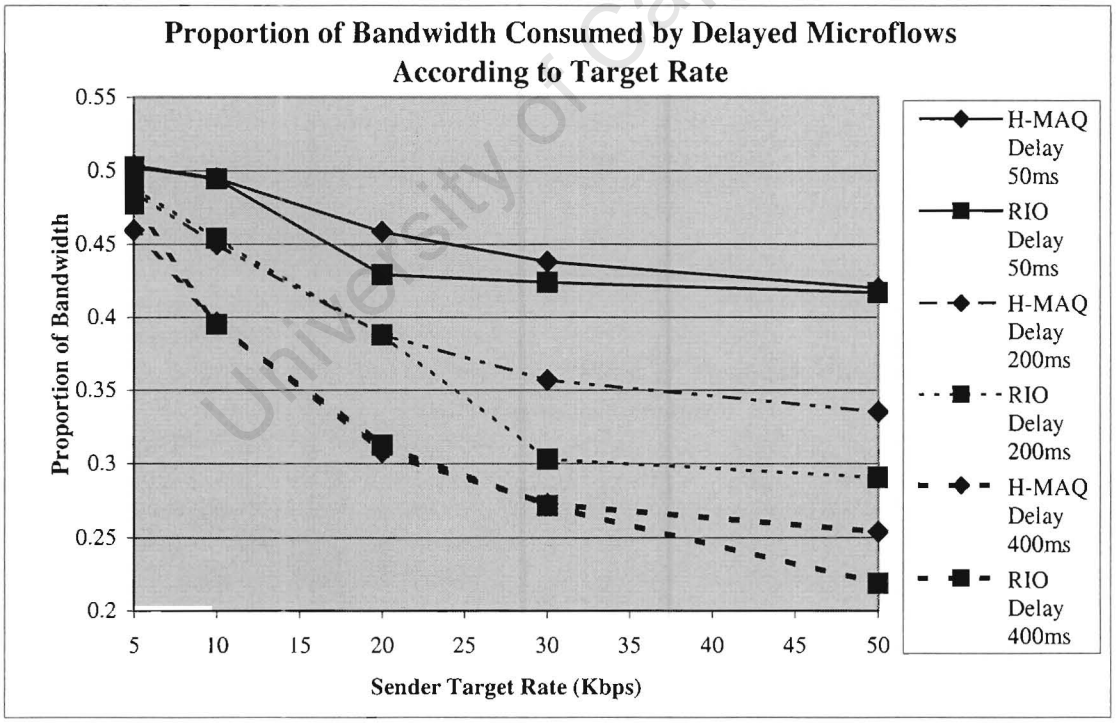

Figure 23: The proportion of bandwidth consumed by the delayed sources according to target rate when all packets are in-profile

Figure 23 shows the results of 30 experiments in which the round trip times for half of the microflows were varied, as well as the source target rates and the core router mechanism. This shows that in general, as the delay time increases, fairness decreases. On the left-hand side of the graph, the delayed sources received approximately 50% of the resources. This proportion decreased persistently as the sources' target rates increased.

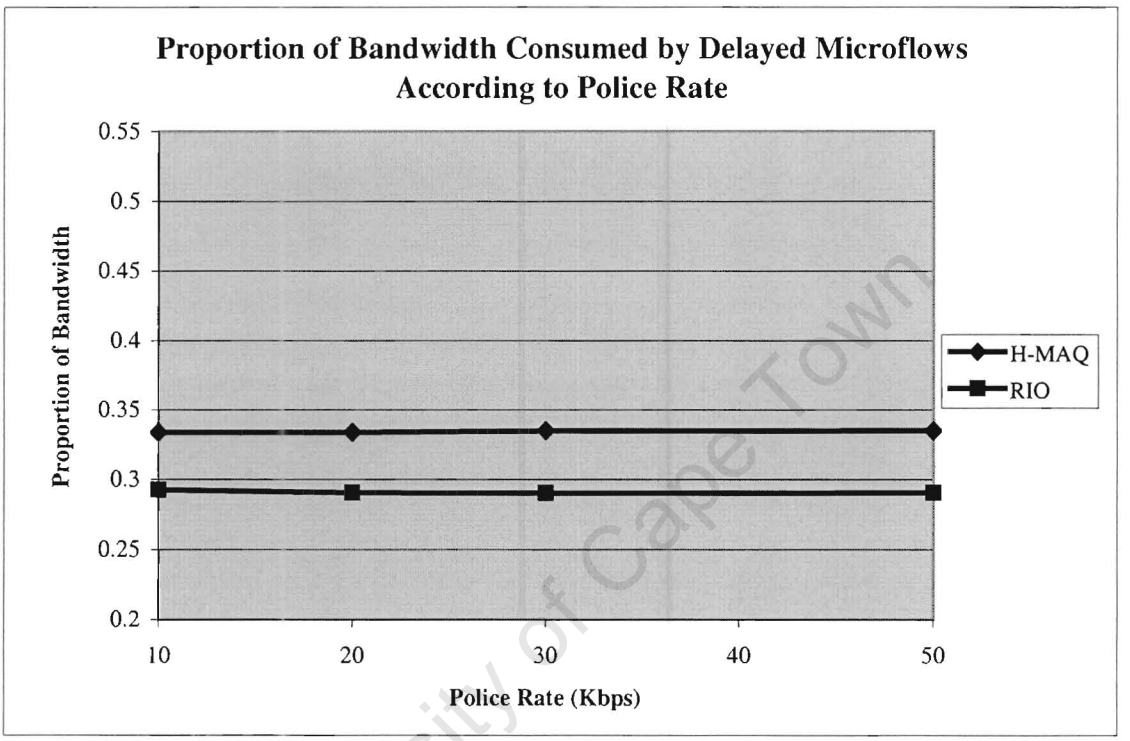

Figure 24: The proportion of bandwidth consumed by the delayed sources according to police rate when the target rate is 50 Kbps and the delay is 200ms

Figure 24 shows the results of 8 experiments where the target rate was set to 50 Kbps and the delay for half of the microflows was set to 200ms, but both the police rate and the core router mechanism were varied. The resultant series are flat graphs for both RIO and H-MAQ.

7.2.3 Analysis of Results

Because half of the TCP microflows were delayed in the experiments, the delayed sources should ideally consume half of the used bandwidth. The delayed sources tended to receive less than this proportion due to the fact that having long round trip times reduces the rate at which TCP sources ramp up their sending rates. Figure 23

shows that for both RIO and H-MAQ, the longer the round trip time of the delayed microflows, the less bandwidth was consumed by these sources.

A further observation is that as the target rates increase, the fairness decreases. This phenomenon is explained as follows: In Figure 23, when the target rates were only 5 Kbps, TCP flow-control did not limit sending rates. As the target rates increased, the inability of the delayed sources to ramp up their sending rates proportionately became a limiting factor. This trend continued towards the right-hand side of the graph where the delayed sources were at the greatest disadvantage.

Figure 23 shows that H-MAQ provided fairness that varied from being equal to that of RIO to being 18% better than RIO.

Figure 24 shows that varying the police rate had no appreciable effect on the relative throughput of the sources that varied in terms of their round trip times. This held true for both RIO and H-MAQ.

7.3 Aggregate Fairness and Control – Transport Protocol

Following is a comparison between the performance of a Diffserv network that has RIO in its core and that of a Diffserv network with H-MAQ implemented in the core, using aggregate fairness and control according to transport protocol as its performance metric. This round of evaluations emulates the scenario when two competing aggregate clients of a Diffserv network are identical in terms of SLA and number of microflows, but whilst one client sends predominantly TCP traffic, the other sends predominantly UDP traffic.

7.3.1 Evaluation

In this round of evaluations, 300 microflow traffic sources were started. Of these, half belonged to Aggregate 1 and half belonged to Aggregate 2. Aggregate 1's sources all used TCP and Aggregate 2's sources all used UDP. The sources all started between 0 and 2 seconds. They sent packets through the network for 100 seconds. Data was collected between 10 and 90 seconds. The target rate of the sources and the police rate

were as described in the Results section. In the case of H-MAQ, each aggregate was explicitly allocated half of the available bandwidth.

7.3.2 Results

Figure 25 gives the results of the evaluation that considers aggregate fairness according to transport protocol.

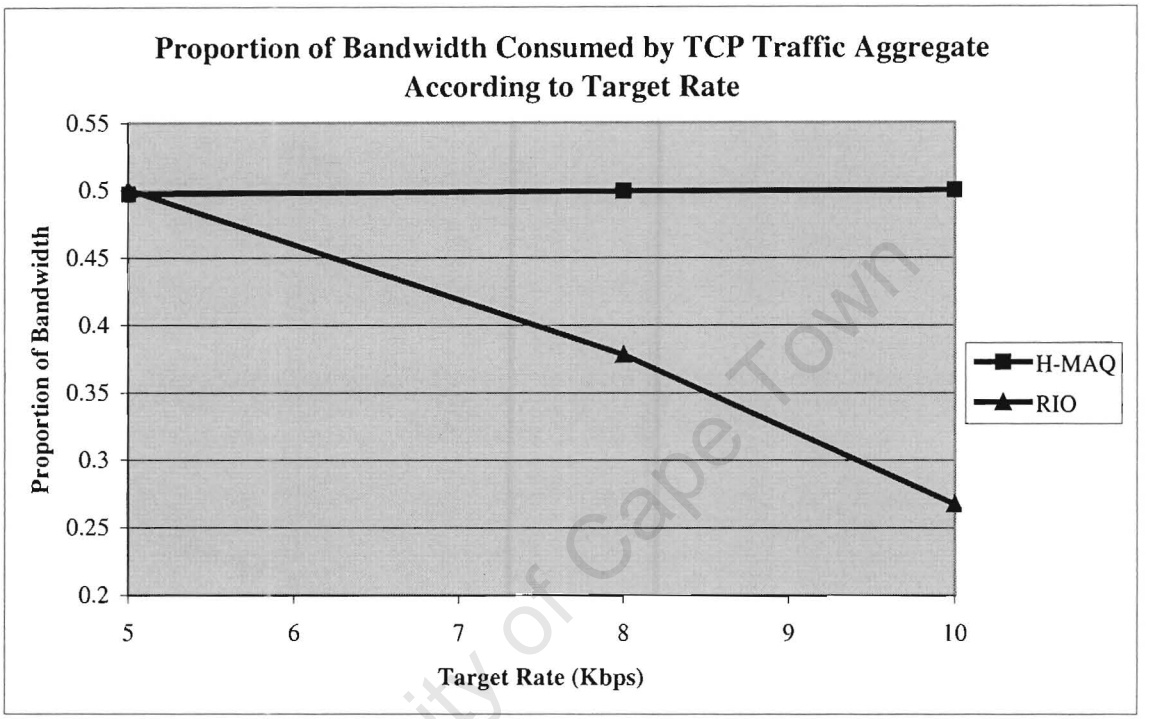

Figure 25: The proportion of bandwidth consumed by the TCP aggregate according to target rate when all packets are in-profile

Figure 25 shows the proportion of used bandwidth consumed by the TCP aggregate when all packets are marked as being in-profile. When the target rate was 5 Kbps, 50% of the throughput was from the TCP aggregate. This was so for both RIO and H-MAQ. In the case of RIO, as the target rates increased the proportion of TCP aggregate traffic decreased linearly. In the case of H-MAQ, increases in the target rates had no effect on the proportion of TCP aggregate traffic. This remained at a constant 50%.

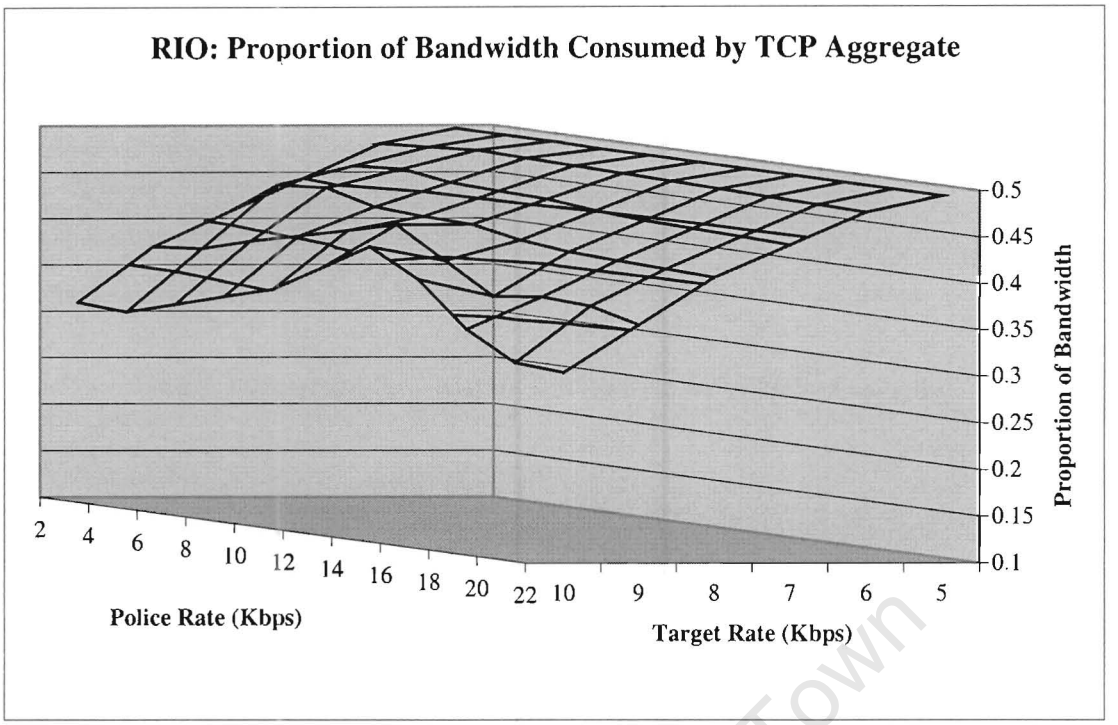

Figure 26: The proportion of bandwidth consumed by the TCP aggregate according to police and target rates when RIO is used in the network core

Figure 26 shows the proportion of used bandwidth consumed by the TCP aggregate when RIO is implemented in the core of the network. This figure shows how this proportion varies according to both the target rate and the police rate. The graph shows that although the TCP aggregate's proportion of bandwidth decreases with increased target rates, there is an optimal police rate that increases the TCP aggregate's proportion. In the case when the target rate is 10 Kbps, the proportion of bandwidth consumed by the TCP aggregate levels off at 30% but peaks at 41% where the police rate is 14 Kbps.

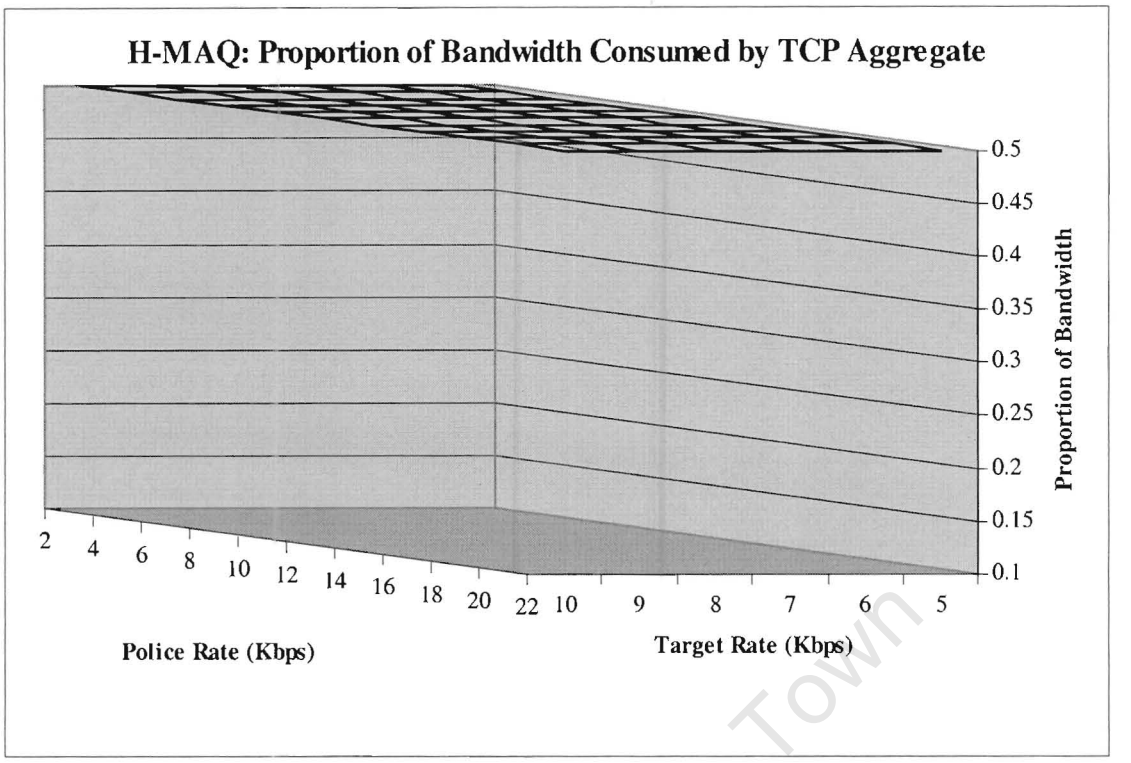

Figure 27: The proportion of bandwidth consumed by the TCP aggregate according to police and target rates when H-MAQ is used in the network core

Figure 27 shows the results for the same evaluation as before, except that in this case the core router implemented H-MAQ. This graph shows that for all police rates and target rates, the TCP aggregate received a constant 50% of the used bandwidth.

7.3.3 Analysis of Results

Because the aggregates are identical in terms of their SLA, they should each consume half of the used bandwidth. In Figure 25, when the target rate was only 5 Kbps, the TCP aggregate received 50% of the bandwidth. This is because there was little contention within the network and the microflow sources were all able to send packets at their target rates. In the case where RIO was implemented in the core, as the target rate increased, the proportion of bandwidth consumed by the TCP aggregate decreased linearly. This decrease signified degradation in the network's performance. In the case of H-MAQ, the TCP aggregate consumed a fixed 50% for all target rates. This demonstrated that the fixed allocation of resources to the aggregates was effective.

Figure 26 demonstrates that when the RIO network is congested, aggregate fairness may be improved by applying effective policing. The "hump" in the graph is explained as follows: When the police rate was low, all packets were out-of-profile and where thus treated identically. When the police rate was high, all packets were in-profile and were still treated identically. When police rates were similar to the target rates, the situation arose where most UDP packets were out-of-profile and most TCP packets were in-profile. This is because TCP's flow-control was causing the TCP packets to be preferentially marked. It was thus shown that RIO's performance can be improved by up to 37% when effective policing and marking is performed at the network edge. But even so, RIO's performance continued to degrade with increased target rates.

By contrast, Figure 27 demonstrates that when H-MAQ was used, all combinations of police rates and target rates resulted in the TCP aggregate consuming 50% of the used bandwidth.

In summary, although effective policing was shown to improve RIO's performance, this was still shown to degrade consistently with increased target rates. The network that had H-MAQ in its core was shown to be fair in all cases and thus demonstrated the precise controllability of H-MAQ networks.

7.4 Aggregate Fairness and Control – Round Trip Times

Following is a comparison between the performance of a Diffserv network that has RIO in its core and that of a Diffserv network with H-MAQ implemented in the core. This section uses aggregate fairness and control according to round trip time as its performance metric. This round of evaluations compares the throughput for two identical aggregates that have different round trip times on their TCP flows. The situation being emulated is one where a Diffserv network has two competing aggregate clients. The clients are identical in terms of their SLA and number of microflows, but they differ in that one client is sending TCP traffic to a nearby branch, and the other is sending TCP traffic to a branch further away.

7.4.1 Evaluation

In this round of evaluations, 300 TCP traffic sources were started, half belonging to Aggregate 1 and half belonging to Aggregate 2. Aggregate 1's packets were subject to a delay on the return path. The duration of this delay was determined by the experiment. The sources all started between 0 and 2 seconds. They sent packets through the network for 100 seconds. Data was collected between 10 and 90 seconds. The target rate of the sources was as described in the Results section as was the police rate. In the case of H-MAQ, each aggregate was explicitly allocated a half of the available bandwidth.

7.4.2 Results

Figure 28 gives the results of the evaluation that considers aggregate fairness when one of the aggregates is subject to a greater round trip time.

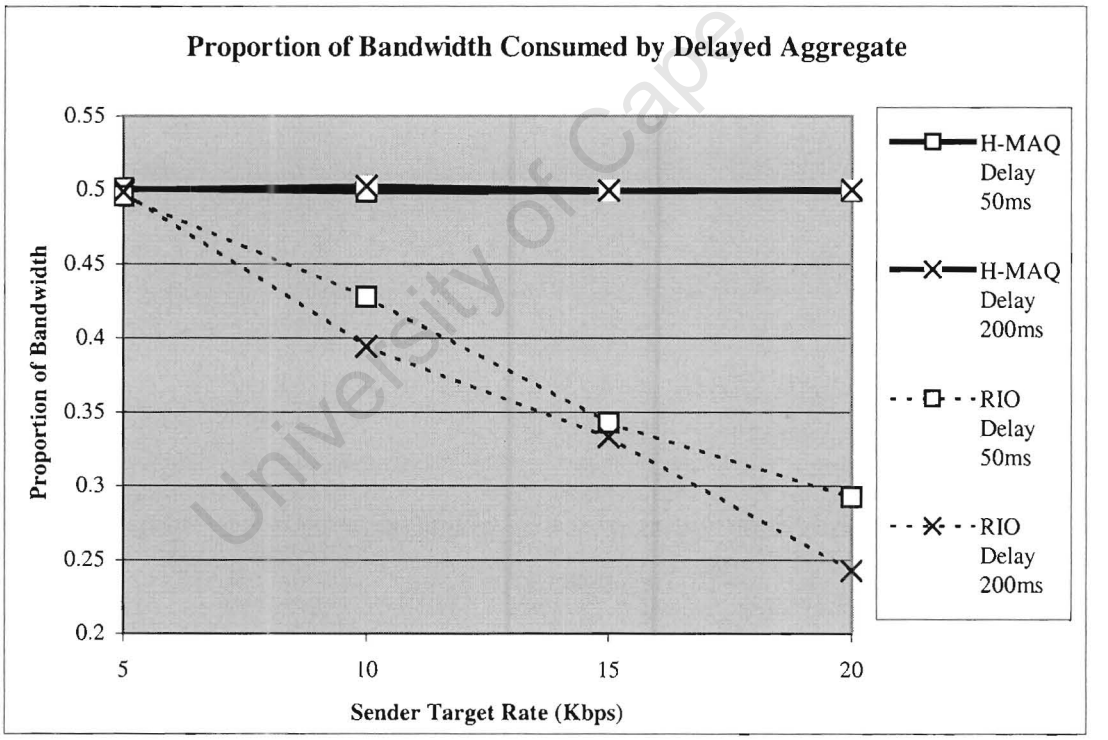

Figure 28: The proportion of bandwidth consumed by the delayed aggregate according to target rate when all packets are in-profile

Figure 28 shows the proportion of used bandwidth consumed by the delayed aggregate when all packets are marked as being in-profile. When the sender target rate was 5 Kbps, the delayed aggregate consumed 50% of the bandwidth. This was so for both RIO and H-MAQ and applied to the 50ms and 200ms delay times. In the case of

RIO, as the target rate increased the proportion of TCP aggregate traffic decreased linearly. Furthermore, increases in the delay time resulted in further decreases to the delayed aggregate's proportion of bandwidth. In the case of H-MAQ, increases in the target rate had no effect on the proportion of TCP aggregate traffic. This was true for both the 50ms and 200ms delay. The proportion of bandwidth for the delayed aggregate remained a constant 50%.

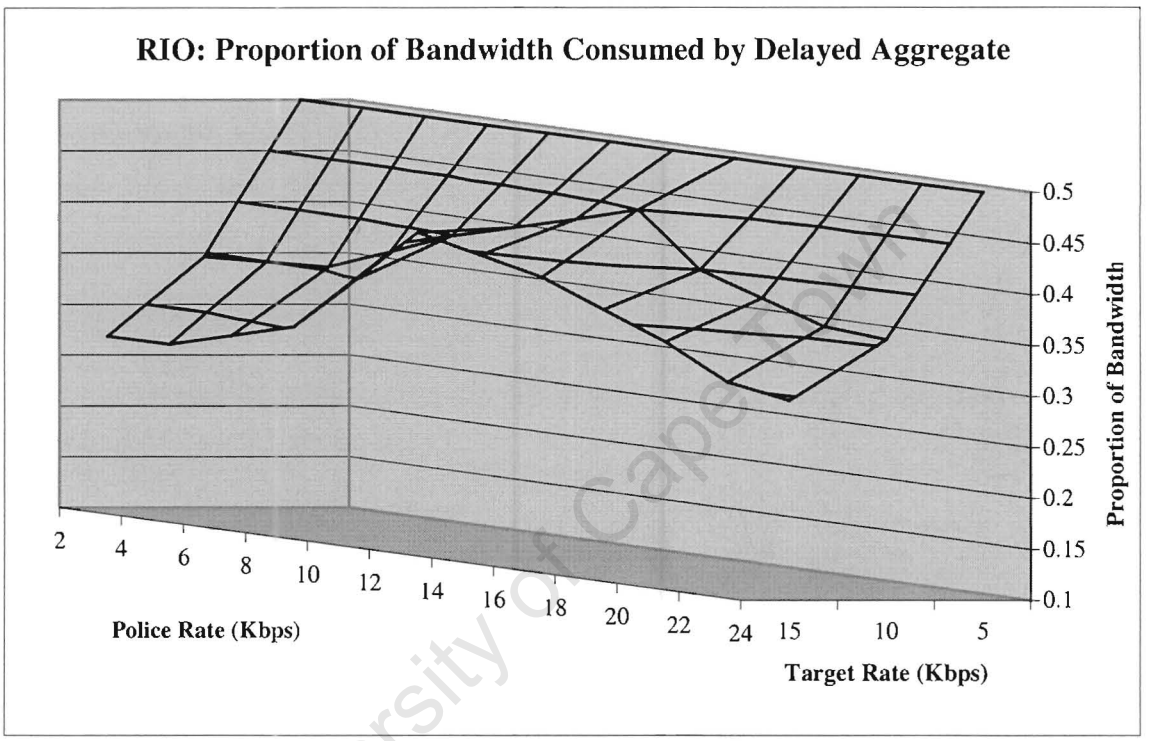

Figure 29: The proportion of bandwidth consumed by the delayed aggregate according to target and police rates when the delay is approximately 50ms and RIO is used

Figure 29 shows the proportion of used bandwidth consumed by the delayed aggregate when RIO is implemented in the core of the network and Aggregate 1 is subject to an additional delay of 50ms. Figure 29 shows how the delayed aggregate's proportion of bandwidth consumed varies according to both the target rate and the police rate. The graph also shows that although the delayed aggregate's proportion of bandwidth decreases with increased target rates, there is an optimal police rate, which increases the delayed aggregate's proportion of bandwidth. For example, when the target rate is 15 Kbps, the proportion of bandwidth consumed by the delayed aggregate levels off at approximately 28% but peaks at 41% where the police rate is 12 Kbps.

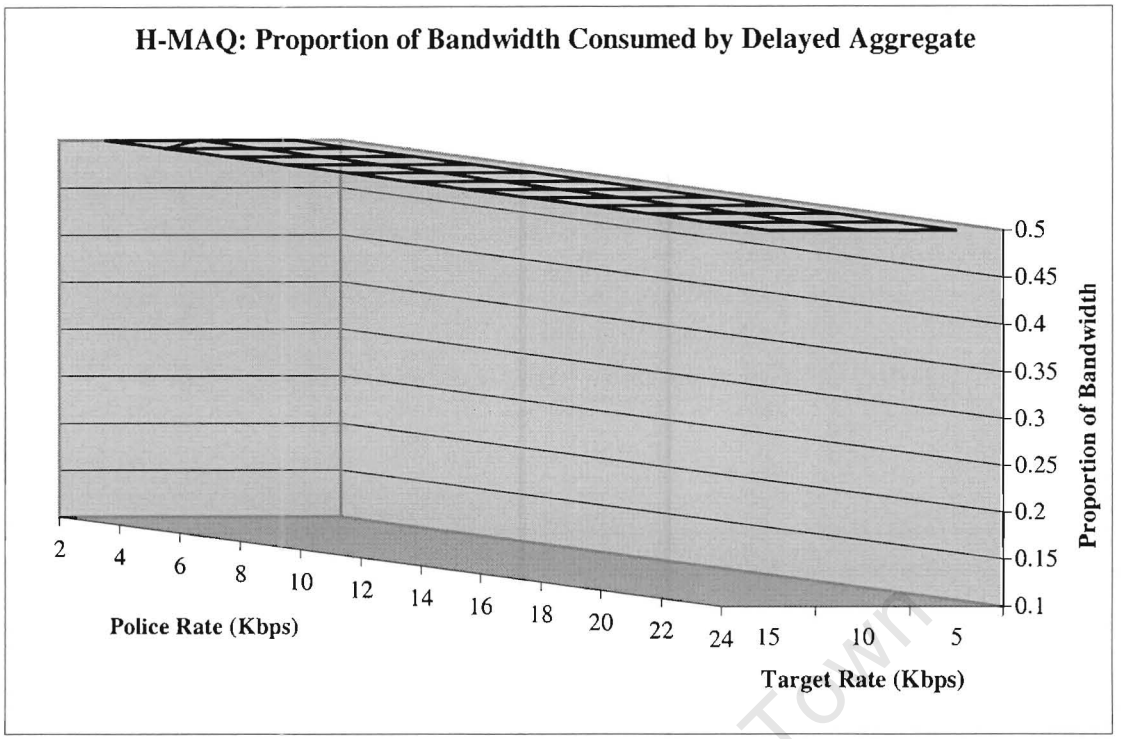

Figure 30: The proportion of bandwidth consumed by the delayed aggregate according to target and police rates when the delay is approximately 50ms and H-MAQ is used

Figure 30 shows the results for the same evaluation as before, except that in this case the core router implemented H-MAQ. This graph shows that for all police rates and target rates, the delayed aggregate received a constant 50% of used bandwidth.

7.4.3 Analysis of Results

Because the aggregates are identical in terms of their SLA, they should both receive 50% of the used bandwidth. On the far left-hand side of Figure 28, the delayed aggregate received 50% of the bandwidth. This was so because there was no contention within the network. The microflow sources were able to send packets at their target rates without being hindered by TCP flow-control or packet losses. In the case where RIO was implemented in the core, as the target rate increased, the proportion of traffic consumed by the delayed aggregate decreased linearly. Furthermore, increased delays resulted in still lower fairness for the delayed aggregate. This decrease signified degradation in the network's performance. In the case of H-MAQ, the TCP aggregate received a fixed 50% for all tested target rates

and delays. This demonstrated that when H-MAQ was used, the fixed allocation of resources to aggregates was effective.

Figure 29 demonstrates that when the RIO network is congested, aggregate fairness may be improved by applying effective policing. The "hump" in the graph is explained as follows: When the police rate was low, all packets were out-of-profile and where thus treated identically. When the police rate was high, all packets were in-profile and were still treated identically. When police rates were similar to target rates, the situation arose where most delayed packets were in-profile and most other packets were out-of-profile. This is because TCP's flow-control was causing the delayed packets to be preferentially marked. It was thus shown that RIO's performance can be improved by up to 52% when effective policing and marking is performed at the network edge. But even so, RIO's performance continued to degrade with increased target rates. Furthermore, perfect fairness could never be achieved by RIO under these circumstances because should the delayed sources indeed achieve their allocated 50% bandwidth consumption, the edge policing and remarking mechanism would no longer provide them with an advantage and the delayed aggregate's consumption would once again drop.

By contrast, Figure 30 demonstrates that when H-MAQ was used, all combinations of police rates and target rates resulted in the TCP aggregate receiving 50% of the used bandwidth.

In summary, although effective policing was shown to improve RIO's performance, RIO's performance was still shown to degrade consistently with increased target rates. The network that had H-MAQ in its core was shown to be fair under all tested conditions and thus demonstrated the precise controllability of H-MAQ networks.

7.5 Aggregate Fairness and Control – Aggregate Service Level

Following is a comparison between the performance of a Diffserv network that has RIO in its core and that of a Diffserv network with H-MAQ implemented in the core, using aggregate fairness and control as its performance metric. This round of

evaluations compares the throughput for two aggregates. One of the aggregates has a SLA that allows it to send a quarter of the total traffic. The other aggregate's SLA allows it to send three quarters of the total traffic. Both aggregates have the same number of traffic sources. The emulation thus reflects the scenario where a Diffserv network has two competing aggregate clients. Both clients are the same size, but one of the clients has subscribed to a cheaper service option.

7.5.1 Evaluation

In this round of evaluations, between 50 and 200 traffic sources were started. Of these, half belonged to Aggregate 1 and half belonged to Aggregate 2. For each experiment, the target rate for all sources was set to 20 Kbps. The police rate for the traffic sources belonging to the aggregate with the higher service level was set to 20 Kbps. The police rate for the aggregate traffic sources belonging to the aggregate with the lower service level was set to 6.667 Kbps. The police rates were thus equivalent to that which a commercial Diffserv network would use to control the aggregates' utilization of the network according to their SLAs. The network's performance was evaluated for both TCP and UDP traffic. The sources all started between 0 and 2 seconds. They sent packets through the network for 100 seconds. Data was collected between 10 and 90 seconds. In the case of H-MAQ, the smaller aggregate was explicitly allocated a quarter of the available bandwidth. The larger aggregate was explicitly allocated the remaining three-quarters.

7.5.2 Results

Figure 31 gives the results of the evaluation that considers aggregate fairness when one of the aggregates has a SLA which allows it to consume one quarter of the total bandwidth.

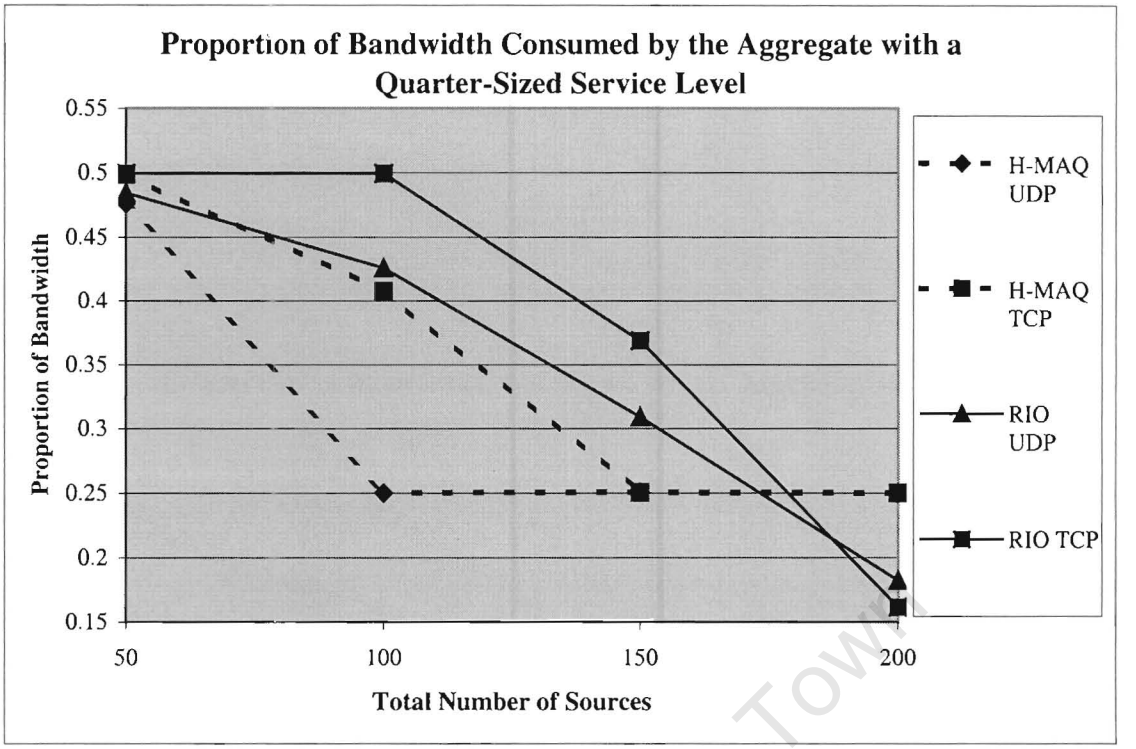

Figure 31: The proportion of bandwidth consumed by the aggregate with a quarter-sized service level according to the total number of TCP sources

Figure 31 shows that when there were only 50 sources, the aggregate with the lower service level consumed approximately half of the used bandwidth. In the case of both RIO TCP and RIO UDP, as the number of sources increased, the proportion of bandwidth consumed by the aggregate with the lower service level decreased slowly. This rate of decrease accelerated after 100 sources so that when there were 200 sources, the aggregate with the lower service level received less than 18% of the used bandwidth.

In the case of H-MAQ, as the number of sources increased, the proportion of bandwidth consumed by the aggregate with the lower service level decreased. This trend continued until the aggregate with the lower service level had reached one quarter of the total bandwidth. This point was reached at 100 sources for UDP and 150 sources for TCP. Beyond this point, the proportion of bandwidth consumed by the aggregate with the lower service level remained fixed at one quarter.

7.5.3 Analysis of Results

Because the aggregate with the lower service level has a SLA that entitles it to a quarter of the total bandwidth, it should indeed consume that proportion of the used bandwidth. Similarly, the aggregate with the higher service level should receive three quarters of the bandwidth. That is, of course, contingent on its traffic sources having sufficient packets to send. Figure 31 shows the proportion of bandwidth received by that aggregate for an increasing number of sources. Because the police rate was unaltered as the number of sources increased, what this graph demonstrates is the changing proportion of bandwidth received by the aggregate with the lower service level as network congestion increases. When there were only 50 sources, there was no congestion in the network. Because no packets were dropped and target rates were reachable, each source was able to send packets at its target rate. The aggregates' consumption of bandwidth was thus proportional to the number of microflows in each aggregate.

In the case of the RIO UDP and RIO TCP series, the increasingly negative slope reflects increasing network congestion. This is explained as follows: Under all network congestion levels, more of the packets belonging to the aggregate with the lower service level were marked as being out-of-profile. When there was little congestion, as was the case when there were 50 sources, few packets were dropped, and so the fact that the aggregate with the lower service level was more strictly policed was of little consequence. As the number of sources increased, so too did congestion levels. This resulted in more packets being dropped. Because the aggregate with the lower service level had more out-of-profile packets in the network, a higher proportion of that aggregate's packets were dropped. The aggregate with the lower service level was thus over-penalized. The steep slope of the RIO UDP and RIO TCP series at the point where they intersect the "0.25 proportion of bandwidth" line indicates the rareness of the conditions under which Diffserv networks with RIO cores provide adequate fairness. Furthermore, it is not possible to control the operating point for Diffserv networks with regards to their position on the x axis of Figure 31 as this is governed by network congestion levels.

In the case of H-MAQ UDP and H-MAQ TCP, the proportion of bandwidth consumed by the aggregate with the lower service level converged to one quarter as the number of sources increased. On the left-hand side of these convergence points, the fact that the aggregate with the lower service level received more than a quarter of the bandwidth reflects the fact that at this low utilization level, the aggregate with the higher service level was not able to consume all of the resources that had been allocated to it. These resources were thus unused or were handed over to the other aggregate. The fact that the UDP sources converged before the TCP sources reflects the fact that UDP sources consume bandwidth more aggressively, due to having no transport layer flow-control.

In summary, the network with the RIO core was shown to be ineffective in providing services to aggregates in proportion to their SLAs, and the network with the H-MAQ core was found to provide precise fairness and controllability with regards to resource allocation.

7.6 Aggregate Fairness and Control – Number of Flows Per Aggregate

Following is a comparison between the performances of a Diffserv network that has RIO in its core and that when H-MAQ is implemented in the core using aggregate fairness and control as its performance metric. This round of evaluations compares the throughput of two aggregates. Both aggregates have identical SLAs. They do, however, differ in that one of the aggregates has one third as many traffic sources as the other. The emulation thus reflects the scenario where a Diffserv network has two competing aggregate clients. Both clients have the same SLA, but one client has twice as many end-users as the other.

7.6.1 Evaluation

In this round of evaluations, 400 traffic sources were started. Of these, 133 belonged to Aggregate 1 and 267 belonged to Aggregate 2. For each experiment, the target rate for all sources was set to 15 Kbps. The police rates were varied with the constraint that the product of the number of sources for Aggregate 1 and the police rate for Aggregate 1 was equal to the product of the number of sources for Aggregate 2 and

the police rate for Aggregate 2. The police rates were thus equivalent to those that a commercial Diffserv network would use to control aggregates' utilization of the network according to their SLAs. The network performance was evaluated for both TCP and UDP traffic. The sources all started between 0 and 2 seconds. They sent packets through the network for 100 seconds. Data was collected between 10 and 90 seconds. In the case of H-MAQ, each aggregate was explicitly allocated half of the available bandwidth.

7.6.2 Results

Figure 32 gives the results of the evaluation that considers aggregate fairness when one of the aggregates has twice as many microflow traffic sources as the other.

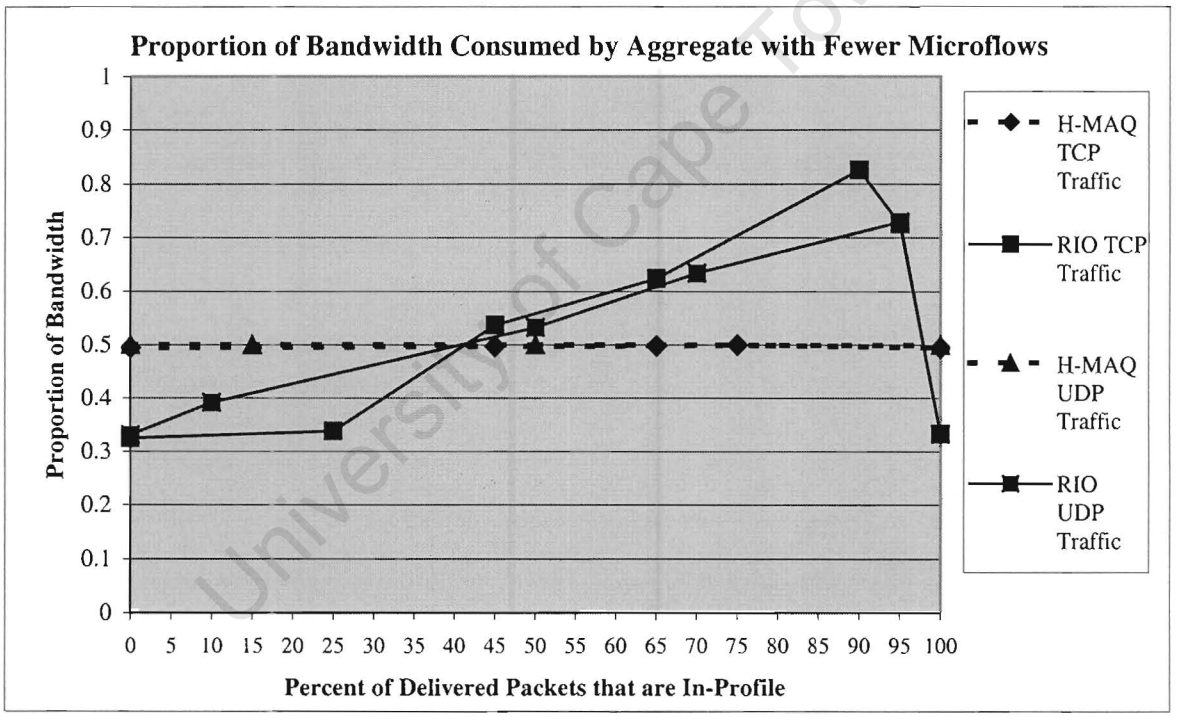

Figure 32: The proportion of bandwidth consumed by the aggregate with fewer microflows according to the percent of delivered packets that are in-profile

Figure 32 shows that in the case of the RIO UDP and RIO TCP series, when the percent of delivered packets that were in-profile was either 0 or 100 (i.e. all delivered packets were out-of-profile or in-profile respectively) the aggregate with fewer sources consumed 33% of the used bandwidth. Between 45 and 95 on the x axis, the aggregate with fewer sources consumed more than half of the used bandwidth.

In the case of both H-MAQ UDP and H-MAQ TCP, each aggregate consumed 50% of the used bandwidth for all x axis values.

7.6.3 Analysis of Results

Because the aggregates are identical in terms of their SLA, they should both receive 50% of the bandwidth. In the case of the RIO UDP and RIO TCP series, when all packets were marked the same, as was the case on the far left and right-hand sides of the graph, the aggregate with a third of the microflows consumed that proportion of the used bandwidth. This is the part of the graph where policing was indiscriminate and all packets were treated identically. Each aggregate thus consumed bandwidth in proportion to its number of constituent sources. Between 45 and 95 on the x axis, the aggregate with fewer sources consumed more than half of the used bandwidth. This is explained as follows: Because Aggregate 1 had fewer sources, each of its traffic sources was allowed a higher data rate before being marked as out-of-profile. Between 45 and 95 on the x axis, most Aggregate 1 packets that were sent were in-profile and most Aggregate 2 packets that were sent were out-of-profile. The policing mechanism was thus shown to be too effective at putting Aggregate 1 at an advantage. This was unfair to Aggregate 2. On the x axis, as one moves from 45 to 0, the proportion of Aggregate 1 packets that were in-profile also decreased. This evened out the treatment that Aggregate 1 and Aggregate 2 packets received. Although there were two instantaneous points at which the RIO network gave 50% of used bandwidth to both aggregates, the percentage of packets that are in-profile at any given time cannot be kept constant as this is governed by network utilization levels.

In the case of H-MAQ UDP and H-MAQ TCP, the fact that the each aggregate always consumed 50% of the bandwidth indicates the high level of fairness and controllability of Diffserv networks with H-MAQ cores.

In summary, a Diffserv network with an H-MAQ core was shown to provide precise fairness and controllability to aggregates with differing numbers of microflows for both TCP and UPD traffic. This is in stark contrast to the case of a RIO network core where poor fairness was prevalent.

Chapter 8

Conclusions

From the results of Chapter 7, a number of conclusions may be drawn regarding the performances of Diffserv networks with RIO and H-MAQ network cores. These conclusions are presented below in two sections: microflow fairness, and aggregate fairness and control. Finally, the overall performance of the proposed solution is considered in light of the previous sections.

8.1 Microflow Fairness

The problem statement of this thesis noted that networks with RIO cores have been shown to provide poor microflow fairness when competing microflows differ in terms of their transport protocol and round trip time. An alternative mechanism, namely H-MAQ, was proposed, specified and implemented. The per-microflow fairness of a Diffserv network with H-MAQ in its core was compared with that where RIO was used in its core. The following points assess the relative performance of the two mechanisms.

- When TCP traffic sources compete for resources against UDP traffic sources, networks with a RIO core were shown to be slightly fairer than networks with an H-MAQ core.

- When TCP sources with long round trip times compete for resources against TCP sources with shorter round trip times, networks with an H-MAQ core were shown to provide equal to better performance than networks with a RIO core.

8.2 Aggregate Fairness and Control

The problem statement of this thesis noted that it is not possible to adequately control resource allocation in Diffserv networks when RIO is used in core network elements. An alternative mechanism, namely H-MAQ, was thus proposed, specified and implemented. This mechanism permits precise resource allocation at an aggregate level. H-MAQ's performance was compared to that of RIO for a number of cases where aggregates were competing for resources. The following points describe the performance of the two mechanisms for different network conditions.

- When aggregates with TCP traffic sources compete for resources against aggregates with UDP sources, the networks with H-MAQ cores were found to provide fair and controllable resource allocation according to the aggregates' SLAs. This was in contrast to the networks with RIO cores, which are unfair to aggregates with TCP sources.
- When TCP aggregates with long round trip times compete for resources against aggregates with shorter round trip times, the networks with H-MAQ cores were found to provide fair and controllable resource allocation according to the aggregates' SLAs. This is in contrast to the networks with RIO cores, which are unfair to the aggregates with long round trip times.
- When aggregates with smaller SLAs compete for resources against aggregates with larger SLAs, the networks with H-MAQ cores were found to be effective in providing precise resource allocation according to the aggregates' SLAs. This was in contrast to networks with RIO cores, which can be unfair to aggregates of both smaller and larger SLAs depending on network congestion levels. This holds true for both TCP and UDP traffic.
- When aggregates with many traffic sources compete for resources against aggregates with fewer traffic sources, networks with H-MAQ cores were found to provide fair and controllable resource allocation according to the aggregates' SLAs. This is in contrast to networks with a RIO core, which can be unfair to

either aggregate, depending on network utilization levels. This holds true for both TCP and UDP traffic.

8.3 Overall Performance of H-MAQ

At the microflow level, the performance of Diffserv networks with H-MAQ cores was shown to be comparable to that of Diffserv networks with RIO cores. At the aggregate level, networks with H-MAQ cores were shown to be capable of providing exact control of resource allocation. Being able to provide this high level of network control means that should H-MAQ be used in Diffserv AF networks instead of RIO, over-provisioning would no longer be necessary to provide guaranteed services to aggregates.

Chapter 9

Recommendations and Future Work

There is scope for research aimed at improving the fairness of H-MAQ at the microflow level. This work should consider alternative drop policies as well as the feasibility and performance of a system whereby a separate RIO queue is allocated to each aggregate.

In the H-MAQ core, when an out-of-profile packet needs to be dropped from a microflow queue, a search is performed to find the out-of-profile packet that is closest to the front of the queue. This packet is then dropped. The author suggests that a modification to the H-MAQ mechanism be investigated which causes the packet at the front of the queue to be dropped. This is regardless of whether it is in- or out-of-profile. Because all packets in a given microflow queue are from the same traffic source, it should not negatively affect that microflow source if an in-profile packet were dropped instead of an out–of-profile packet. The benefits of this modification would be as follows:

- Dropping the packet that is at the front of the queue rather than doing a search to find the first out-of-profile packet would result in a more computationally efficient algorithm.
- Because the dropped packet would always be the one at the front of the queue, the traffic source would be informed of the packet loss sooner. This would improve TCP responsiveness.

The effect of different packet marking policies needs to be investigated, this is especially necessary when microflow and aggregate-unaware mechanisms such as RIO are used in the network core. In particular, there should be investigations into the performance of Diffserv networks when marking mechanisms mark only surplus packets as being out-of-profile rather than marking all packets as being out-of-profile when microflow sources exceed their police rate.

Appendices

Appendix A: Asynchronous Transfer Mode

This appendix introduces ATM as well as well as describing its use in carrying IP traffic.

A.1 Introduction to ATM

Asynchronous Transfer Mode (ATM) is a networking technology based on asynchronously switching small 53 Byte cells through networks. ATM is connection orientated, which means that an explicit connection setup is required before data may be sent across an ATM network. Because ATM was designed to support real-time traffic such as voice and video, it uses sophisticated congestion control mechanisms that interleave data traffic with control cells. These congestion control mechanisms are able to limit traffic on a per-connection basis as well as a per-aggregate basis.

ATM data links are divided into a number of virtual paths, each of which is divided into a number of virtual channels. This configuration enables network elements to easily perform shaping or policing on paths or on individual channels. A Virtual Channel Identifier (VCI) together with a Virtual Path Identifier (VPI) may be used to uniquely identify a virtual channel along a given link. A permanent Virtual Circuit

(PVC) is a permanent connection between two ATM end systems that uses a fixed VPI/VCI.

A.2 IP Over ATM

ATM is implemented extensively throughout the Internet. This is done using the IP over ATM architecture. IP over ATM works as follows: IP packets entering the ATM network are broken into a number of ATM cells. The cells are sent through the ATM network. At the egress of the ATM network, the IP packets are re-assembled whereupon they can continue towards their destination.

Appendix B: RIO Development and Testing

Because the author's implementation of RIO was to be used as the benchmark against which the proposed mechanism was to be compared, it was necessary that the RIO implementation be correct. Note that although the proposed mechanism was tested in a similar way to that of RIO, its verification is not be presented in a separate appendix as the results of this study adequately demonstrate its correctness.

B.1 RIO Development

The RIO mechanism was implemented using a simple FIFO drop-tail queue as a template. This kernel module was written by Fred Kuhn of Washington University's Applied Research Laboratory. Using a drop-tail module as a template ensured that the basic buffer management operations such as initialising a packet scheduler and dropping an arriving packet had already been verified. The RIO module was implemented according to pseudo-code obtained from the relevant papers [25,28].

B.2 RIO Testing

The first stage of testing consisted of a number of complete code walkthroughs. During this process, each line was critically evaluated in the context of the program. Once this was complete, the second stage of testing began. During this stage, the author performed real-time monitoring of the variables that formed part of the RIO mechanism. This took place whilst packets were moving through the router. These variables were outputted to a terminal window by the SPCs on with the RIO kernel modules were run. Only once the author was satisfied that RIO's internal variables were correct did the final round of tests begin. The final round of tests was aimed at ensuring that RIO behaved as it was designed to.

Figure 33 not only demonstrates that the RIO implementation performed as expected, but also offers an insight into how the mechanism works. The graph shows the delay incurred from the time that each packet was transmitted until the time that it reached the receiver. This delay is a function of the time spent by packets in the buffers of the router. The network topology was as described in Section 6.3. Figure 33 compares the delay when a simple FIFO drop-tail queue was implemented at the router with the delay when a RIO queue was used. In the case of RIO, there is a further comparison between when the packets are policed as being in-profile versus when they are out-of-profile. This distinction is meaningless in the case of drop-tail as drop-tail ignores the DSCP of packets. In this experiment, 70 UDP traffic sources were started simultaneously. They each sent out 500 byte packets that had exponentially distributed inter-arrival times with an average data rate of 40 Kbps. This meant that a total of 2.8 Mbps were being sent. A maximum number of 200 packets were permitted in the MSR's output buffer. In the case of RIO, *ThreshIn*, *MaxThreshIn ThreshOut* and *MaxThreshOut* were set to 40, 80, 90 and 140 respectively. The router's output link data rate was throttled to approximately 160kbps to ensure congestion in the router's output buffer. Note that the delay incurred in the testbed router was far greater than that which would be permitted on commercial network.

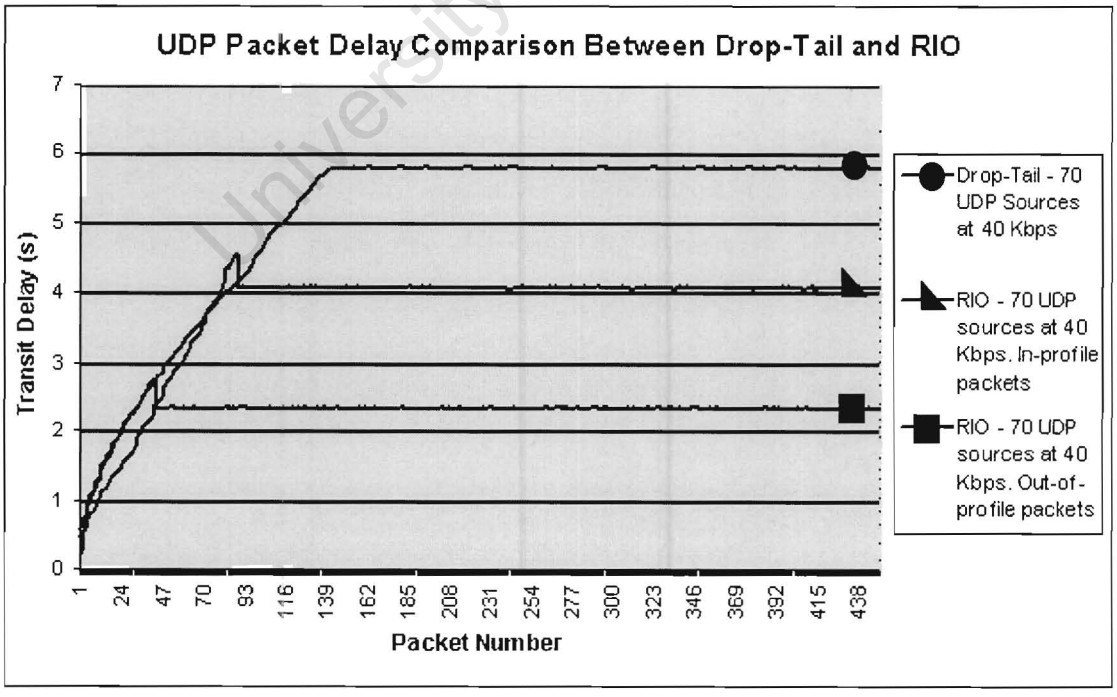

Figure 33: Relative queuing delay for RIO and drop-tail

Figure 33 shows that in the case of the drop-tail queue, the buffers filled up progressively at the start of the experiment and remained at that high level. In the case of RIO with in-profile packets, the buffers filled up as with drop-tail. They then stabilized at a lower level. In the case of RIO with out-of-profile packets, the buffers filled up as before, but stabilized at a still lower level. These three levels reflect the maximum number of packets allowed in the FIFO drop-tail queue, *MaxThreshIn* and *MaxThreshOut* respectively.

A further point that is of interest is that for both RIO data series, there was an initial overshoot before buffer occupancy levels stabilized. This overshoot was expected, as there is a lag before RIO's average buffer occupancy variables reach their maximum thresholds and reflect persistent congestion.

References

[1] Webopedia: "The 7 Layers of the OSI Model", [Online]. Available: webopedia.internet.com/quick_ref/OSI_Layers.asp, August 2003

[2] IETF Charter: "IP Version 6 Working Group", [Online]. Available: www.ietf.org/html.charters/ipv6-charter.html, August 2003

[3] S. Deering, R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1997

[4] A. Odlyzko, "Internet growth: Myth and reality, use and abuse", iMP: Information Impacts Magazine, November 2000

[5] TeleGeography, Inc. Press Release: "Global Internet Backbone Growth Slows Dramatically", [Online]. Available: www.telegeography.com/press/releases/2002/16-oct-2002.html, October 2002

[6] A. Nikologiannis, M. Katevenis, "Efficient Per-Flow Queuing in DRAM at OC-192 Line Rate using Out-of-Order Execution Techniques", Proc. of IEEE International Conference on Communications, Helsinki, June 2001

[7] RedHerring Magazine: "Explained: How Internet core routers deliver", [Online]. Available: www.redherring.com/mag/issue102/800020080.html, September 2001

[8] V. Kumar, T. Lakshman, D. Stiliadis, "Beyond Best Effort: Router Architectures for the Differentiated Services of Tomorrow's Internet", IEEE Communications Magazine, page 152, May 1998

[9] Light Reading: "Internet Core Router Test", [Online]. Available: https://www.juniper.net/products/features/core/core_router_test.pdf, March 2001

[10] K. Thompson, G. Miller, R. Wilder, "Wide-Area Internet Traffic Patterns and Characteristics (Extended version)", [Online]. Available: www.vbns.net/presentations/papers/MCItraffic.pdf, December 1997

[11] Caida: "Top applications (bytes) for Subinterface 0[0]: SD-NAP Traffic", [Online]. Available: www.caida.org/analysis/workload/byapplication/sdnap/, June 2002

[12] S. McCreary, K Claffy, "Trends in Wide Area IP Traffic Patterns", [Online]. Available: www.caida.org/outreach/papers/2000/AIX0005/AIX0005.html, September 2002

[13] Caida: "Packet Sizes and Sequencing", [Online]. Available: www.caida.org/outreach/resources/learn/packetsizes/, August 2002

[14] A. Broido, K Claffy, E.Nemeth, "Packet arrivals on rate-limited Internet links", [Online]. Available: www.caida.org/~broido/coral/packarr.html, November 2000

[15] T. Bonald, S. Oueslati-Boulahia, J. Roberts, "IP Traffic and QoS Control - Towards a Flow-aware Architecture", Proc of World Telecommunications Congress, Paris, September 2002

[16] IETF Charter: "Integrated Services Working Group", [Online]. Available: www.ietf.org/html.charters/intserv-charter.html, September 2000

[17] R. Braden, L. Zhang, S. Berson, S. Herzog, S. Jamin, "Resource ReSerVation Protocol (RSVP) - Version 1 Functional Specification", RFC 2205, September 1997

[18] IETF Charter: "Differentiated Services Working Group", [Online]. Available: www.ietf.org/html.charters/diffserv-charter.html, March 2002

[19] K. Nichols, S. Blake, F. Baker, D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, December 1998

[20] J. Heinanen, F. Baker, W. Weiss, J. Wroclawske, "Assured Forwarding PHB Group", RFC 2597, June 1999

[21] N. Seddigh, B. Nandy, P. Pieda, "Bandwidth Assurance Issues for TCP Flows in a Differentiated Services Network", Proc. of the IEEE GLOBECOM, Rio De Janeiro, page 6, December 1999

[22] K. Chan, R. Sahita, S. Hahn, K. McCloghrie, "Differentiated Services Quality of Service Policy Information Base", RFC 3317, March 2003

[23] N. Li, M. Borrego, S. Li, "Achieving per-flow fair rate allocation in Diffserv", ACM Transactions on Modelling and Computer Simulation, Volume 11(2), page 161, April 2001

[24] T. Ferrari, "End-To-End Performance Analysis with Traffic Aggregation", Proc. of TNC, Volume 34(6), page 905, May 2000

[25] D. Clark, W. Fang, "Explicit Allocation of Best Effort Packet Delivery Service", IEEE/ACM Transactions on Networking, Volume 6(4), August 1998

[26] R. Makkar, I. Lambadaris et al, "Empirical Study of Buffer Management Schemes for Diffserv Assured Forwarding PHB", Proc. of Ninth International Conference on Computer Communications and Networks, Las Vegas, October 2000

[27] J. Heinanen, F. Baker, W. Weiss, J. Wroclawske, "Assured Forwarding PHB Group", RFC 2597, June 1999

[28] S. Floyd, V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance", IEEE/ACM Transactions on Networking, Volume 1(4), page 397, August 1993

[29] M. El-Gendy, K. Shin, "Equation-Based Packet Marking for Assured Forwarding Services", IEEE INFOCOM, New York, 2002

[30] I. Yeom, A. Reddy, "Modelling TCP Behaviour in a Differentiated Services Network", Texas A&M University ECE Technical Report, May 1999

[31] S. Low, F. Paganini, J. Wang, S. Adakha, J. Doyle, "Dynamics of TCP/RED and a Scalable Control", IEEE INFOCOM, New York, June 2002.

[32] N. Christin, J. Liebeherr, T. Abdelzaher, "A Quantitative Assured Forwarding Service", Proc. of IEEE INFOCOM, Volume 2, page 864, New York, June 2002

[33] J. Ibanez, K. Nichols, "Preliminary Simulation Evaluation of an assured service", [Online]. Available: www.globecom.net/ietf/draft/draft-ibanez-diffserv-assured-eval-00.html, August 1998

[34] I. Stoica, H. Zhang, "Providing Guaranteed Services without Per-Flow Management", ACM SIGCOMM Computer Communication Review, Volume 29(2), page 81, October 1999

[35] H. Chow, A. Leon-Garcia, "A Feedback Control Extension to Differentiated Services", [Online]. Available: www.globecom.net/ietf/draft/draft-chow-diffserv-fbctrl-00.html, March 1999

[36] L. Breslau, E. Knightly, S. Shenker, I. Stoica, H. Zhang, "Endpoint Admission Control: Architectural Issues and Performance", Proc. of ACM SIGCOMM 2000, Stockholm, August 2000

[37] A. Habib, B. Bhargava, "Unresponsive Flow Detection and Control Using the Differentiated Services Framework", [Online]. Available: citeseer.nj.nec.com/habib01unresponsive.html, 2001

[38] I. Andrikopoulos, L. Wood, G. Pavlou, "A Fair Traffic Conditioner for the Assured Service in a Differentiated Services Internet", Proc. of the IEEE International Conference on Communication, Volume 2, page 806, New Orleans, June 2000

[39] B. Suter, T. Lakshman, D. Stiliadis, A. Choudhury, "Buffer Management Schemes for Supporting TCP in Gigabit Routers with Per-Flow Queuing", IEEE Journals in Selected Areas in Communications, Volume 17(6), August 1999

[40] W. Eddy, M. Allman, "A Comparison of RED's Byte and Packet Modes", Computer Networks, Volume 42(2), June 2003

[41] Y. Bernet, P. Ford, R. Yavatkar, F. Baker, L. Zhang, M. Speer, R. Braden, B. Davie, J. Wroclawski, E. Felstaine, "A Framework for Integrated Services Operation over Diffserv Networks", RFC 2998, November 2000

[42] ClearSpeed White Paper: "Layer 3/4" Classification and Routing", [Online]. Available: www.clearspeed.com/technology.php?wp, 2002

[43] Cisco White Paper: "The Evolution of High-End Router Architectures", [Online]. Available: www.cisco.com/warp/public/cc/pd/rt/12000/tech/ruar_wp.htm, January 2001

[44] EZchip press release: "EZchip Introduces a 10G/OC-192 Traffic Manager to Broaden its Next-generation Network Processor Architecture", [Online]. Available: www.ezchip.com/html/press_011022.html, October 2001

[45] J. Henry, E. McGinnis "Banking on the Internet", [Online]. Available: www.sagharbor.com/banking/banking-chapter%20one.htm

[46] Lucent Technologies Press Release: "Lucent Technologies Introduces New Multi-Terabit Switch Router for Small Regional Networks", [Online]. Available: www.lucent.com/press/0600/000606.nsc.html, June 2000

[47] ClearSpeed Reference Design: "A ClearConnect 40G Packet Processor", [Online]. Available: www.clearspeed.com/technology.php?wp, 2002

[48] Cable Datacom White Paper: "QoS: One HFC Network, Multiple Revenue Streams", [Online]. Available: www.cabledatacomnews.com/whitepapers/

[49] S. Choi, J. Dehart, R. Keller, F. Kuhns, J. Lockwood, P. Pappu, J. Parwatikar, W. Richard, E. Spitznagel, D. Taylor, J. Turner, K. Wong, "Design of a High

Performance Dynamically Extensible Router", Proc. DARPA Active Networks Conference Exposition, San Francisco, page 42, May 2002

[50] F. Kuhns, J. DeHart, R. Keller, J. Lockwood, P. Pappu, J. Parwatikar, E. Spitznagel, D. Richards, D. Taylor, J. Turner, K. Wong, "Implementation of an Open Multi-Service Router", Washington University in St. Louis, Technical Report, August 2001

[51] T. Chaney, J. Fingerhut, M. Flucke, J. Turner, "Design of a Gigabit ATM Switch", Proc. of INFOCOM, page 2, March 1997

[52] D. Taylor, J. Lockwood, T. Sproull, J. Turner, D. Parlour, "Scalable IP Lookup for Programmable Routers", IEEE INFOCOM, New York, June 2002

[53] J. Lockwood, D. Taylor, "Design and Implementation of a Field Programmable Port Extender", [Online]. Available: www.arl.wustl.edu/gigabitkits/Workshop_00_01/slides/fpx-workshop_011000_lockwood.pdf, January 2001

[54] W. Eatherton, "Hardware-Based Internet Protocol Prefix Lookups" MS thesis, Washington University in St. Louis, May 1999

[55] J. DeHart, "A Smart Port Card Tutorial", [Online]. Available: www.arl.wustl.edu/projects/gigabitkits/Workshop_00_07/spc_tutorial/slides/spchw.pdf, July 2000

[56] Z. Dittia, J. Cox, G. Parulkar, "Design of the APIC: A High Performance ATM Host-Network Interface Chip", IEEE INFOCOM, Boston, page 179, April 1995

[57] W. Richard, "Smart Port Card Version 2 (SPC-II) Architecture", [Online]. Available: www.arl.wustl.edu/gigabitkits/Workshop_02_06/slides/richard_spc2.pdf, June 2002

[58] The GNU Zebra Home Page, [Online]. Available: www.zebra.org, September 2003

[59] P. Pappu, J. Parwatikar, J. Turner, K. Wong, "Distributed Queuing in Scalable High Performance Routers", INFOCOM 2003, San Francisco, March 2003

[60] M. Shreedhar, G. Varghese, "Efficient Fair Queuing Using Deficit Round Robin", SIGCOMM, Boston, page 231, October 1995

[61] F. Baker, C. Iturralde, F. Le Faucheur, B. Davie, "Aggregation of RSVP for IPv4 and IPv6 Reservations", RFC 3175, September 2001

[62] E. Kohler, R. Morris, B. Chen, J. Jannotti, M. Kaashoek, "The Click Modular Router", ACM Transactions on Computer Systems, Volume 18(3), page 263, August 2000

[63] P. Heegaard, "GenSyn - a Java Based Generator of Synthetic Internet Traffic Linking User Behaviour Models to Real Network Protocols", Presentation at ITC Specialist Seminar on IP Traffic Measurement, Modelling and Management, Monterey, September 2000

[64] The Public Netperf Homepage, [Online]. Available: www.netperf.org/netperf/NetperfPage.html, September 2003

[65] The Network Time Project Home Page, [Online]. Available: www.ntp.org/, September 2003

[66] D. Mills, "NTP Precision Time Synchronization", [Online]. Available: www.eecis.udel.edu/~mills/database/brief/precise/precise.pdf, January 2003