Bandwidth Management and Monitoring for IP Network Traffic: An Investigation

Barry Vivian William Irwin

Submitted in fulfilment of the requirements
for the Master of Science
in the Department of Computer Science
Rhodes University

January 2001

Abstract

Bandwidth management is a topic which is often discussed, but on which relatively little work has been done with regard to compiling a comprehensive set of techniques and methods for managing traffic on a network. What work has been done has concentrated on higher end networks, rather than the low bandwidth links which are commonly available in South Africa and other areas outside the United States.

With more organisations increasingly making use of the Internet on a daily basis, the demand for bandwidth is outstripping the ability of providers to upgrade their infrastructure. This resource is therefore in need of management. In addition, for Internet access to become economically viable for widespread use by schools, NGOs and other academic institutions, the associated costs need to be controlled.

Bandwidth management not only impacts on direct cost control, but encompasses the process of engineering a network and network resources in order to ensure the provision of as optimal a service as possible. Included in this is the provision of user education. Software has been developed for the implementation of traffic quotas, dynamic firewalling and visualisation.

The research investigates various methods for monitoring and management of IP traffic with particular applicability to low bandwidth links. Several forms of visualisation for the analysis of historical and near-realtime traffic data are also discussed, including the use of three-dimensional landscapes. A number of bandwidth management practices are proposed, and the advantages of their combination, and complementary use are highlighted. By implementing these suggested policies, a holistic approach can be taken to the issue of bandwidth management on Internet links.

Preface

The writer would like to acknowledge the help and assistance given by a number of individuals during the course of this research. My first thanks are to my supervisor, Mr George Wells for his input and for allowing me to follow my own ideas. I also wish to thank three fellow students for their contribution to this work; David Siebörger and Fred Fourie for their assistance in the development of the three-dimensional visualisation tools, in particular the animation sequences, and Guy Halse and David Siebörger for intellectual debate and discussions relating to the topic under study, often in the small hours. My appreciation also goes to Kevan Watkins and 'Jaco' Jacot-Guillarmod of the Rhodes University Information Technology Division for their willingness to facilitate my data collection and testing. My appreciation also goes to the the Department of Computer Science at Rhodes University for the facilities and equipment without which this research would not have been possible.

Finally, a very special thanks is expressed to my parents, for their support of this research, particularly for their critical discussion of the style and for proofreading.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Bandwidth management is a topic which is often discussed, but on which relatively little work has been done with regard to compiling a comprehensive set of techniques and methods for managing traffic on a network. What work has been done has concentrated on higher end networks such as ATM, rather than the very low bandwidth links which are commonly available in South Africa and other areas outside of the United States.

Internet bandwidth is an expensive, and scarce, resource. With more organisations increasingly making use of the Internet on a daily basis, the demand is outstripping the ability of providers to upgrade their infrastructure. This resource is in need of management, so that optimal use can be made of it.

The researcher's interest in the management of network bandwidth was originally stimulated by a problem experienced while running the network at a local NGO, the Grahamstown Foundation. During the annual National Festival of Science, Engineering and Technology (SciFest), exhibitors and visitors were making extensive use of the Internet, causing the line to be saturated, despite additional bandwidth having been installed in anticipation of increased usage over the event. This experience together with many of the bandwidth saving efforts being introduced at Rhodes University prompted the idea of conducting an investigation into what management techniques were available, and the effectiveness of each.

Local schools were also found to be experiencing the problem of facing ever in-

creasing costs for their Internet access, both from the access charges for their links, and per megabyte traffic charges. For Internet access to become viable for use by schools, NGOs and other academic institutions, the associated costs need to be controlled wherever possible.

Bandwidth management not only impacts on direct cost control, but encompasses the processes of engineering a network and network resources in terms of ensuring the provision of as optimal a service as possible, as well as providing user education so as to attain the best possible use of the resources available.

1.1 Aims

This research aimed to examine a number of bandwidth management techniques that could be combined to develop a holistic bandwidth management plan for an organisation. Methods for the monitoring and analysis of network traffic would also be explored.

An intended outcome was to provide a summary of the techniques currently available, along with some indication of their value, appropriateness for application in certain environments, and integration with other techniques. In some cases software would be developed by the researcher in order to provide a 'proof of concept' implementation of ideas that had been raised, or where existing software had been found to be lacking. It was envisaged that these implementations could be greatly improved on, and in many cases, be expanded further.

1.2 Document Layout

Chapter 2 provides an introductory background for the reader, dealing with protocol level issues of the TCP/IP suite with regards to its historical development, operation, design and known performance issues. It also discusses work that has been carried out by other researchers in the area of TCP/IP performance evaluation, and the improvement of the performance of the TCP protocol.

Chapter 3 deals with network monitoring techniques and the subsequent storage of network data that has been collected during the monitoring process. The visualisation of this data is also dealt with and a number of visualisation techniques are introduced and discussed. Two case studies are presented in order to provide examples of working implementations of the ideas and techniques presented.

Chapter 4 constitutes the bulk of this document. A number of bandwidth management techniques are presented, discussed and evaluated. Case studies are presented for several of the techniques, illustrating possible methods of implementation and providing a means for evaluating the theoretical techniques discussed.

Chapter 5 concludes the study, outlining possible future developments arising from this work. It also evaluates the extent to which the aims of this study have been achieved.

Several appendices are included as a part of this document. These contain technical and other information, the inclusion of which would have interrupted the flow of the document's progression.

Chapter 2

TCP Evolution and performance evaluation

This chapter provides a historical background to the the development of the Internet Protocol (IP) [78], Transmission Control Protocol (TCP) [76] and User Datagram Protocol (UDP) [136] as used on the Internet today. TCP is the protocol responsible for the majority of traffic on the Internet, and is used as the basis for such higher level protocols such as electronic mail (SMTP), the World Wide Web (HTTP) and file transfers (FTP). It is felt by the researcher that an understanding of the operation of lower order protocols such as TCP/IP is essential in order to be able to effectively manage current IP based networks.

Many of the issues which are dealt with in the following discussion are still prevalent on networks today. The TCP protocol has undergone many enhancements to improve both its reliability and performance in response to changing network conditions and to cater for the vast improvements in networking technology that have occurred over the last twenty years since its initial design. That these protocols are still functioning, are in widespread use, and have been successfully modified over the course of their history bears testament to the foresight of their designers, as well as the many individuals who have contributed to their continued growth and development. What follows in Section 2.2 is a summary of the development that TCP has undergone in order to improve performance and reliability.

2.1 Standards Process

The following text makes reference to three types of documents which are fundamental to the development of the Internet Protocol family. These are the 'Request For Comment' (RFC), 'Internet Standard' (STD) and 'Best Current Practice' (BCP) documents. These documents, published by the Internet Engineering Task Force (IETF), detail the design of protocols and other information relevant to the operation and growth of the Internet [29]. A central repository of this document series is maintained at the IETF website (`http://www.ietf.org/`), and mirrored by numerous sites around the world.

Each distinct version of an Internet standard is published as a part of the RFC document series. Included in this series are other documents, such as discussions of new research and memos regarding the status of the Internet. The series was started in 1969 as part of the original ARPANET project and is controlled as a whole by an RFC Editor, under the direction of the Internet Architecture Board (IAB). An RFC can be classed under a number of tracks indicating its status - 'Informational', 'Standard', 'Standards Track' for new developments, 'Experimental', or Obsolete. This classification was only introduced in its earliest form by RFC1083 (1988) [81] and followed up by RFC1100 [82], RFC1130 [141] and RFC1140 [142]. Postel provided further enhancements and clarification to the standards process in RFC1200 [143] (1993). The current standard is defined by STD1 (RFC2700) [149]. Many of the older RFCs, prior to the introduction of this notation scheme, other than those marked as standards or obsolete, remain unclassified. Certain RFCs standardise the way in which operations are performed, and are republished as a document in the Best Current Practice series [29]. Those RFCs documenting Internet standards are included in the Standard document series. The authors of RFC1796 [72] note that not all RFCs are standards, or are to be followed, and that attention must be given to when RFCs are declared obsolete.

As a part of the standards development process, a document will undergo a number of iterations as an Internet Draft. Internet drafts have a lifetime of six months, after which, if they have not been modified, or published as an RFC, they are removed. These drafts are not a formal publication of a standard or change but are

rather a means to provide a document for discussion and development by the Internet community as a whole. As such they should only be referenced as works in progress as they are liable to change at any time. Details of the standards process are published in RFC2026/BCP9 [29].

2.2 TCP Evolution

2.2.1 Early developments

The Transmission Control Protocol's earliest definition was in the Internet RFC761 published on the 1st of January 1980 entitled *DoD Standard Transmission Control Protocol* [77]. This was substantially revised and reworked, resulting in the publication of RFC793 [76] in September 1981, which was then republished as Internet Standard STD007 . This protocol was intended to provide a reliable session oriented means of connection between two networked hosts, using the Internet Protocol as defined in RFC791 (IP) [78] as an underlying transport mechanism. The original design stated that :

> "In principle, the TCP should be able to operate above a wide spectrum of communications systems ranging from hard-wired connections to packet switched or circuit-switched networks " - RFC 793, p1 [76]

Following the initial publication of the TCP standard, other researchers made a number of contributions, namely Clark in RFC813 [38] discussing the operation of the window and ACK strategies, and Postel's work, RFC879 [139], published in late 1983 detailing maximum segment sizes (MSS). Nagle presented an algorithm (the Nagle Algorithm) for better handling of congestion control in 1984 in RFC 896 [110]. The first publicly available implementation of the TCP specification was as a part of the 4.2BSD Unix distribution released by the University of California (Berkeley) in late 1983. However, once use of this became more

widespread, particular issues arose with regard to congestion, as noted by Van Jacobson in the introduction to his seminal 1988 paper *Congestion Avoidance and Control* [85]:

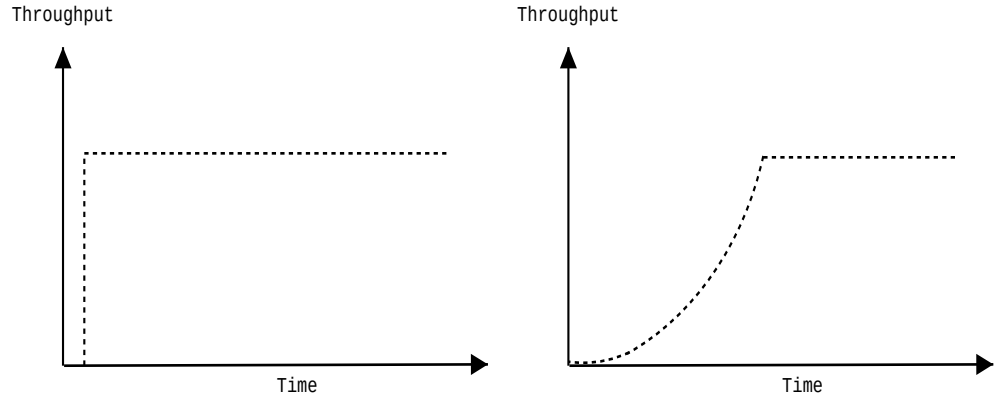
> In October of '86, the Internet had the first of what became a series of 'congestion collapses'. During this period, the data throughput from LBL to UC Berkeley (sites separated by 400 yards and three IMP[1] hops) dropped from 32 Kbps to 40 bps. Mike Karels and I were fascinated by this sudden factor-of-thousand drop in bandwidth and embarked on an investigation of why things had gotten so bad. We wondered, in particular, if the 4.3BSD (Berkeley UNIX) TCP was mis-behaving or if it could be tuned to work better under abysmal network conditions. The answer to both of these questions was "yes".

The paper detailed the introduction of seven new algorithms into the TCP stack in the then current 4.3BSD. These were later incorporated into the main BSD source tree resulting in the 4.3BSD-*Tahoe* release.

One of the more noticeable improvements in TCP performance was due to the new 'slow-start' algorithm in which a connection starts slowly, and speeds up until an equilibrium is reached. This provided a dramatic improvement over the original implementations which would start a connection transmitting as fast as possible, often resulting in congestion and subsequent packet loss, which in turn led to retransmission – which is a waste of bandwidth – and reduction of the throughput performance. The differences between the two approaches can be seen in Figure 2.1{a,b} showing 4.2BSD and the improved 4.3BSD-*Tahoe* respectively. The nett result of the implementation of this algorithm is that the TCP connection becomes self clocking, transmitting only as fast as it receives responses.

Other notable improvements to the TCP stack were the adjustment of the Retransmission Timeout Estimator (RTO), for handling of higher latencies on heavily congested networks, as well as a more efficient means of computation in order to reduce CPU overhead [85]. The exponential RTO 'back-off algorithm' developed

[1] Internet Message Processor (IMP) was an early form of router on the ARPANET



(a) 4.2BSD (b) 4.3BSD with slow-start

Figure 2.1: The 4.3BSD *Tahoe* slow start algorithm

by Karn at Bell Communications Research, rectified the problem where TCP performance would be further degraded during periods of congestion due to multiple successive retransmissions which would further congest the network. The new algorithm provided for an update of the RTO estimator, only if an acknowledgement (ACK) was received from the other host, failing which the data would continue to be resent at a constant interval until an ACK was received, or the connection timed out [85]. A further enhancement to TCP in combating congestion enabled dynamic sizing of the sliding window used to keep track of outstanding data, as opposed to the fixed window which had been used. If congestion was detected, the window size was halved, but was increased by one slot for each successful packet transmitted. The maximum size of packets to be transmitted is determined by the minimum value of the receiving host's advertised window and the local congestion window. These changes resulted in a significant performance gain [85].

The development of 4.3BSD-*Reno* in 1990 implemented the fast recovery/ retransmit and header prediction algorithms, which were published as an RFC by Stevens some seven years later in RFC2001 [165]. The TCP protocol was now able to effectively handle network congestion, but was facing another problem brought about by the increasing popularity of what became known as 'Long Fat Networks' (LFN) [88, p2] – networks with a high bandwidth, but also a high

latency. The increase in the use of satellites as an intercontinental bandwidth delivery tool serves as a prime example of this problem. Depending on the orbit of the satellite the latency could be up to half a second (twice the 239.6ms ground-station to satellite to ground-station latency) for a satellite in geostationary orbit [6], with $\pm$120ms latency for each of the satellite to ground legs [134]. Ground-station sites on the periphery of the satellite's view can experience a latency increase of up to 279ms [6], resulting in Round Trip times (RTT) of 558ms (twice the ground-station-to-ground-station latency).

The effect that this issue could have on TCP performance was first raised by Jacobson and Braden in RFC1072 [87] in October 1988, and again in June 1989 by Fox in RFC1106 [53] and McKenzie in August 1989 with RFC1110 [101]. These proposals were revisited in Jacobson *et al*'s October 1990 publication entitled *TCP Extension for High-Speed Paths* (RFC1185) [89]. The proposals were reworked and subsequently made obsolete by the publication of RFC1323 [88] which proposed a number of improvements to handle this and associated problems. They were predelicted on the premise that the performance of a TCP connection is governed by the bandwidth delay product (also referred to as the bandwidth*delay product or BDP) – the product of the connection bandwidth and the latency as measured for a round trip for a packet. The calculation as described in RFC1323 [88] is as follows:

$$bandwidth(bits/second) \times delay(second) = data(bits)$$

The result of this calculation is the amount of data that is required to 'fill the pipe' between the two network endpoints, and is the bufferspace that should be implemented by both parties in order to achieve maximum throughput. Performance problems arose with the TCP protocol when this value became large (Jacobson states in RFC1072 that values greater than 10^5 bits can be problematic and gives the example of a 45Mbit/second connection with a 30ms latency producing a bandwidth delay product of the order of 10^6 bits) [87]. The standard TCP implementation allocates 16 bits for the window sizing [76] [163] which means that the maximum amount of outstanding data is 2^{16} bytes or 64KB. With a low latency, as on a LAN, rates of 10Mbit/second can easily be achieved. However looking at a congested or high latency link with a 1.5 second RTT, the maximum throughput

is 43KB/s as determined by the formula that the maximum throughput is equal to $2^{16}(bytes)/RTT(seconds)$ [88]. On a 100Mbit/second LAN with a 1ms latency (2ms RTT), the throughput is 524KB/s. This raises the issue of the TCP performance being sub-optimal to the potential throughput of 12.5Mbytes/second, resulting in only a 4% utilisation. The solution to this was to introduce window scaling option in order to increase the maximum window size to 32 bits. In order to stay compatible with older TCP implementations, the structure of the header could not be changed, and so the existing window sizing value in the header is used as a scaling factor for the internal window held in memory on the host machines when used in conjunction with the window scaling option in the packet header. The use of 32 bits would mean that 100Mbit/second speeds (12.8MB/second) would still be sustainable with a 40 second RTT[2]. The ability to perform accurate per-flow RTT measurement was also proposed in RFC1323 [88], by means of adding a timestamp to each packet. This was not covered in the original specification as at that time it was a relatively costly operation to perform with regards to then current CPU processing power. It was also at that time difficult to obtain an accurate source of timing.

As a result of the window scaling options in combination with networks having high bandwidth delay products, there is the potential that the 32 bit sequence number for each TCP segment could 'wrap', resulting in there being two segments in transit with the same sequence number. When ACK packets are received by the sending system from the remote host there is a problem in terms of which of the identically numbered segments have been received, and which needs to be retransmitted. To protect against this, the 'Protection Against Wrapped Sequences' (PAWS) algorithm was introduced in 1993 [88]. This algorithm makes use of a combination of the sequence number and the timestamp to produce a unique value to aid in the detection of duplicate TCP segments, thereby providing a solution for the problem described above.

[2]To take this to extremes, assuming that there was a reliable means of 'high-speed' communications (running at the speed of light since a radio link would be the only viable means with current technology) between the Earth and Mars (the RTT for Earth-Mars communications is about 560 seconds), a throughput of approximately 936KB/s would be attainable – a similar throughput to that experienced on a 10Mbit Ethernet LAN. Thus the current 32 bit window would suffice for interplanetary communications.

TCP *Vegas* in 1994 was a further enhancement of the TCP protocol originally developed by the University of Arizona [43] [30], based on the *Reno* source code. The increased throughput (40-70% in some cases) that this version has exhibited is largely due to the congestion anticipation algorithms as opposed to the reactive congestion avoidance methods used in the *Reno* version. The major feature of this enhancement is that gradual changes are possible in the congestion window so as to maintain a higher transmission rate, as opposed to the halving introduced in *Tahoe*. In addition it also features a *slower* 'slow-start' algorithm as well as *faster* 'fast retransmits' [31].

As with the other modifications, the *Vegas* enhancements do not change the standard as originally defined in RFC 793 [76], but rather add more features which are inter-operable, and compatible with the existing legacy TCP implementations. In the case of *Vegas* the modifications are only required in the sending side of the connection. As far as real-world implementations go, the majority of operating systems in the Unix family are based on the 4.4BSD sources, or derivations thereof, and therefore support the features of this base TCP/IP implementation. Other TCP stacks implementations available (such as the Microsoft Winsock stack) also incorporate these features introduced in TCP-*Reno*.

The overriding constraint that all of these improvements on the original specification have is that, for the sake of compatibility, any changes made are required to be inter-operable with older implementations. Relatively early into the development TCP (September 1987), Postel released the *TCP and IP Bake off* (RFC1025) [140], which provided a list of test procedures for validating compliance with standards. More recently, the publication of RFC2398 (1998) [130] by Parker and Schmecel provides a list of tools that TCP implementers can use for validating their stack. The TCP requirements for Internet hosts was published in October 1989 as RFC1122 [23] (and later republished together with RFC1123 [24] as STD3) and included the requirement that hosts support slow start and congestion avoidance – fast retransmit and fast recovery were only published after this RFC.

Many of these features are tunable by the system administrator. FreeBSD offers control of these through the use of the *sysctl* utility. Table 2.1 shows some of the

variables that can be configured under FreeBSD [55] [54]. Linux offers similar functionality in the version 2.2.0 and later kernels.

Table 2.1: TCP option variables under FreeBSD

sysctl variable	Details
`tcp.rfc1323`	Implement the window scaling and timestamp options of RFC 1323
`tcp.rfc1644`	Implement Transaction TCP (T/TCP) as per RFC 1644
`tcp.mssdflt`	The default value used for the maximum segment size (MSS) when no advice to the contrary is received from MSS negotiation.
`tcp.rttdflt`	The value of the default maximum TCP Round Trip Time.
`tcp.sendspace`	Maximum TCP send window.
`tcp.recvspace`	Maximum TCP receive window.
`tcp.delayed_ack`	Delay ACK to try and piggyback it onto a data packet.

Notes: These variables can be set by the administrator in order to toggle the functionality of features or to tune performance. The variables can be accessed under `net.inet.`<*variable*>. For further details can be found in the *sysctl(8)* and *tcp(4)* manual pages.

The TCP protocol has continued to evolve, with the submission by Stevens in 1997 of RFC2001 [165] detailing minor improvements on the existing congestion avoidance algorithms of slow start, congestion avoidance, fast retransmit, and fast recovery as implemented in *TCP-Reno*. This work has since been superceded by the publication of RFC2581 in August 1999 by Allman *et al* [7]. Floyd and Henderson's RFC2582 [48] provides another experimental modification to the fast recovery algorithm, implemented by these authors as the TCP-*NewReno* protocol.

A recent experimental extension to aid in congestion avoidance is that of Explicit Congestion Notification (ECN). This is implemented by the introduction of two flags in the IP header utilising the last two bits of the Type of Service (TOS) field. Routers or receiving hosts are able to to provide notification of con-

gestion by means of an Explicit Congestion Notification to the sending host by setting the appropriate flag. This was initially proposed by Ramakrishnan and Floyd in RFC2481 [146]. The two flags are CE indicating congestion has been experienced, and ECT, indicating that the sender has an ECN-capable TCP transport [146]. This extension is not compatible with the historical operation of TOS as detailed in RFC791 [78] (see Section 2.3 for further details), but rather relies on the updated definition of this field as per RFC2474 [115]. Although still classed as an experimental RFC, the writer has seen signs of its use, as noted in Appendix B .

2.2.2 TCP for Transactions (T/TCP)

TCP provides a reliable byte orientated stream of data to be passed between hosts. It does however have a number of shortcomings, one of the most significant being that there is a sizable overhead involved in the three-way handshake as a part of the connection setup and teardown (an overhead of seven packets before any data is transferred [163, p232]). This is problematic when dealing with small transactions, which may occur in high volume, as there is a relatively high overhead in connection setup in relation to the data transferred. The need to provide a reliable means of low overhead transaction processing was recognised as early as 1995 with Braden's publication of RFC955 [22] raising this issue. This document was soon followed by RFC962 - *TCP-4 Prime* [129] by Padlipsky which assessed and expanded some of the proposals which had been put forward. Braden published further extensions to the TCP protocol in RFC1379 (1992) [25], and then as an experimental extension to the TCP stack in RFC1644 *T/TCP – TCP Extensions for Transactions Functional Specification* (1994) [26]. A more in-depth discussion of the operation and implementation of T/TCP was published in Stevens' *TCP/IP Illustrated Volume III* [164]. T/TCP is intended to provide a backward-compatible extension to the TCP protocol "for efficient transaction-oriented (request/response) service" [26, p1], providing a reliable alternative to the IP datagram based UDP service [136]. The nett result of these extensions is that the total packet count required drops to three [26, p6].

2.2.3 Selective Acknowledgement

Another enhancement made to the TCP stack is that of Selective Acknowledgement (SACK). This allows for the situation where multiple packets may have been lost in the transmission between hosts. The traditional response would be for the receiver to inform the sender of the lost packets at the rate of one packet per RTT cycle. SACK improves this by enabling the receiver to rather specify which data packets have been received. The sender is then able to selectively resend only the missing packets. This process is described in RFC2018 [100], building on some of the issues which were raised in RFC1072 [87], with regards to TCP performance over LFN's. Floyd *et al* suggest in RFC2883 [49], an extension to the proposed SACK algorithm which allows the sender to infer the order in which packets have been successfully transmitted to the destination host.

The authors of RFC2018 have also proposed a complementary Forward Acknowledgement (FACK) method of congestion control, which operates in conjunction with SACK [99].

2.3 The Integration of Quality of Service

When the initial TCP specification was compiled, no provision was made for any true quality of service (QoS) features within the protocol. What was provided was an interface to the Type of Service (TOS) field contained in the IP header. This is an eight bit field contained in the header as described in RFC791 [78]. Little use was initially made of this option. Almquit clarified and expanded on the use of this feature in RFC1349 (1992) [11], which also introduced a fourth possible TOS value, that of a minimal cost path. The TOS feature was designed to allow higher level protocols using IP as a transport mechanism to provide details as to the precedence of the packet, and what trade-offs should possibly be applied when making a decision as to which path to choose. Of the eight bits available in the TOS field, only four (bits three to six) are used, their function being summarised in Table 2.2. The initial three bits (zero to two) of the field form the precedence field which, although defined in RFC761 [77] and RFC795 [138], was initially ignored

by most IP implementations [163] and set to zero (routine precedence). The use of this is however currently supported by Cisco routers [179]. These settings are not guarantees, but are a best effort service.

Table 2.3 shows common settings for various protocols. The 4.4BSD-*Reno* release was one of the first implementations of an IP stack that actually made use of and set values for the TOS fields. RFC2474 [115] provides an update on the use of the TOS field, and its depreciation along with the IPv6 Traffic Class field in favour of a backward compatible Differentiated Services (DS) field. The architecture for the implementation of this feature in the TCP/IP stack is described in RFC2475 [21].

The DS field makes use of the first six bits of the TOS field (this can be seen by the large TOS values captured by *tcpdump*). Traffic from hosts that utilise either the TOS precedence bits, or the DS field output produced by *tcpdump* will often show large values such as 0x40, which are not directly comparable to the values listed in Table 2.3. The remaining two bits are used for Explicit Congestion Notification as described by Ramakrishnan and Floyd in RFC2481 [146]. The use of the TOS/DS field is discussed further in Section 4.4.

Table 2.2: Specification of Type of Service values

Bit Pattern (bits 3-6)	tcpdump TOS Value	Interpretation
0000	0x00 or null	normal service
1000	0x10	minimise delay
0100	0x08	maximise throughput
0010	0x04	maximise reliability
0001	0x02	minimise monetary cost

Formal QoS additions to the IP protocol and implementation of guaranteed delay and bandwidth at the network level are detailed in RFC2212 [156]with a view to being incorporated into the standard. RFC 2998 discusses details regarding the introduction of a QoS based routing scheme for the Internet building on the work initially presented in RFC2386 [40]. Huston in RFC2990 [73] provides a summary of current QoS work for implementation on the Internet, and a discussion of future research and development in this field.

Table 2.3: Type of Service settings for common protocols

Application	Hex Value	Details
Telnet Rlogin	0x10	Interactive protocol therefore a minimal delay is desirable
FTP control data	 0x10 0x08	 Interactive commands The data flow should be optimised for maximum throughput
SMTP command phase data phase	 0x10 0x08	 Fast interactive response Maximum data throughput
ICMP	0x00	no special handling necessary
SNMP	0x04	UDP based so try for more reliable links
NNTP	0x02	High volume non critical traffic so minimal cost

[after Stevens [163, p35]]

2.4 Packet Sizing

Although the Internet Protocol allows for a gateway to fragment a packet if it exceeds the maximum transmission size of a link, this is not always desirable, and can lead to a degradation in performance, or even communication problems [94]. Consequently, work has been done on incorporating a method of enabling the IP stack to discover the optimal MTU (Maximum Transmission Unit) for a path between two hosts (PMTU). This was first proposed in RFC1063 (1988) [108], and made obsolete in November 1993 by Mogul and Deering's publication of RFC1191 [107], which proposed the incorporation of these features as a draft standard. The MTU discovery operates by sending packets of a given size, with the 'do not fragment' (DF) flag set. When the MTU is too large for a gateway to transmit, the gateway returns an ICMP error message (ICMP type 3 code 4 [163]) to the sending host, indicating that the packet required fragmentation in order to be transmitted, but this was not possible due to the DF flag being set. The sending host then reduces the MTU size and retries the transmission to the target host [107].

Having an optimal MTU is not only important in terms of the transmission packets

without fragmentation, but on smaller links can provide a major influence on the link performance. The default MTU use by hosts is 576 bytes [23]. The actual MTU used for transmission is initially the minimum of the transmission media MTU and the 576 byte default. On links with a configurable MTU, such as those running PPP, the administrator can tweak the performance of the link towards two extremes, depending on the primary role of the link. In order to obtain as high a throughput rate as possible, the MTU should be set to a large value, typically 1500 bytes over a dialup modem link. The disadvantage of this is that the latency is increased, as a packet has, theoretically, to wait for the transmission of the previous 1500 byte packet, in addition to the link protocol overhead which is used for encapsulation. The opposite of this is to use smaller MTU settings. Lower settings increase the responsiveness for interactive connections such as remote terminal access, but the efficiency of the link is decreased, since the ratio of link protocol overhead to IP payload is much larger. The publication of RFC2923 [96], documents a number of problems with the current TCP implementations for the discovery of the optimal PMTU.

2.5 Future Development

Work has continued on enhancing TCP, particularly with regards to performance over satellite links. The issue was first raised as an Internet RFC346 [134] by Postel in 1972, who expressed the need to take the much increased link latency into account when sizing buffers. The Internet BCP28 (RFC2488) discusses a number of methods specific to enhancing TCP performance over satellite communications, and some problems specific to using this transport medium [6]. In particular, due to the higher bit-error rates inherent in satellite communications, the TCP congestion control mechanism may be triggered due to lost packets, when in fact there is no congestion.

The effect of asymmetric links, where outgoing data is sent via a lower bandwidth terrestrial line and the response is received via satellite down-link are also discussed [6]. This asymmetric bandwidth layout is a fairly common practice, and is discussed further in Chapter 4 (see Section 4.12). A more recent document

in the form of RFC2760 [8], published in February 2000, expands on the work presented in BCP28. Related work is RFC2757 [109] which discusses operation at the other end of the bandwidth spectrum, which has become common with technologies such as wireless Wide Area Networks (WAN) and network access via GSM (Global System for Mobile Communications) enabled units. An experimental modification to the sizing of the Initial Window size in TCP has been presented in RFC2414 [5]. Tsaoussidis *et al* [170] raise an interesting point with regards to wireless networks, and mobile devices, by evaluation the energy efficiency of the various error recovery strategies in *Tahoe*, *Reno* and *NewReno* TCP implementations. This should be taken into consideration as the number, and use of mobile devices increases.

2.6 Summation

TCP is by no means a perfect protocol and RFC2525 [132] lists a number of known problems with the TCP standard and associated extensions. Included in the document are possible problems arising out of the failure to fully implement the protocol according to specification.

The Internet is constantly evolving to incorporate new technologies and to solve problems that may arise. In particular, work is being done on improving the flow management abilities of the protocols used. This is in response to increasing demands for QoS features in order to provide reliable audio and video distribution. Congestion and the impact of LFNs on network traffic are probably the two biggest issues of concern to most administrators. A network link under load will experience an increased latency and subsequent congestion. If the traffic on this link is not carefully managed, the congestion and the resultant effect on traffic can render the line unusable. In order to avoid this, an ideal is to aim for a maximum utilisation of the network link that is somewhat lower than the rated capacity. This would allow for traffic bursts when they occur, while minimising the congestion that may be experienced. In many cases the worst congestion experienced in a link between two hosts is on the final hop links between the host's and their network access providers. This is supported by the findings of Odlyzko in his paper *Data*

networks are lightly utilised , and will stay that way [117].

Chapter 3 discusses a number of methods for monitoring traffic on a network. A number of techniques for analysing the collected data by means of graphical visualisation are also put forward.

Chapter 3

Monitoring and Visualisation

3.1 Monitoring

A network administrator needs to perform some form of traffic monitoring in order to have a foundational basis for making decisions about bandwidth management. This is necessary in order to ascertain what traffic is present on the network and which traffic needs to be factored into an overall management strategy. This same monitoring process allows an administrator to gauge what normal traffic levels on a particular network are. Informed decisions regarding network bandwidth can only be made after monitoring, and the subsequent analysis of collected traffic data.

Not all monitoring results are of equal value to the administrator. In order to gain meaningful information from the exercise, the right monitoring policies must be put in place to determine what requires monitoring, and where the optimal position on the network is for performing this monitoring. In essence, monitoring of the right information is as important as performing the monitoring at the right places on the network.

Monitoring can be divided into two broad classes, active and passive, each of which has its own advantages, disadvantages, and particular suitabilities.

3.1.1 Active monitoring

This is usually an active probe sent out onto the network by the monitoring software, and which by its very nature, consumes bandwidth on the network. Because of this, the type and applicability of a probe should be carefully considered. For example, it would be impractical to attempt to assess the throughput of a 64Kbit/second Diginet line by performing a 1Mbyte transfer every 15 minutes, as this would constitute a substantial proportion of the bandwidth (14.5 % of the available bandwidth in the given 15 minute interval, assuming a throughput of 64Kbit/second). In most cases a 300KB (4.3% utilisation) or 100KB (1.4% utilisation) transfer would be sufficient, although smaller transfers might suffer from too higher a variation due to external factors influencing the traffic on network. A balance needs to be struck between the level of detail required, the perceived accuracy, and the impact of the monitoring on the resource.

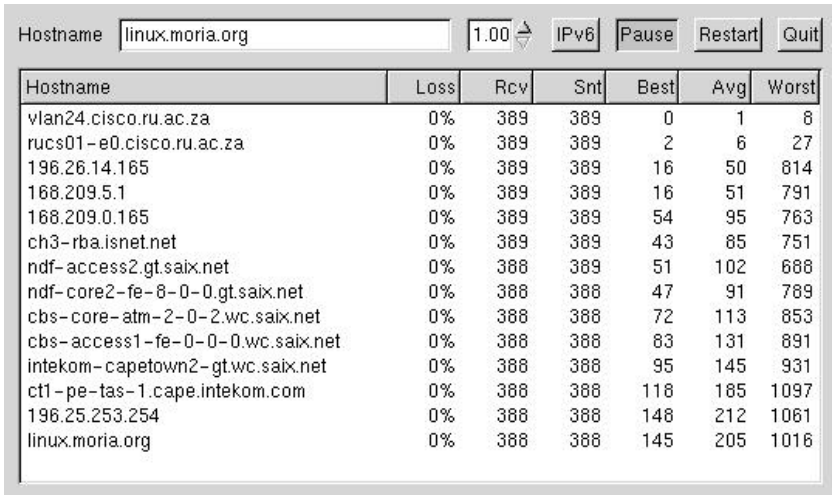
Active monitoring has the advantage of being able to measure variables such as real available bandwidth (quantified by throughput of a given amount of data) on a link, latency and packet loss, which would not be possible using passive means. Care also needs to be taken if probes are frequent to ensure that they are excluded from other monitoring, in order not to adversely skew results. An alternate approach is to actually include the bandwidth used by these probes in the usage summaries, which can prove useful in making the administrator aware of the impact of these tests on the total bandwidth.

3.1.2 Case Study 1: Bandwidth Probe

An example of active monitoring is the bandwidth probe monitoring that the researcher carried out from August 1999 until October 2000. The purpose of this probe was to measure the available throughput between two hosts on the Internet. The origin test system, `rucus.ru.ac.za` is located at Rhodes University with `linux.moria.org` housed at a local NGO, serving as the target system. These two hosts were selected by the writer since although they were physically within 1km of each other, with regards to network topology they are fairly distant (14 hops), providing a route passing through three separate Internet Service Providers

(ISP). The route between the systems is shown in Figure 3.1, which also shows the cumulative Round Trip Times (RTT) for each node on the route round South Africa. This figure also displays the best, average and worst RTT measured for each node, which can be useful in determining bottlenecks by comparing the difference in RTT between consecutive nodes. Testing was performed by means of the *tcpblast* utility described in Section 3.1.8.3.

A *tcpblast* test of 100Kbytes was run every 15 minutes and the resultant data stored in a database. This size was chosen as it represented a compromise between a medium sized file download, and an average webpage size (including graphics), while having a minimal impact on the 64Kbit/s line to which the target system was connected. The size also allowed for the averaging out of any short-term performance boosts or decreases that may have occurred. A larger test size would provide more accurate results but this had to be balanced against the impact it would have on the primary use of the line. The effect of the testing was calculated to be negligible on the normal day-to-day operation and activities of the link both between the University and its ISP (utilising 0.18% of the available bandwidth in a 15 minute period) and the NGO to ISP link (1.4%).


Hostname linux.moria.org 1.00 IPv6 Pause Restart Quit

Hostname	Loss	Rcv	Snt	Best	Avg	Worst
vlan24.cisco.ru.ac.za	0%	389	389	0	1	8
rucs01-e0.cisco.ru.ac.za	0%	389	389	2	6	27
196.26.14.165	0%	389	389	16	50	814
168.209.5.1	0%	389	389	16	51	791
168.209.0.165	0%	389	389	54	95	763
ch3-rba.isnet.net	0%	389	389	43	85	751
ndf-access2.gt.saix.net	0%	388	389	51	102	688
ndf-core2-fe-8-0-0.gt.saix.net	0%	388	388	47	91	789
cbs-core-atm-2-0-2.wc.saix.net	0%	388	388	72	113	853
cbs-access1-fe-0-0-0.wc.saix.net	0%	388	388	83	131	891
intekom-capetown2-gt.wc.saix.net	0%	388	388	95	145	931
ct1-pe-tas-1.cape.intekom.com	0%	388	388	118	185	1097
196.25.253.254	0%	388	388	148	212	1061
linux.moria.org	0%	388	388	145	205	1016

Figure 3.1: Route between `rucus.ru.ac.za` and `linux.moria.org`

Tests were run automatically, and the results submitted by email. These email messages were processed by a script developed by the researcher, and the processed data stored in a database. A number of front-ends for this database were

developed by the writer for retrieving the data. One was an interactive web-based browser for the data held in the database. This tool allows a user to view daily throughput data for any date held within the database. The output graphs are generated in realtime in response to user queries. This was found to be very useful when providing evidence to the NGO's ISP of problems with the throughput, resulting in a line upgrade at the end of September 1999. An example of the graphs produced by this front-end can be seen in Figure 3.2. The smooth slope between 04h00 and 11h15 indicates a period where no results were obtained due to the target system being unreachable during the period of the line upgrade. The improvement in available bandwidth is apparent from 18h00 onwards in comparison with the available bandwidth between 00h00 and 04h00.

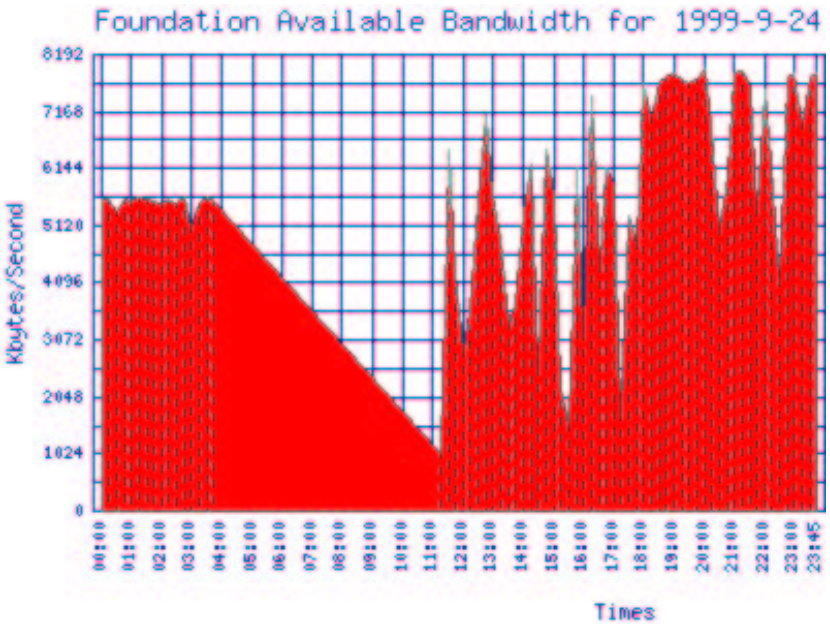


Figure 3.2: Example output from the Bandwidth Probe

The data collected by this active probe was also used for the landscape and colour map visualisations discussed in Sections 3.3.2.4 and 3.3.2.3 respectively.

3.1.3 Passive monitoring

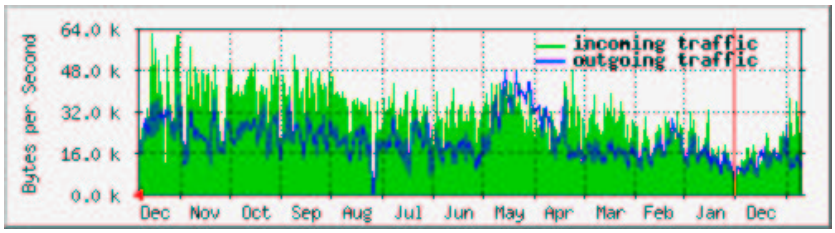
Data is collected passively, either by means of remote RMON[1] and SNMP[2] agents on networking equipment, or by the accounting software on the local machine. This form of monitoring is not suitable for some measurements, such as determining the actual bandwidth available between two hosts. Passive means would only be able to calculate the bandwidth currently in use or current throughput on a link and subtract this from the theoretical maximum bandwidth or throughput of the link. This value may not be an accurate indication, particularly if there are a number of intervening hops, or if the link is being aggregated into a larger network 'pipe' further upstream, and if this pipe is congested. Data from RMON/SNMP or other agents on networking equipment is collected by software probes on the network. The bandwidth used by these retrievals is in most cases negligible, as they are performed via the LAN interface of WAN routers and other network access equipment, and forms a relatively insignificant portion of the total LAN traffic. If data is being retrieved over a WAN or other low-speed link, the impact of this will have to be assessed in a similar manner to active monitoring.

3.1.4 Case Study 2: Internet Traffic Monitor

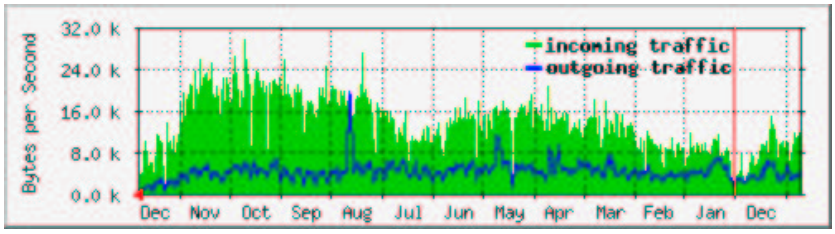
This case study describes the implementation and operation of the Internet traffic monitor set up by Rhodes University. During the last year, there had been a large increase in the utilisation of Rhodes University's Internet links as shown in Figure 3.3{a,b} respectively for the two Internet links. Following from this increase, and a need to ascertain the composition of the traffic, the researcher suggested to the University Information Technology Division, that a system be implemented that would provide a breakdown of what constituted the total Internet traffic. This is similar to the FlowScan system developed by Plonka [133].

A monitoring system was setup by the Information Technology division with input from the writer using *tcpdump* [90] for data capture and *RRDTool* [119] for data

[1]The Remote Monitoring SNMP Specification as defined in RFC 2819,2895 [174] [19]
[2]Simple Network Management Protocol as defined in RFC 1157 [33]



(a) Internet Solution link (512Kbit/s)



(b) Uninet link (256Kbit/s)

Figure 3.3: Rhodes Internet link utilisation

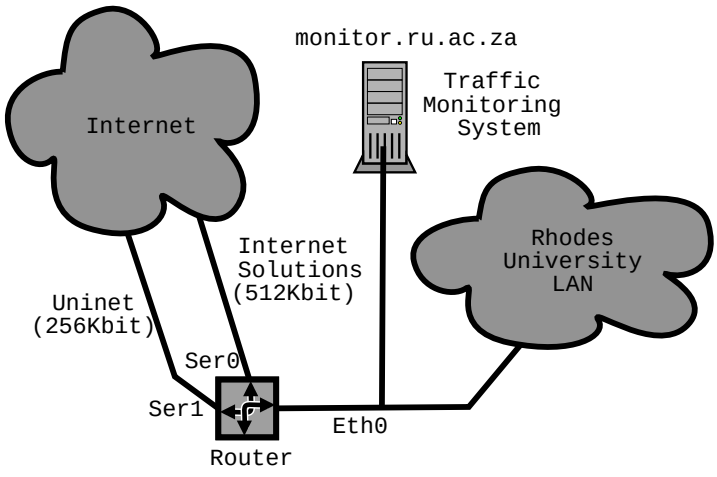
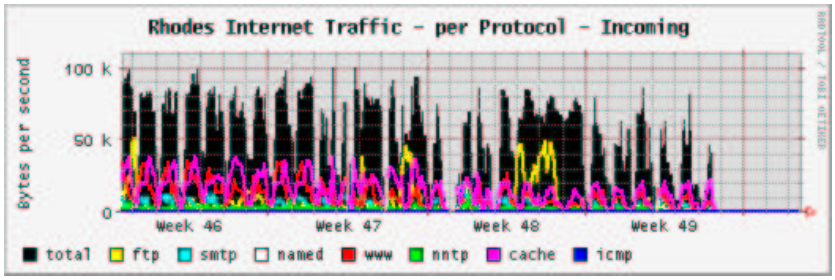
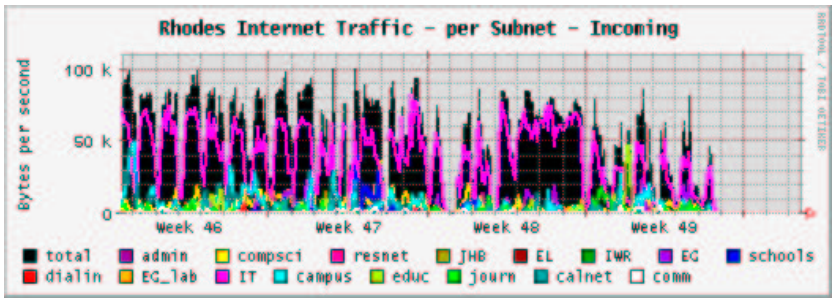


Figure 3.4: Logical layout of the traffic monitoring system



(a) Incoming Traffic by Protocol



(b) Incoming Traffic by Subnet

Figure 3.5: Example output from monitor system

Note: The peak values shown in these graphs are greater than the maximum of either of the links as shown in Figure 3.3 due to the aggregation of both links' throughput by the monitoring system.

storage and visualisation. Figure 3.4 shows details of the logical layout of the system. This layout allowed the monitoring system to see all Internet traffic passing in and out of the University network. Traffic between local hosts was excluded by means of a filter expression to the *tcpdump* utility. Traffic data was processed and summaries produced on a per subnet and per protocol basis. Examples of these outputs can be seen in Figure 3.5. The resultant graphs allow for quick visual inspection as to what protocols are consuming the majority of bandwidth, as well as which subnets are the most active with regards to bandwidth utilisation.

3.1.5 Suitability of monitoring

When preparing to install a monitoring system, the right type of monitoring needs to be determined in order to accurately obtain the data required. The impact that the monitoring method selected will have on the network should also be considered. Care should be taken not to poll too often, either actively or as a part of passive monitoring, as this can cause serious degradation of network performance due to the loading of the network infrastructure such as routers, causing possible packet loss and increased latency[3]. This was experienced by the researcher, where a new network management system was installed at the University, and all the probes enabled. The result was that the gateway router was placed under such load, that it was dropping packets. Some of the effects of polling are discussed in Section 4.3.3.

The right level of detail should also be selected and a decision made as to what routine monitoring is appropriate. As a rule, high level, low intensity monitoring is sufficient for most purposes, with lower level, high detail monitoring being implemented during periods of troubleshooting or when more detailed analysis is required.

[3]Latency is a measure of the delay between a packet being sent and received between two hosts.

3.1.6 Passive Monitoring Tools

This section evaluates a few of the many network monitoring tools that are currently available and their suitability for certain monitoring tasks. Tools are grouped under two broad categories corresponding to active and passive monitoring. Active tools are discussed in Section 3.1.8. All these tools are freely available on the Internet under various Open Source licenses. The Freshmeat software repository `http://freshmeat.net` provides a good starting point for locating these tools for download. Tools are also available from the sites referenced in the discussion.

3.1.6.1 Tcpdump

Tcpdump [167] is probably the most ubiquitous and flexible tool for monitoring network traffic on an IP network and is available for a plethora of operating systems. This tool can also perform decoding of many of the higher level protocols running on TCP/IP such as NFS and SMB (Microsoft Windows file sharing) [90]. Support is also included for decoding of IPX (Novell NetWare) traffic. The package provides the ability to decode the packet headers and optionally the packet payloads into an ASCII format for display. This output can be further processed and interpreted. Functionality is also available to save captured traffic to a file, either as whole packets, or as just the packet headers (the default operation), or to read data from a capture file for 'after the fact' processing.

Figure 3.6 shows an example of output, illustrating the decoding of four packets showing the capture time, originating host and port, destination host and port, transport protocol and other protocol specific information that may be contained in the packet. Variable levels of reporting are also possible, the figure contains the minimal output as produced by the `-q` switch. The four packets show a passive FTP session (packets one and three), ICMP echo request (ping) and a NNTP Network news connection. Vast quantities of information can be produced by this utility, but support is included for an advanced filtering logic ('pcap logic') which allows for relatively fine grained selection of traffic based on a number of parameters. Further details can be found in the tcpdump manual [90]. Over the period of

```
11:18:57.803163 lizard.ru.ac.za.3263 >
                  ftp.is.co.za.1940: tcp 0 (DF)
11:18:57.805279 centauri.sac.ecape.school.za >
                  ftp.is.co.za: icmp: echo request
11:18:57.807292 ftp3.za.freebsd.org.3452 >
                  rucus.ru.ac.za.2782: tcp 1460 (DF)
11:18:57.807360 join.news.pipex.net.59972 >
                  quagga.ru.ac.za.nntp: tcp 0 (DF) [tos 0x40]
```

Figure 3.6: Example *tcpdump* output illustrating decoding of four packets

Note: Actual *tcpdump* output varies depending on the implementation, this is output as presented by the version developed at Lawrence Berkeley Labs. The output lines above have been split after the '>' for readability.

the program's development, much of the core functionality has been moved into a separate library for packet capture (*libpcap*). As such, this utility has become a front-end that implements the many features of the library. The pcap library is widely used as a basis for many other tools. Case Study 2 (Section 3.1.4) discusses a monitoring system that is implemented using *tcpdump* as the principal tool for gathering information. This program was used extensively throughout the course of this research for data gathering, protocol decoding and diagnosis of network problems.

3.1.6.2 IP Flow meter (ipfm)

This package makes use of the *pcap* library to obtain network data for processing. *Ipfm* [35] serves as a useful tool for analysing bandwidth usage by hosts on a network, as it stores a table of all IP host pairs communicating and the total amounts of traffic flowing in each direction. This table is periodically output to disk in text format which can be further processed. Tools such as this can be invaluable in determining which hosts on one's network are responsible for the majority of traffic, and which external hosts are popular. The pcap library filter logic allows for the selection of traffic (such as the exclusion of traffic between hosts on the local LAN) if required.

3.1.6.3 Trafshow

Trafshow [173] provides a real-time monitoring of traffic on a network segment, showing data flows between IP address/port pairs. Flows are ordered by byte-count, making it easy to determine what hosts are using the majority of traffic. The average transfer rate over the lifetime of the connection is also displayed. An example of the output produced can be seen in Figure 3.7.

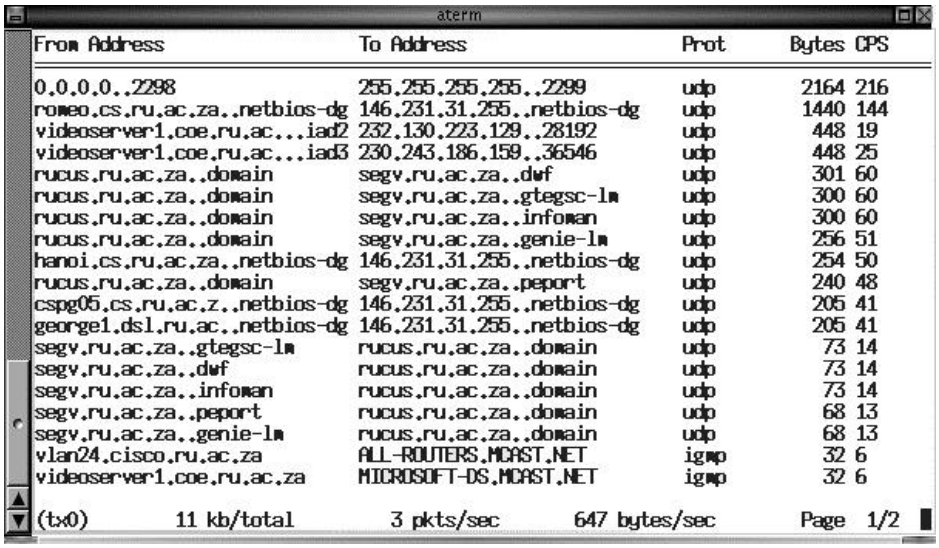


Figure 3.7: Example *trafshow* output of network traffic

3.1.6.4 NTOP

Ntop [45] provides a display similar to the Unix top(1) command, but for network traffic. It supports a number of advanced features, such as being able to log data directly to a database backend. An option is also avaliable for the utility to run as a daemon, whereby remote users can access the data through the internal web-server. Data collected can be ordered by a number of metrics. In many ways, *ntop* provides much of the similarity of a RMON agent. The researcher did however experience problems when using this software on FreeBSD, due to a known bug which caused the program to use all available CPU time. This problem is not present under Linux. This tool did prove to be able to provide considerable insight into the traffic flowing on the network.

3.1.6.5 SNMP

This covers a class of tools such as MRTG [121] and Cricket [4], which collect counter statistics from network infrastructures, and visualise these statistics by means of graphs (See Section 3.3.2). The most common use for these tools is graphing the InOctet and OutOctet counters on router interfaces, which respectively provide counts of the number of bytes passing in and out of the interface. Many other tools also support SNMP [33] monitoring. This includes monitoring performed by RMON agents, which can be useful in determining the top hosts with regards to traffic, as well as the distribution of packet sizes on a network. The researcher implemented a utility for Unix systems with a very limited subset of RMON functionality. which collected data on the traffic between hosts on the network.

3.1.7 Sniffer Placement

A ‘sniffer’ is a program which can potentially retrieve and process all the traffic that is present on the network segment to which it is attached. This is achieved by placing the network interface into promiscuous mode, rather than the standard mode which only retrieves packets marked with the network card’s MAC address. On switched networks, a sniffer is only able to potentially see the traffic on the particular segment to which it is attached.

It should be noted that those tools based on libpcap are in fact sniffers, and as such suffer from the same shortcomings as products directly billed as such. This is an important consideration in a switched network environment, which is becoming more common. A further consideration is the actual capacity that the software utilities are able to monitor. The majority of switches allow for a port to be configured as a monitor port for a number of ports, thereby receiving a duplicate of any traffic being passed on the monitored ports. This can prove problematic if a single 100Mbit/second port is being used to monitor more than one port of the same speed, as at times of peak load the link between the monitor system and the switch will be saturated (as opposed to the switch backplane). The result of such a situation is a potential data loss by the monitoring station and consequently the

monitor system not having a complete set of data available for capture. A general solution to this is to only monitor the up-link port of the switch (which is usually limited to 100Mbit/second in any case), as this port imposes the limit at which the others can exchange information with hosts not on the switch.

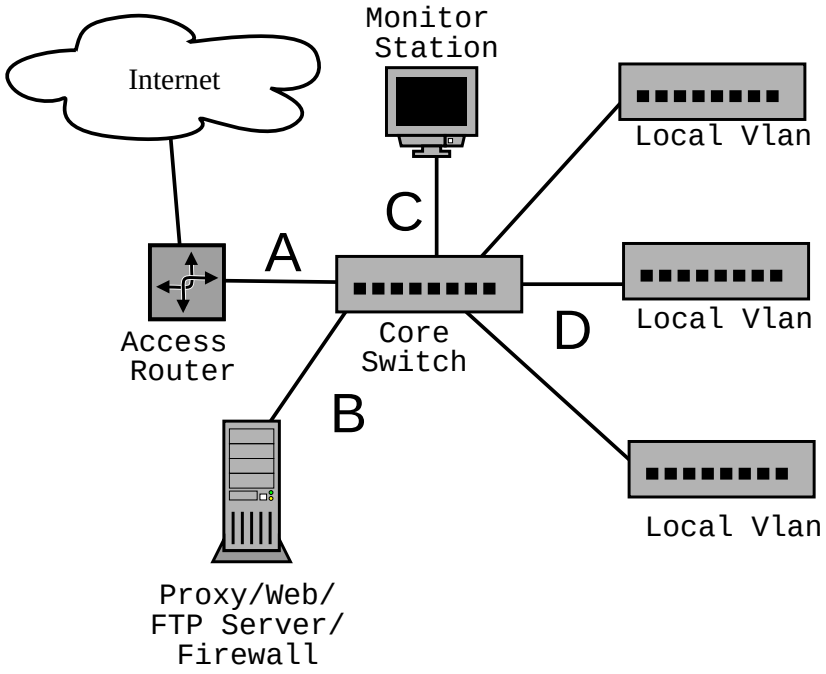


Figure 3.8: Placement options for a sniffer on a network

Placement of the host running the monitoring software on the network should also be carefully considered. Figure 3.8 illustrates the suggested placement options for monitoring workstations. Each of the points labelled A-D in Figure 3.8 can be used for monitoring, but has advantages and disadvantages depending on what the sniffer is meant to monitor.

The researcher has found that the link indicated by A provides the prime site for monitoring traffic on the network's Internet link since both incoming and outgoing Internet traffic will be visible on this link. Site B should be used for specifically monitoring traffic from web (HTTP), FTP, or Proxy servers. A dedicated monitoring machine can be connected to the network as shown by link C. If a core switch is used, the port that this machine is connected to on the switch can be configured as a monitor port, in order to duplicate the traffic from one of the other the links

that the administrator wishes to have monitored. Using this technique monitoring can easily be changed without the need to relocate monitoring hardware, or perform network re-cabling. Specific portions of the network can be monitored using the links indicated by D. This would most likely be in order to assess traffic local to the particular network segment, as Internet traffic could be monitored via point A with appropriate use of filters on the capture program to select this traffic.

3.1.8 Active Tools

3.1.8.1 Ping

This is a tool, included with most modern operating systems, that allows one to send ICMP [137] echo requests (ICMP type 8 [137] [163, p71]) to a host, and receive echo responses (type 0), from which the round trip time (RTT) can be calculated. This provides a measure of the latency to the host being tested. Ping can also be used to determine the packet loss rate (plr) of the link between the sending and target hosts. In its default configuration ping uses a minimal amount of bandwidth (64 bytes per packet under FreeBSD, Linux and Microsoft Windows), but most implementations support variations to the sending frequency and packet payload size. These options should be used with care as they can have a detrimental effect on a network, often referred to as a ‘ping flood’.

3.1.8.2 Bing

Bing [18] is a utility which attempts to quantify the bandwidth available between a local and remote host on a point-to-point link, based on the RTT timings of different sized ping packets on each end of the link. A moderately accurate assessment of the available bandwidth between two hosts is determined, even over routes with multiple hops such as the Internet, but the accuracy decreases exponentially as the number of hops increase. A comparison of *bing* performance to that of the *tcpblast* utility (see Section 3.1.8.3) is shown in Table 3.1.

Table 3.1: Example *bing* Bandwidth test results

Test host	# of hops	Connection Type	*bing* . estimation (Kbit/s)	*tcpblast* estimation (Kbit/s)
`barry1.dsl.ru.ac.za`[1]	2	200KB/s DSL	840	1601
`rucus.ru.ac.za`[2]	1	10Mbit/s Ethernet	7598	10113
`epsilon.ru.ac.za`[3]	1	10Mbit/s Ethernet	6206	7713
`linux.moria.org`	14	64Kbit/s Diginet	48	61.7
`ftp.is.co.za`[4]	9	512Kbit/s Diginet	365	410

Notes:

1 This is the largest anomaly presented with only 52% of the bandwidth reported. The results were consistently repeated on several independent occasions

2 This was a directly switched connection

3 Although directly switched, the slowdown experienced in comparison to `rucus.ru.ac.za` is most likely due to the old Lance Ethernet transceiver in the SparcClassic IPX workstation that was used for testing.

4 This is a case where bing is useful. Since a direct *tcpblast* connection was not possible to `ftp.is.co.za` the value used is to a machine on the same network, but *bing* is able to provide a useful indication of the bandwidth available

3.1.8.3 Tcpblast

The TCP throughput of a connection between two hosts is measured by this utility [92]. This software not suitable for use over links that implement any form of compression (such as PPP and modem links) since it does not use a random fill for its packet payloads. In experiments the researcher was able to achieve 11.5Kbyte/sec *tcpblast* rates over a 33.6Kbit/s analogue modem. These results were limited by the speed of the serial port UART[4] of 115Kbit/s, the high bitrate being achieved by the compression used as part of the PPP protocol. With PPP and modem compression disabled a result of 3.3Kbyte/sec was obtained. The writer developed a modified version of this utility which uses a random fill, and was able to achieve more realistic throughput rates. It did however raise the issue of the effect of enabling some form of compression on a network link, either in software (e.g. PPP) or hardware (the MNP5 compression that forms part of the modem connection). Another derivative of the same name which adds UDP tests and other functionality is available from the TUCOWS Linux software archive, but was not tested by the researcher.

3.1.8.4 Netperf

Netperf [69] is a full featured TCP and UDP performance testing system developed by Hewlett Packard Information Networks Division. Detailed reporting is available, and includes the ability to set various statistical confidence levels for the output of tests. This package has the tendency to generate very large amounts of traffic (several gigabytes of network traffic in total when running the comprehensive test group) and as such is not recommended for use on slower links.

3.2 Data Storage

Monitoring a network, by its very nature, produces very large datasets. These datasets can either be processed and discarded or, more usefully, stored for a pe-

[4]Universal Asynchronous Receiver/ Transmitter

riod of time. The keeping of historical data is essential for any kind of meaningful analysis and visualisation, or for the development of any statistical models. Without a record as to what has happened in the past, the administrator is unable to adequately plan for or anticipate future problems. If, for example, it can be determined that historically the organisation's bandwidth needs to be doubled every 30-36 months this can be timeously factored into a network development strategy, and budgeted for, instead of the case where the bandwidth slowly becomes choked, and crisis management has to be put in place – often at much higher cost. The level of detail in which these records are kept depends greatly on what the administrator plans to use them for. In some cases a simple graphical log such as that produced by the MRTG utility (see Section 3.3.2.1) may be sufficient, while other circumstances may require much more detail. When starting to collect and store network statistics, it is advisable to collect in too much detail, rather than too little for a couple of months until one has decided what data is actually necessary.

The format in which data is stored should ideally allow for easy access and retrieval, while also providing a space efficient means of storage. It was found by the researcher, as implemented in Case Study 1 (see Section 3.1.2), that an SQL[5] based relational database fulfilled these requirements, along with providing fast access and the ability to construct queries dealing with related data. With regards to space efficiency, the raw data produced by the test system used in Case Study 1 (Section 3.1.2) was 19 MB, the resultant database and indexes were 1.9 MB, a 90% decrease in the storage requirements.

The frequency at which samples are collected (the granularity of the data), together with what information is being stored (level of detail), greatly affects the total size of storage required. The researcher used an SQL database to store captured traffic flows for a week for Rhodes University's Internet access links as part of the analysis described in Appendix D. Traffic records were stored at five minute intervals. This data ended up consuming approximately fifteen gigabytes of storage once placed in the database – still a significant reduction from the 20 gigabytes of raw data.

The content of the data stored should also be evaluated. After further process-

[5]Structured Query Language

ing the same database was reduced to eight gigabytes, with only a minimal loss of data, which was not relevant to the aspect being studied. A further reduction was made possible by using a 15 minute aggregation instead of the original five minutes. A balance thus needs to be found between the required accuracy, in terms of granularity and storage requirements for traffic data. This is particularly important as data ages. Decisions need to be made as to whether it is necessary to maintain data with a five minute resolution a year after the event, or if fifteen minute or hourly data aggregates are sufficient. The sampling resolution of data can always be decreased during data processing using techniques such as aggregation, but it is impossible to re-acquire higher resolution samples once they have been aggregated, and the original samples lost.

Data can be stored in several stages of aggregation, for example, five minute detail for the last 24 hours, 30 minute detail for the last week, and hourly for older data. The problem of data loss due to aggregation is a problem befalling the storage format discussed in Section 3.3.2.1. Figure 3.9 illustrates the relationship between storage space requirements and data accuracy in terms of lossless and lossy storage. As a rule, the higher the sampling rate, the higher the data storage requirements will be. It may be necessary to have per minute data for the last hour, but it is unlikely that this would be needed 24 hours later or a month later.

3.3 Visualisation

3.3.1 The need for visualisation

It is important for a network administrator to be able to evaluate traffic both quantitatively and qualitatively.

Data collected by monitoring systems can quickly accumulate to unmanageable amounts, making it difficult to process in its raw numeric format. The administrator needs to be able to interpret these large amounts of data, often collected from a number of sources, at a higher macro level, in order to assess overall usage trends, for troubleshooting and for future capacity planning. The detailed data can

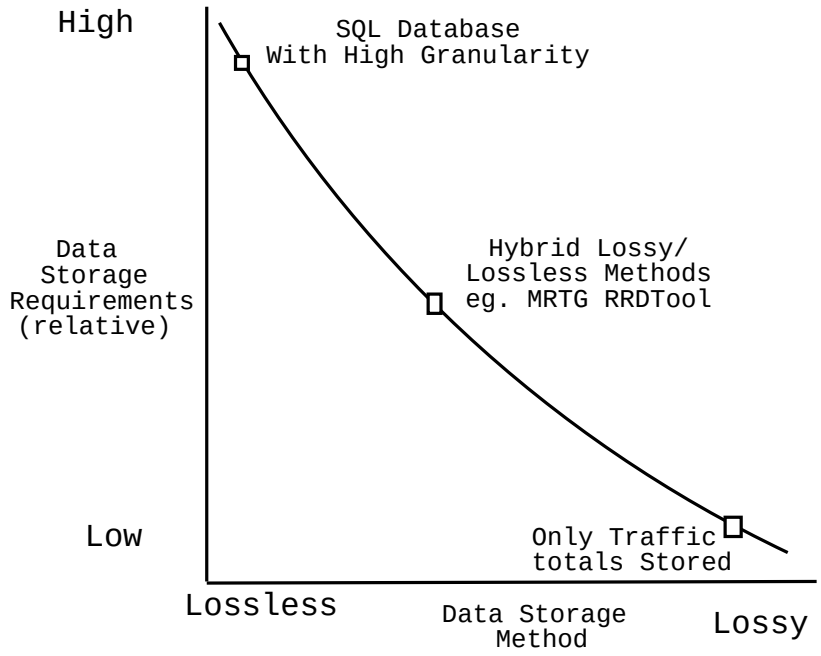


Figure 3.9: Data Storage Methods vs Space Requirements

be referred to in the event of further information being required about events that may be brought to light through the visualisation processes.

Visualisation entails the transformation of data from a numerical tabular format to a graphical representation. This allows for the concise representation and display of large amounts of data. The images produced provide a 'higher bandwidth' means of transmitting information from a screen to a viewer than the traditional display of tables of numerical data. This validates the old adage of "A picture is worth a thousand words" coined in 1927 by Fred Barnard[6] .

In a visualisation image, certain features or information worth highlighting can be enhanced using techniques such as colouring in order to draw the attention of the viewer and to aid in interpretation. Information that may not be immediately

[6]FRED BARNARD was an advertising manager in America in the early 1920's. He coined the phrase while selling advertising for placement in the trams. He originally claimed this phrase was an ancient Chinese proverb which was attributed to CONFUCIOUS, in order that people would take him seriously. The statement 'One picture is worth a thousand words' was printed in *Printers' Ink*, 10 March 1927, p. 114 (the predecessor to the current *Marketing/Communications* trade publication), this statement was an adaptation of his earlier phrase 'One look is worth a thousand words' which was printed on 8 December 1921. [67] [102]

apparent with one form of visualisation may be highlighted by another. As with many problems, in order to arrive at a solution a number of tools may be required, and the 'right tool' used for the task at hand.

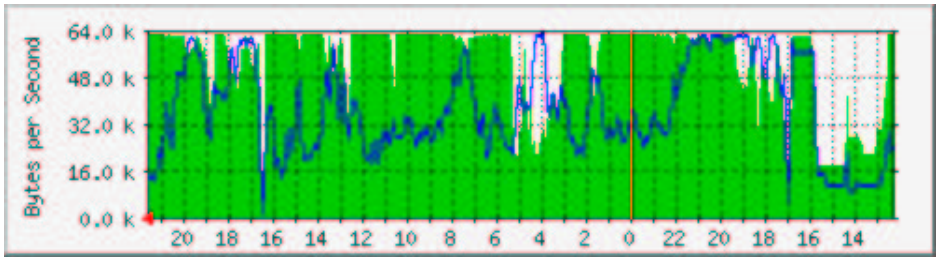
Visualisation tools are important, not only for visualising traffic but also as a modelling tool for answering 'what if' scenarios. Using visualisation tools an administrator can perform a visual comparison between the 'before' and 'after' images resulting from a simulated, or even real change.

3.3.2 Visualisation types

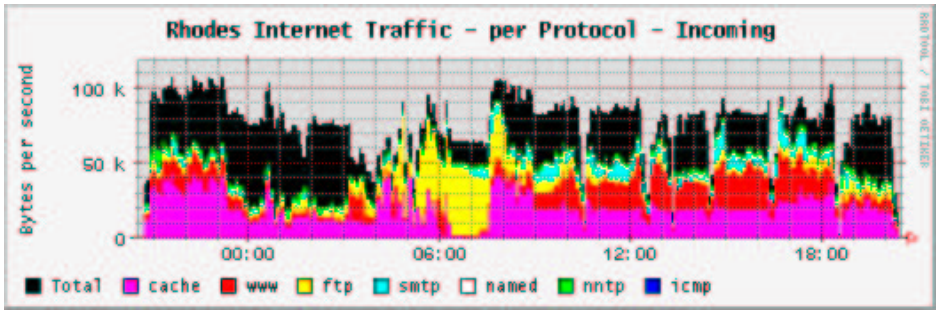
Four broad classes of visualisation are discussed, along with some implementations of each. These are dealt with in increasing order of complexity. The greatest benefit in using the visualisation techniques described, is in using them as complementary methods, and making use of the particular strengths and weaknesses particular to each. Included in the discussion are examples of the output produced by the researcher either through the use of existing tools, or using software tools that were developed by the writer for this purpose. Implementation and operational issues regarding the tools used to produce the visualisations below are discussed in Appendix C.

3.3.2.1 Simple Graphing

Simple graphing is one of the easiest and most common forms of visualisation and is consequently one of the most frequently used. It consists of a simple plotting of a traffic usage graph on a Cartesian plane (standard *x/y* graph) and is the form produced by Tobias Oetiker's highly popular MRTG [121] and *RRDTool* [119] packages. See Figure 3.10 for examples of the output from these utilities. Samples of network bandwidth currently utilised are plotted as a percentage of total link capacity. Incoming and outgoing traffic are usually plotted as two separate series on the same graph. This form of visualisation is quick and easy to implement and to interpret, and can be used to give an indication of when further in-depth analysis should be performed.



(a) MRTG output



(b) RRDTool

Figure 3.10: Example *MRTG* and *RRDTool* Output

The method of data storage used by the utilities mentioned above is a 'lossy' method. Because of the particulars of the storage format, they suffer from a substantial loss of detail for longer term records. This is as a result of the design principals stated by Oetiker [118] which allow for a fixed datafile size, and faster operation. This is due to the mechanism of aggregating records as they age, as a means of compression. *RRDTool* does however support a much more flexible format for its data-files, allowing the administrator to specify the level of detail to be maintained in the database at creation time [120].

Other implementations need not have such limitations. A system which utilises a database backend for data storage can be implemented at the expense of larger storage requirements and complexity, but this is offset by the increased flexibility and the fact that the data is then available to other applications as well.

A web-based implementation of such a system was developed by the writer, using Perl and PHP, which allows a user to browse daily, weekly and monthly graphs for any period of time (depending on data availability in the database). This system is discussed further in Case Study 1 (Section 3.1.2).

3.3.2.2 Layered Graphs

Layered or stacked graphs are used to represent information in a way that is easier to understand than simple graphs. Analysis of certain types of network traffic data, such as a protocol or subnet breakdown of total traffic, are inherently better suited to this form of visualisation.

While a simple graph is useful for monitoring spikes in traffic or link utilisation, it is not apparent what the cause of the increase in network traffic may be. A solution to this would be to have multiple graphs for various traffic sources and/or protocols. The total traffic can also in most cases can be further subdivided along the lines of its constituent protocols. In the case of IP traffic one can distinguish between TCP and UDP traffic, and more usefully between the specific protocols running on top of the TCP/UDP transport. These protocols are identified by the TCP or UDP port numbers which they use. A list of registered TCP and UDP port

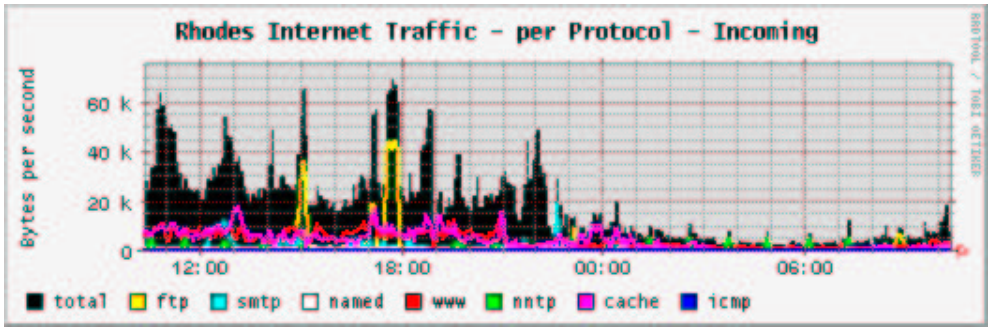
numbers is maintained by the IANA[7] [84]. These graphs can then be compared to a total traffic graph, and the peaks visually matched to determine which protocol is consuming the major portion of the available bandwidth. An alternative providing for easier interpretation would be to plot these as multiple series, represented as a fractional portion of the total, and overlayed on the total graph, an example being presented in Figure 3.11a. Such a display provides a more intuitive means of assessing what traffic constituents are having an influence on overall traffic flow.

The major disadvantage of such a system is the increased complexity in traffic accounting in order to subdivide the traffic. If a graph is to represent the traffic attributed to the full range of TCP and UDP ports, it could conceivably have a maximum of 131 000 distinct series – or separate graphs if using simple graphing. In practice the full range of ports is seldom covered, even on relatively large networks with diverse user populations and traffic, and the number requiring monitoring can in most cases be limited to ten to fifteen. Appendix B contains an analysis of traffic which also highlights ports which have a high byte count. Analysis of this traffic sample by the researcher showed that approximately 98% of TCP and 90% of UDP ports were used (See Section B.4), with the majority of the total IP traffic being concentrated in the top fifteen ports for each protocol. TCP traffic also constituted the bulk of the total IP traffic. When configuring the output, the readability of the graphs can be enhanced by the choice of contrasting adjacent colours. For better visual appeal the number of series graphed should be limited to less than twenty, as beyond this it becomes difficult for the the eye to distinguish colours. If output is going to be in gray-scale, fewer series should be considered.

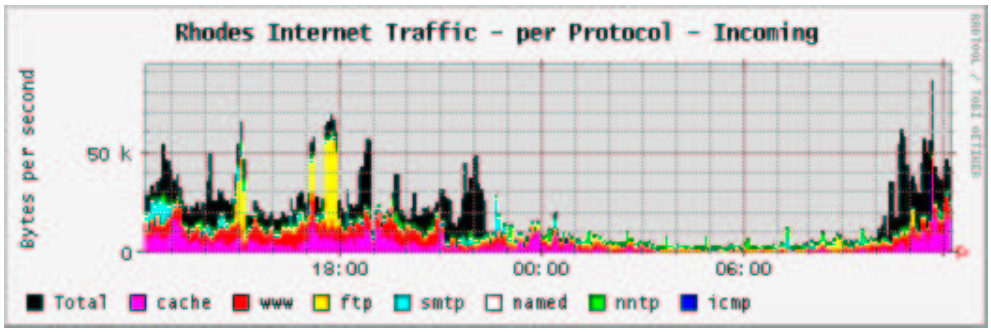
The advantage of this format is that once the initial complexity of setup has been done, it is easy to ascertain what protocols are using network bandwidth, and in what proportion. Using these as a base for visualising simulation results provides a means for being able to perform a quick visual comparison between ‘before and after’ measures. An example of such a system is discussed in Case Study 2 (Section 3.1.4).

The difference between displaying the same data as a ‘simple’ line graph, and as a layered graph is illustrated in Figure 3.11. The outputs illustrated in this Figure

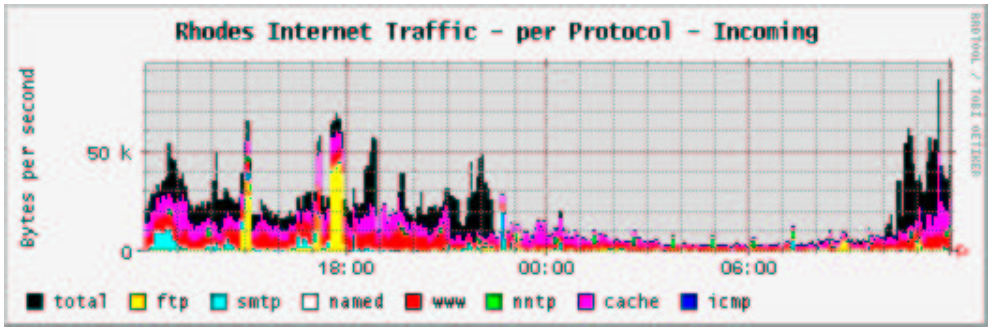
[7] Internet Assigned Numbers Authority



(a) Simple line graphs



(b) Sorted stack graphs



(c) Stacked values with random ordering

Figure 3.11: Layered graphs

were produced using the *RRDTool* package, using data collected by the system described in Case Study 2 (Section 3.1.4). Figure 3.11a shows a multi-series line graph which allows for accurate reading of the levels that each protocol monitored is consuming, but it is difficult to see the total contribution of a single variable to the total. The stacking of the variables in Figure 3.11b allows for quick visual interpretation of the makeup of the link traffic. This also discloses the 'ghost' bandwidth – the portion of the bandwidth that is made up of traffic which is not directly being monitored. In this case it was determined by the researcher to consist mostly of passive FTP transfers, which are difficult to measure due to the wide range of ports than can possibly used [144] [24][8], and to Napster traffic. The graph in Figure 3.11b does however present difficulty in ascertaining the actual value of a variable, other than via an interpolation of the value in relation to other variables on which it may be stacked. For this reason the ordering of the stack can be an important factor in producing easily interpretable images. As only the lower layers are easy to accurately quantify, traffic should be placed in order of significance (or volume), which in many cases is similar to the order of volume of traffic. Figure 3.11c shows the same data as 3.11a and 3.11b, but with the order of the layers in sequential order, illustrating the difficulty that can arise in interpreting these graphs.

3.3.2.3 Colour Maps

This visualisation method requires a more substantial amount of data in order to produce usable images, and as such is consequently more suited to longer term analysis. Colour maps are generated as flat (two-dimensional) projections of a three-dimensional (3D) mesh which represents the traffic flows over time (using time and day as *x*,*y* values respectively, and the traffic measurement as the *z* value). False colouring or shading is provided as a function of the 'height' of traffic on

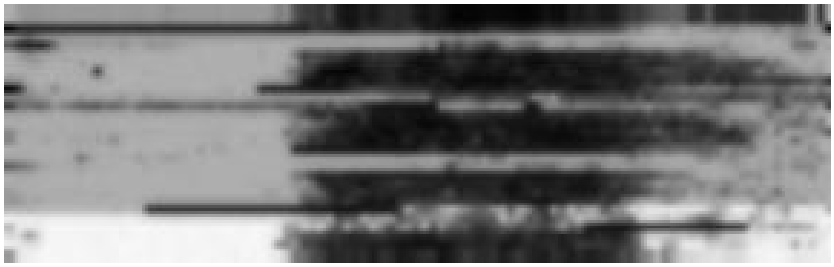
[8]Passive FTP operates by the server providing the client with a high-numbered TCP port to connect to in order to receive data. The connection is then originated from the client side, using the next allocated port on the client host. This provides a fairly random combination of ports, hence the difficulty in monitoring passive FTP data traffic. This differs from the standard FTP operation where the client provides the high-numbered port number to which a connection is initiated from TCP port 20 (ftp-data) on the server.

the three-dimensional mesh. The outputs of this method allow for better long term trend analysis than traditional (*x*,*y*) type Cartesian graphs, due to the fact that the temporal data is represented in an order in which comparisons can be made. Of particular interest are the trends visible with the data aligned by day of week. Time of day provides another insightful view, and this is particularly suited for 3D visualisation as discussed in Section 3.3.2.4.

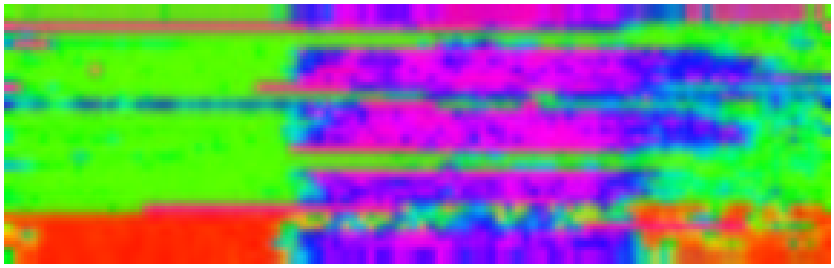
Figure 3.12{a,b} show an example of the same data displayed as a grey-scale image, and as a false colour map. The accompanying gradient key is illustrated as 3.12c. These colour maps are produced by a system developed by the writer which interacts with the database used in Case Study 1 (Section 3.1.2). The mapping of colour values to a point is performed by determining the appropriate pixels value from a pallette using the data value as an index. This is similar to the method by which the 'brightness' value of each pixel in the grey-scale image is derived. Each of these images has its value: 3.12b highlights some of the low-points of traffic which are not immediately apparent from 3.12a due to the darker colours not being as easily distinguished by the viewer's eye.

Interpretation of these images shows a marked increase in the throughput after hours (as indicated by the lighter grey/green sections respectively on Figure 3.12{a,b}). Also of interest is the horizontal banding present in both 3.12{a,b}, which is as a result of the increased throughput over weekend periods, due to the lower usage of the line. The white/red sections of colour on the lower sections of the graph clearly show the result of the upgrade of the link capacity from the regional ISP hub to the ISP backbone, which occurred on the 24th of September 1999. This is highlighted particularly well using the three dimensional visualisation technique as described in Section 3.3.2.4, and illustrated in Figure 3.15. Figure 3.2 provides another view of the event as produced by the researcher's monitoring system. Further details of this system and the measurement of this data used are given in Case Study 2 (Section 3.1.4).

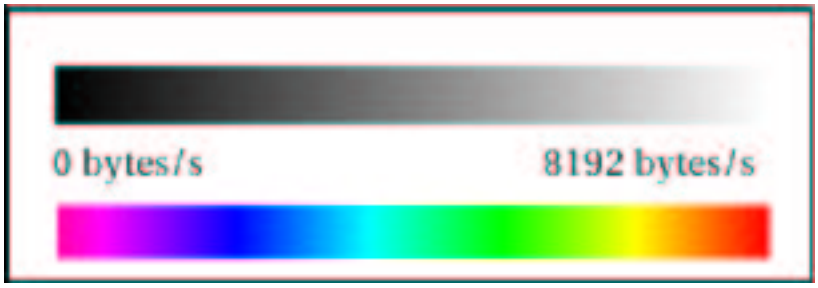
An alternative to the height-field projection method is to produce a 2D projection of the 3D mesh contours. These contours are then given a false colouring in a similar manner to the colouring of pixels in the height-field projection generation. This is useful in determining the actual rate of change of utilisation of bandwidth,



(a) Greyscale shading



(b) False Colour Mapping of Greyscale image



(c) Gradient Key for (a) and (b)

Figure 3.12: Example Colour Maps

Note: These images are visualisations of throughput tests as described in Case Study 2 for September 1999. Vertical axes are days of month with the 30th being at the origin of the *y* axis. The *x* axis is the time from 00h00 to 23h55.

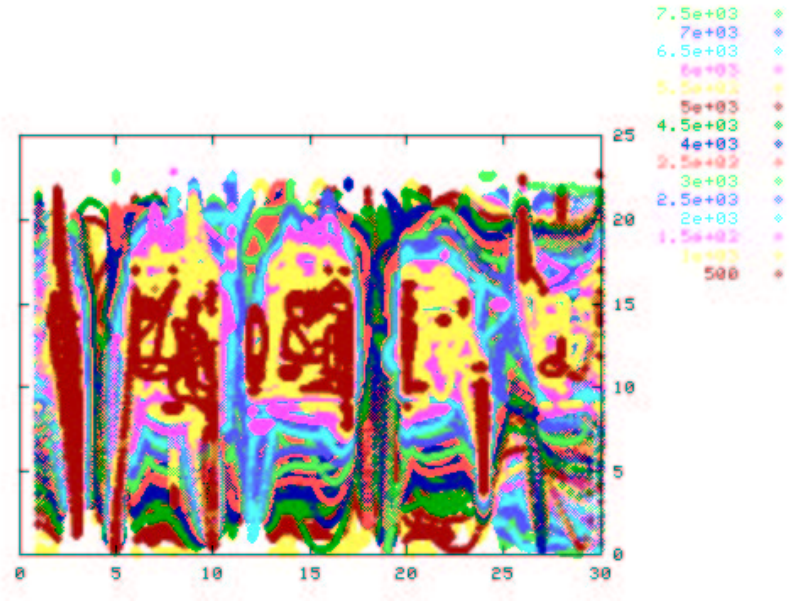


Figure 3.13: Contour map

over the colour gradient seen on colour map views. Re-occurring spikes in usage will be clearly represented as ridges of closely grouped lines. These images can be interpreted in a similar nature to standard geographical topographical maps. Figure 3.13 illustrates an example of a contour map. The image displayed by no means ideal. The tool used to generate the data was not ideally suited for this type of data processing, and development of a suitable tool was outside the scope of this work, as it was felt by the researcher that the colour maps provided a stronger means of interpretation.

3.3.2.4 Three-Dimensional Landscapes

Using longer term historic data, 3D landscapes can be generated as a means of visualising traffic. These landscapes can be examined for features such as valleys, plateaux and crevices, each of which correlate to certain traffic and bandwidth utilisation characteristics. Ideally such landscapes should be navigable to allow for viewing from any angle, giving the viewer the freedom to explore the model. These models essentially present the same data as the colour maps of Section

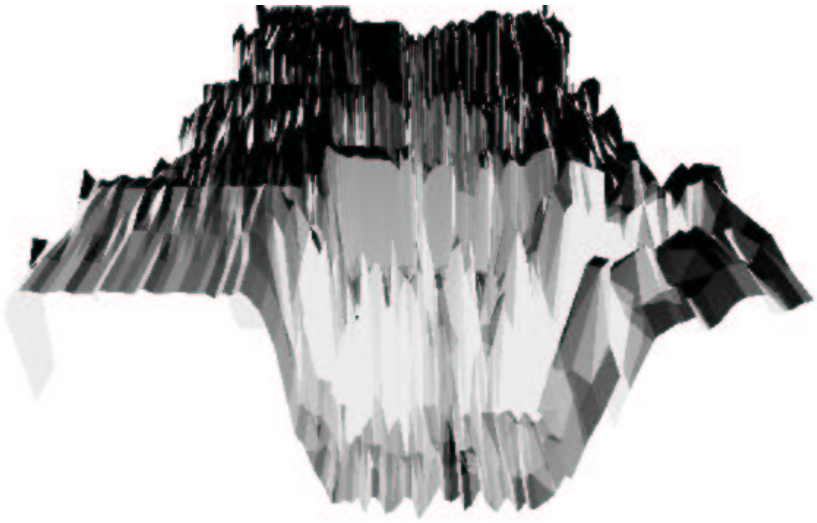
3.3.2.3 but with the added benefit that the viewer is able to clearly see the three-dimensional nature of the data. Data such as measurements of bandwidth availability or usage lend themselves to three-dimensional visualisation, as the viewer is able to correlate the visual 'height' of data displayed directly with the quantity of the bandwidth measurement. Three-dimensional landscape visualisations can take a number of forms, each with their own benefits:

Static: These are the simplest form of 3D landscape and are often sufficient to allow the viewer to ascertain overall trends. However the viewing angle from which these are produced can impact greatly on their usefulness, and informational capability. Further images of the model displayed in Figure 3.14 are available on the accompanying CD-ROM. Figure 3.15 is an example of the same data as the previous figure, but presented from a different angle and view, having been elongated , so as to accentuate the information for each day displayed.

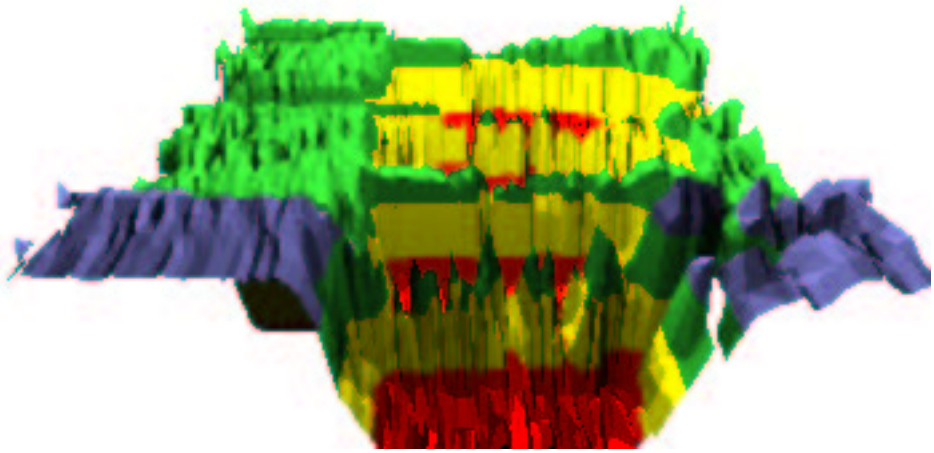
Interactive: Models that can be manipulated by the viewer in real-time allow a further degree of freedom to explore features. These are however computationally 'expensive' to create and operate, although they provide a unique means of interacting with the model. VRML[9] was found to be a relatively lightweight format for the creation of models, and allowed viewing without particularly specialised software. Depending on the modelling package used to create the three-dimensional landscape, an interactive model may already be present as part of the process of obtaining a statically rendered image. An example of a VRML model of the landscape shown in Figure 3.15 is contained on the accompanying CD-ROM.

Pre-rendered: An animated sequence of static images of a model can be pre-rendered by a visualisation application. What the animation actually displays determines its interpretation value. This technique allows for batch type processing of data, while still affording the viewer a more extensive view of the landscape, although more limited than an interactive view. Two

[9]Virtual Reality Markup Language



(a) Simple 3D Landscape



(b) Landscape with coloured Strata

Figure 3.14: 3D landscape visualisations

Note: 3D visualisation of the same data as shown in Figure 3.12. The view presented is from the 30th of the month looking towards the 1st. The left hand side is equivalent to 00h00 through to 23h55 on the right. Height is a measure of the available bandwidth as measured in Case Study 2. The coloured bands in (b) are at 2Kbyte/second intervals.

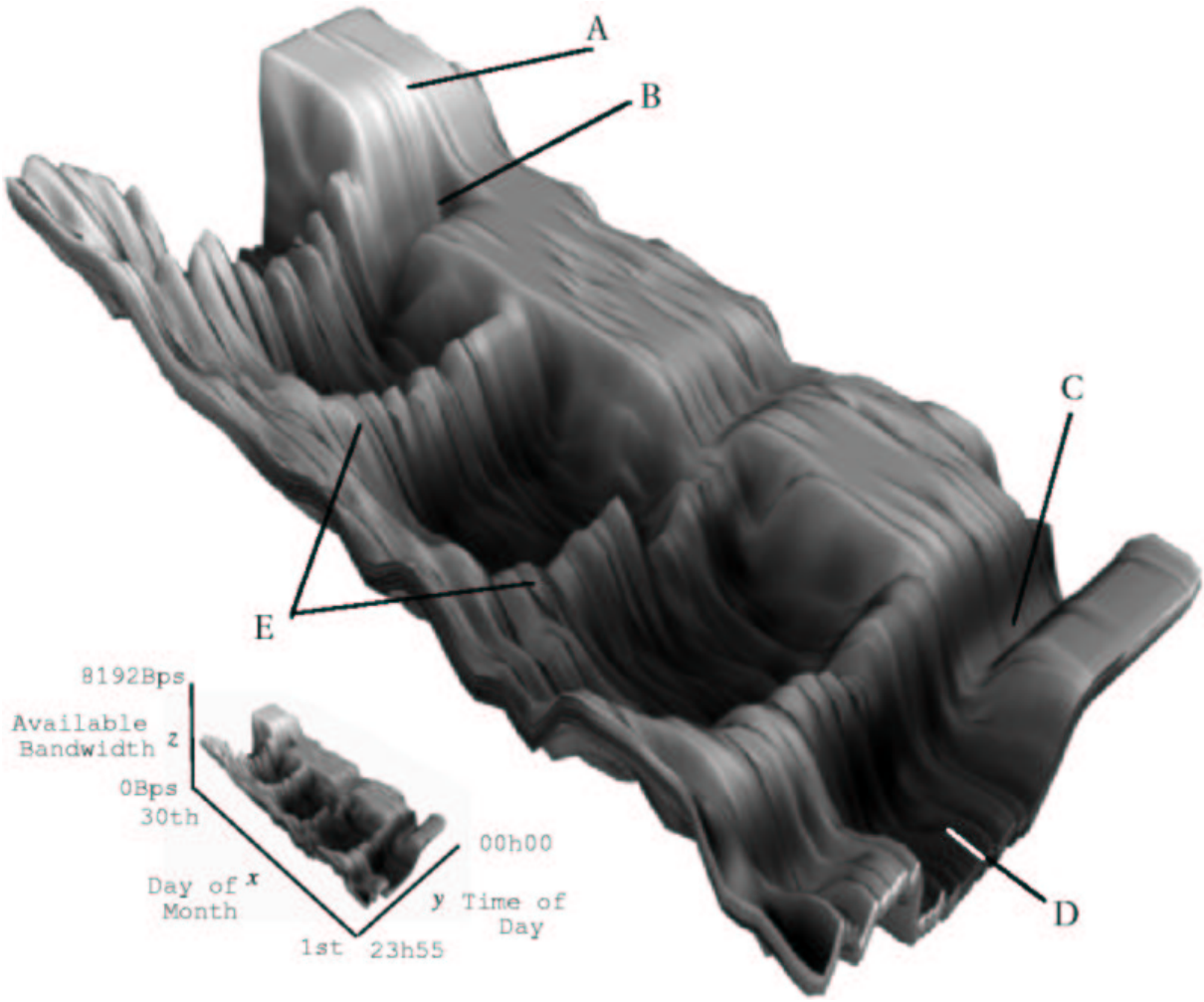


Figure 3.15: High quality gradient shaded 3D landscape

useful animation views are a flyover of the model where the view moves down the centreline of the model, and the rotation, where an oblique view of the model is rotated through 360 degrees. Both of these animation paths allow for the viewing of the majority of a model's salient features. Example animations are contained on the accompanying CD-ROM.

The creation of a three-dimensional visualisation provides the ability to navigate through the model and view features which may not immediately be apparent in the flat representation, or are hidden in the initial static 3D view which may be presented. An enhancement to the system is to provide colouring of the landscape by defining strata, which correlate to various traffic levels. These provide an easily understandable visual indication as to what the level is at a particular point of the landscape, which is particularly useful for ascertaining the depths to which crevasses in a plateaux descend. Colours should be chosen as to contrast well, particularly if the images are going to be output as grey-scale (such as when printing). This is illustrated in Figure 3.14. The tools and techniques developed by the writer used to produce these landscapes are discussed in Appendix C.

These strata can also be blended into a smooth gradient which can provide a better visual appeal to the image, but at the cost of substantially increased processing, and a significant decrease in the ability to obtain accurate values off the image. This technique results in both the use of colour and shape for conveying information to the viewer. Figure 3.15 illustrates the same dataset as Figure 3.14, but after some smoothing, and the use of a gradient shading. A coloured gradient can be applied to these models using the same technique as with the colour maps. Figure 3.16 shows the effect of performing false colouring on the same dataset.

A number of interesting features in the dataset are highlighted through the use of three-dimensional modelling. These are marked on Figure 3.15 as follows:

A A lightly coloured plateau is prominent in the upper centre of the image. This shows the substantial increase in throughput achieved after the line and equipment upgrade of the target host's ISP link between Port Elizabeth and Cape Town on the 24th of September 1999. This should be compared to the

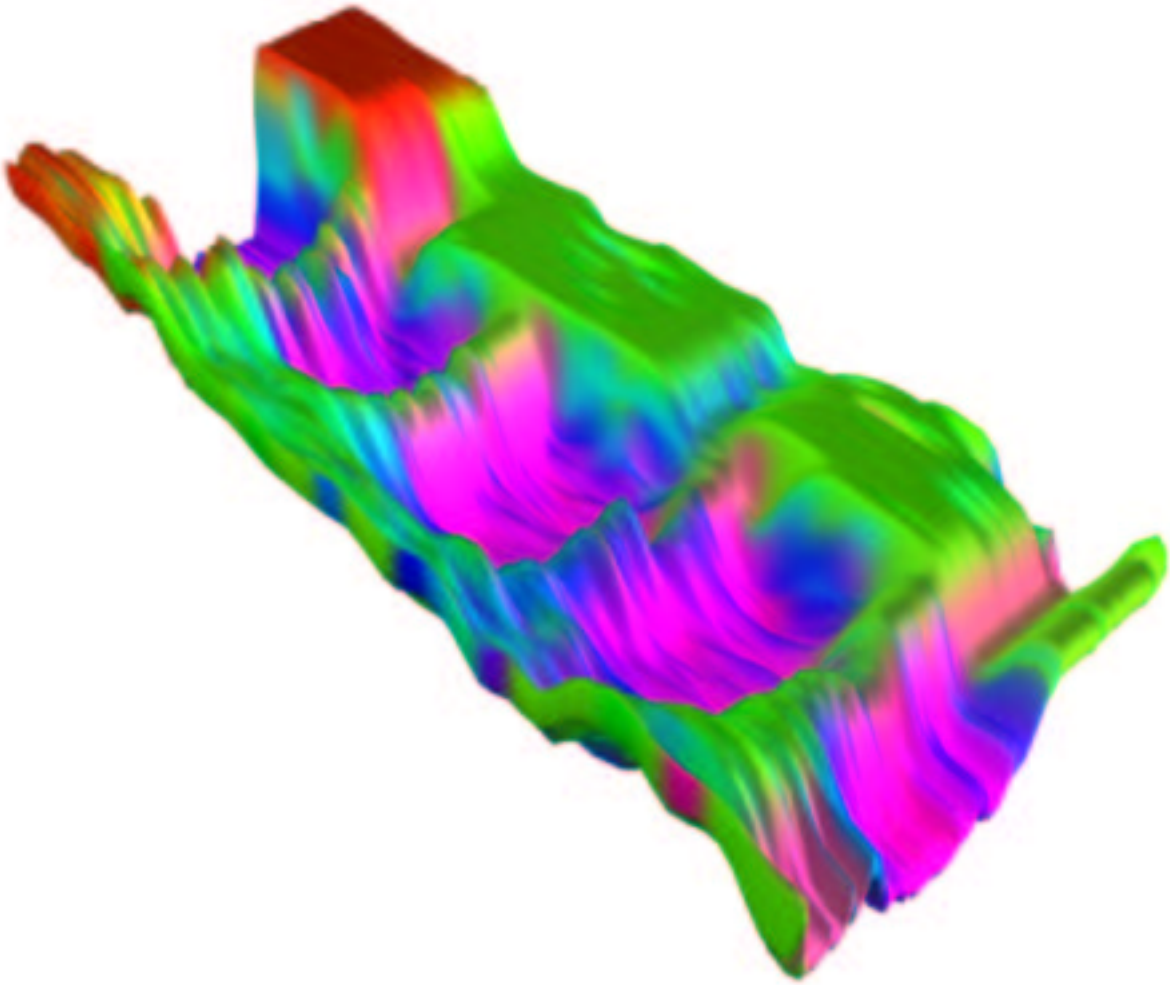


Figure 3.16: High quality three-dimensional landscape with false colour gradient shading.

Note: The gradient used is the same as that used in Figure 3.12.

previous 'maximum' achieved. These maximum areas are between 00h00 and approximately 06h00.

B A deep trough or crevasse in the landscape indicates a lack of bandwidth, in this case due to an outage caused by a line upgrade. By the time the outage was resolved, the daily usage load was already present, and as such the increase in bandwidth was only apparent the following evening.

C This point shows the result of a period of heavy overnight traffic on the line, and hence decreased available bandwidth for the throughput test. The resultant valley is noticeable from the otherwise relatively level plateaux during the early hours of the morning.

D During work hours the network path between the two test sites was close to saturation point, with throughput results often sub 1Kbyte/second. This point shows the deep troughs with relatively flat floors, indicating this lack of bandwidth. The small hills on the trough floor correlate with mid morning and afternoon tea breaks.

E The sharp peaks, two of which are indicated by this label, are the due to the increased bandwidth available over weekends. Despite a high number of dialup users during the weekend, the bandwidth is still higher than during the week, most likely due to the fact that users are constrained both by the number of modems available at the POP[10] and the limitation of the maximum speed of the modems. During the week there is little dial-up activity during office hours, but most businesses are linked to the POP by means of higher speed digital lines[11].

The disadvantage of this form of visualisation is that it is computationally intensive to produce the rendered 3D views, in particular the animations and interactive models which allow the viewer to observe the entire landscape model. Sample animations of the models presented above are included on the accompanying CD-ROM.

[10] Internet Service Provider's local dial-up Point of Presence

[11] Further details of the network topology related to the testing can be found in Case Study 1 (Section 3.1.2).

Landscape visualisation does have several benefits to its credit. It is visually appealing, and has been found by the researcher to be relatively intuitive to interpret, often more so for non technical people than simple graphs. This is particularly true when trying to convey trends, as an entire trend can be captured in a single all encompassing image rather than several separate graphs.

3.4 Other Interpretation Methods

Methods other than the visualisation techniques previously discussed are available for interpretation. Many of these rely on large amounts of numerical processing of captured monitoring data. One radical interpretation tool is *Peep* [58], developed by Michael Gilfix of Tufts University and presented at the December 2000 USENIX-LISA Conference. This system uses a monitoring system and an auralisation engine which translates monitored events into audible output.

3.5 Summation

The success in using visualisation techniques on data collected hinges on three facts:

- The quality of the underlying data that has been collected. This data should be as complete and as specific to the visualisation requirements as possible. Often if one is unsure of what needs monitoring, it is better to gather too much information than too little, as it can always be subjected to some form of post processing in order to obtain the relevant information. Otherwise it is impossible to provide a good visualisation based on incomplete or misrepresentative data.

- The right visualisation technique should be selected, in order to satisfy the task at hand. Each of the techniques previously discussed has its own merits and disadvantages. These should be understood, and combinations of techniques used where applicable. For example, it would most likely prove

fruitless to use the 3D landscape techniques for monitoring current or 'real-time' bandwidth utilisation due to the fact that there is unlikely to be sufficient data to be able to create a viable three-dimensional model.

- Correct interpretation of the output of the visualisation can determine the usefulness of the effort expended in performing the monitoring and visualisation. Depending on the technique being used, some practice may be required. An analysis such as that provided for Figure 3.15 may not be immediately apparent to the observer, and often knowledge is required of other external events that may have influenced either the monitoring process, or the variable being monitored.

The tools and visualisation techniques that have been discussed are not an exhaustive list of available resources. The visualisation techniques are broad classes which may encompass many refinements and combinations of techniques. In particular the work done by the researcher on three-dimensional landscape generation as a means of visualising longer term records has mostly been proof of concept work and consequently has not been explored in full. This technique could benefit greatly from further investigation.

Armed with the information obtained by monitoring and analysis, the administrator can prepare to implement the traffic management techniques discussed in the following chapter. Continued monitoring is important in order to be able to assess the effect, positive or otherwise, that the traffic management is having.

Chapter 4

Bandwidth Management Strategies

> "Resource-constrained environment" [are] fancy Pentagon words that mean there isn't enough money to go around.
>
> - GENERAL JOHN W VESSEY JR[1]

Network bandwidth, particularly on WAN or Internet links is currently a scarce resource, and is likely to remain so for the foreseeable future. These links are also often the primary sources of congestion on the Internet, with the high capacity backbone links suffering relatively little congestion [117].The major reason for this is that as fast as organisations are able to upgrade their networking infrastructure, often at great cost, technology and user demands push it to the limits. Recent years have seen the explosion of the World Wide Web, from its humble beginnings serving only textual content, to the current plethora of bandwidth hungry 'rich content' such as streaming audiovisual data and increasingly complex (and consequently larger) web pages. Network infrastructure costs are often a significant portion of an organisations annual budget, and as such they need to be careful managed. This chapter addresses several strategies that can be implemented in order to better manage bandwidth on the network as noted in Chapter 1. To this end number of categories of techniques for bandwidth management are discussed.

Each of the strategies discussed below has its particular strengths and weaknesses, and some are often best implemented in conjunction with others. Strategies that

[1]US Army, Chairman, Joint Chiefs of Staff. New York *Times* 15 July 1984. [157]

are discussed are:

1. Quota Systems
2. Traffic Shaping
3. Queueing and Quality of Service
4. Active Content Control
5. Limitation of Service
6. Threshold Management
7. Caching Proxy Servers
8. Compression
9. Network Address Translation
10. Satellite Connections
11. User Education

4.1 The ying-yang of bandwidth management

Developing and implementing a bandwidth management policy for an organisation is a balancing act between the two extremes of completely free access, and a completely draconian set of access policies. Both of these can lead to poor performance/utilisation of the available network bandwidth.

With unrestricted access there is a loss of performance due to line congestion and increased latency as a link approaches saturation. A policy with too many restrictions can create a situation in which there is bandwidth available, the link has a low latency and is uncongested, yet poor performance is perceived by users as their access attempts are hampered by the restrictions. The perception of poor performance, as ascertained by the researcher, is caused by a number of factors:

- Restrictions interfere with day-to-day operation and use, by being restrictive to the point of creating an unusable resource. Users will often rather not make use of a resource where they are hampered in its use.
- Restriction systems often impose a noticeable latency in the response of even legitimate requests because of the often large number of restriction rules that are required to be processed.

Restrictive policies do however have their place, particularly in the case where the bandwidth saturation level of a network link has been reached, and measures need to be put in place to bring usage down to more reasonable levels, or alternately where a low link congestion must be maintained due to the primary nature of the link (eg. a supermarket chain piggybacking Internet access over data lines primarily installed for EFT transaction systems[2]). In most cases restrictive policies are implemented as an emergency stopgap measure in order to try and solve an unexpected demand for bandwidth. Heavily restrictive policies are not a recommended general bandwidth management practice because of the limiting of the usefulness of the resource to end users.

An Internet link for an organisation is a costly resource both in terms of the initial outlay for equipment as well as the monthly levies required by the access and telecommunications providers. As such it is in the organisation's best interest to make as full and fruitful use of the resource as possible. Users should therefore be encouraged to make responsible use of the resource, as with any other within the organisation, such as telephone systems or photocopiers. High utilisation of the Internet connectivity resources can be directly correlated to a greater 'benefit per unit cost', yielding a greater return on investment than if the lines were to have low usage.

The degree of restrictiveness of a management policy is in many ways determined by the type of organisational environment in which it is intended to be implemented. As a general rule, corporate environments are more tolerant to the introduction of restrictive policies than academic institutions such as universities and

[2]This is the case with the Pick 'n Pay group in South Africa, who have used their existing ETF infrastructure to deliver Internet and Intranet connectivity to their stores via the primary Internet connection located at head office [20].

research organisations. The motivation behind the organisation's need and use of its Internet link also needs to be taken into account when a management policy is put together.

4.2 Traffic Accounting Strategies

In order to perform any kind of meaningful network management, data regarding the traffic on the network needs to be gathered for analysis as discussed in Chapter 3. Traffic accounting mechanisms can be grouped into three broad categories:

4.2.1 Sub-Network Accounting

This is the accounting of network traffic back to a particular workgroup, or group of machines. This is usually performed at the IP level, by collecting statistics for a range of IP addresses or appropriate subnet for a workgroup. Sub-network (sub-net) based accounting is best suited to the situation where subnetting is performed along the lines of organisational structures (often through the use of VLANs[3]), rather than physical network topology.

4.2.2 Host Accounting

Network traffic statistics are collected on a per host basis, associating traffic with a particular IP address, which is unique to every host within an organisation. This is useful where there is a fixed association between an IP address and a physical host, such as where addressees are either 'hardwired' or handed out using statically configured Dynamic Host Configuration Protocol (DHCP) [46]. Where DHCP is used with a pool of allocatable IP addresses, this attribution of traffic to a particular host can become significantly more involved, as hosts can change addresses based on free entries in the DHCP pool at the time of lease negotiation. However this complexity is not insurmountable and can be overcome through the use of long

[3]A VLAN is a technology which allows multiple logical networks to run over the same physical infrastructure, and is usually implemented on a switched network.

client lease times (which result in the same address being handed back to the host during a given time frame despite rebooting of the client system [46]). An alternate solution can be created by using the Ethernet MAC address (or other appropriate hardware address of a machine) of the networked host as recorded by the the DHCP server or other means. The latter can, however, be difficult to achieve and implement on large sub-netted and/or switched LAN's.

4.2.3 User Accounting

Traffic is accounted back to a particular user. This is especially useful for traffic passing through a proxy server (usually a caching web proxy) or firewall that requires authentication. Most methods for attributing traffic to a user are based on the assumption that, no matter where on the network the user's traffic originates, the same username and password are used. This permits for the development of a flexible system for collecting usage data for a user on a network, even though they may be accessing the network form several different points.

An example of this is the method used at Rhodes University to account for web traffic originating from students in the three large public access laboratories. An 'ident server' [161] is installed on the Windows NT machines, which when queried reports the student number of the user currently logged in at that workstation. All these laboratories use the same credentials for logon, and so the relationship between a student and the student number returned by the system is fixed. When a request is made by the students' web browser to the web proxy server, the proxy server queries the ident server on the client system in order to ascertain the student number (which is the same as the user login) of the user making the request. This information is included in the proxy log files together with the details of the web request. Using this system, web traffic can be accounted back to a user regardless which of the nearly 250 machines in laboratories they use. The same method for identifying users is also used by the RUCUS proxy system as described in Section 4.3.2. An alternate to this transparent method would be to implement explicit user authentication to use the proxy server. This authentication can either be a separate database, or can be tied into the centralised authentication system using

technology such as LDAP, as described by Halse [63].

The ability to account network traffic back to a particular user (whether an individual or group account) allows for traffic to be 'billed' to the user. This is irrespective of whether there is actual monetary transaction is involved, billing against a virtual account such as a traffic quota (as discussed in Section 4.3), or merely as a measure to make users aware of their traffic usage and its impact on other users.

This system is similar to the common business practice of accounting for staff telephone calls. Even if this is not in reality billed back to them, it is often enough to make them aware of the costs of using the resource. It has been found by the writer that many users are completely unaware of the effect that their traffic may have on network resources as a whole. When usage is presented to them in terms which are commonly understood, such as percentage of total traffic, surprise is often expressed as to the quantity and impact of their actions. The writer has experienced several successes in combating network 'abuse' through this method of making users aware of usage, and its effect on other users on the system. Peer pressure is also a very valuable tool in this regard. If usage statistics are posted in a public forum such as notice boards, Intranet website or local network news server, peer pressure from those adversely affected will often persuade the abuser back into line. Individual user's privacy concerns may need to however be addressed before posing of usage reports in such a public forum. Often a summary report indicating traffic totals, rather than content is sufficient.

4.3 Quota Systems

Quota based traffic management systems work on the basis of allocating a fixed quantity of 'credits' to a user, which are valid for a given period. These credits can be specific to certain types of traffic, or for a particular application. The premise behind the quota system is that some form of penalty is introduced once the quota is exceeded, possibly with the penalty increasing as further limits are exceeded. The implementation of such a penalty system has been found by the writer to be most effective when some form of progressive penalty scheme is introduced

as opposed to a flat denial of service when the quota is exceeded. One form of progressive penalty is to perform 'traffic shaping' to slow down traffic for the user (this is discussed in further detail in Section 4.4). As successive quota limits are exceeded, the shaping becomes more restrictive, with the final result of the traffic being denied. For a discussion of such an implementation refer to Case Study 3 (Section 4.3.2). Another option is to provide an increased monetary charging scheme, with traffic over a given rate charged at a correspondingly higher rate, dependant on the quota level exceeded. This is similar to the escalating charging scheme used by many municipalities during times of water shortages, where, after a given number of units, the charge rate rapidly escalates [44].

The calculation of a reasonable baseline quota for use in a system is essential to the overall success of the implementation. A quota should be determined so as to have a minimal impact on the majority of users, while rapidly and noticeably affecting network abusers. A preferable method to merely guessing is to perform some medium to long term analysis of user traffic patterns, and from this deduce what a reasonable limit would be, both for a non-effect baseline and as an absolute maximum. These limits should be periodically reviewed, particularly if usage patterns change or if bandwidth upgrades are made. The administrator must also make a decision with regards to the user having a single unified quota for all traffic, or if quotas will only be based on specific types of traffic, with each type having a separately maintained quota, for example web based and non-web based traffic quotas. The type of quota selected also impacts on the sizing of the quota granted to users. Table 4.1 shows quota levels that are implemented in systems the researcher has experienced, mostly systems developed at the University, and a commercial product, Trend Micro's Web Manager [169] [20] with the default quota setting for comparison.

4.3.1 Running Quota Systems

The running of a quota-based system requires careful consideration in order to be effective. One of the primary concerns should be how often to update the quota data. This is mainly determined by the required speed of response by the system

Table 4.1: Examples of implemented traffic quota sizes

System	Quota	Details
Trend Web Manager	30MB/week	Single level quota, Access is denied once this is exceeded
Rhodes Web Quota	49MB/week 28MB/week	Thresholds are set which result in approximately 10x and 5x slowdown in performance
Rhodes Non-Web Quota	70MB/week 35MB/week	Threshold limits which result in a 4x and 2x decrease in performance respectively
RUCUS Web Quota	42MB/week 28MB/week 14MB/week	A sliding scale of increasingly delayed access culminating in the denial of service once the maximum is reached

to a user exceeding their quota. While the ideal would be to have almost instant feedback, the reality is that it is often impractical to process accounting logs in near real time, as these may be located on a completely different system to that running the quota control. More frequent updates also need to be evaluated in the light of the increased processing overhead that would be required, and the impact this may have on other systems. A realistic time frame is to update quotas on a daily basis. This does however allow the perpetrator of the network abuse to have an entire day to continue making use of resources before being restricted. This 'lag' will be the case whatever update interval is chosen, as the quota updated will always be one interval behind real usage. Quota updates should be fully automated in order to maintain accuracy and integrity, but should provide a facility for manual intervention in the case of exceptions, such as resetting or adjusting a quota.

The processing of log information for quota purposes should not interfere with other applications analysing and processing the logs. It is also not desirable for any systems such as proxy servers to suffer from any, even temporary, disruption as a result of new restrictions being activated following quota processing. If this is unavoidable, restriction reloads should be scheduled for a period where very little or no user traffic is anticipated. Decisions also need to be made as to whether new restrictions should affect connections already in progress when they are introduced. This is of particular concern where a new set of restrictions may result

in a user's download being aborted in mid transfer.

4.3.1.1 Calculation

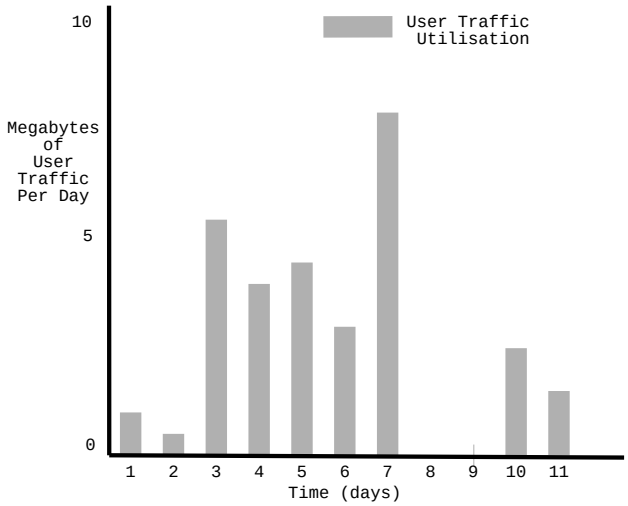
The process of calculation of the traffic bill for a user is an important part of the development and implementation of a quota system. Decisions such as what constitutes billable traffic need to be made: are some sites to be regarded as 'free'? The complexity of these decisions is directly related to the complexity of the actual accounting process for generating the summation of a user's traffic. Incentives can be added to the billing process, such as giving off-peak traffic a lower weighting in order to encourage off peak use, (in a similar fashion to off-peak telephone rates) to encourage use of the expensive infrastructure that would otherwise be predominantly dormant after normal work hours. Another incentive is to use variable billing rates depending on sites, with 'work related' sites having a very low weighting or even a zero weighting for sites related to core business. The problem with this is greatly increased technical complexity, but it is still feasible. A further consideration may be to take charging rates from one's service provider into account, as certain types of traffic for example, access to software mirror or news sites located on the ISP backbone, may be offered at a discounted rate (in the case of per unit traffic charges). These savings can be incorporated in the quota charging and calculation.

4.3.1.2 Quota Renewal

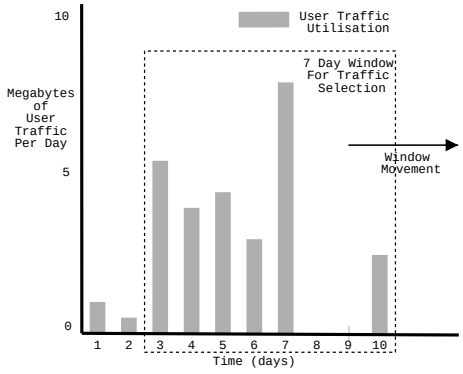
The process for renewing and/or replenishing user quotas presents a number of options:

Automatic Renewal: The quotas are automatically renewed after a given time period. This is similar to the method used in Case Studies 3 (Section 4.3.2) and 4 (Section 4.3.3), where a time period (sliding window) is used, and the user quota level is determined by averaging total traffic during the window over the period of the window.

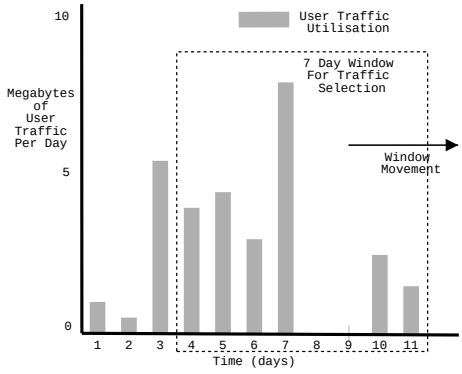
The advantage of this method is that it spreads the effect of a short periods of high utilisation over the entire window. This was found by the researcher to be regarded



(a) User traffic



(b) Sliding Window principle



(c) Sliding Window progression

Figure 4.1: The ‘sliding window’ principle for quota allocation and renewal.

by users as a fair method, due to the fact that people's usage is not constant, which may result in a number of downloads on one day, followed by several days of minimal traffic. Provided that the download on a particular day does not exceed the total quota allocation for the window, penalty will result. If the usage total for a given day exceeds the quota allocation for the entire period, a period of decreased performance will be experienced by the user, which (providing there are no other excessive downloads) will return to normal levels once the day in question passes out the window. The method of averaging the traffic over the window period allows for the inevitable pattern of users having a day of 'heavy traffic' followed by several days of little or no traffic. This pattern can be seen in Figure 4.1a. Figure 4.1{b,c} illustrates the process of calculation of the windowed traffic periods, as only traffic falling into the window period (as indicated by the dotted box) is used in calculating the quota level for the user. The result of this is that although there is no explicit renewal of the quota, it is effectively being increased by an amount equivalent to the maximum threshold level divided by the window period.

Fixed allocation: A user is allocated a fixed number of quota credits for a given time period. These are decreased as traffic is billed against the quota, and access is denied once credits have been exceeded. A variation of this is that once the quota reaches a pre-determined minimum value, the user may apply for a replenishment of the quota.

Two issues that need to be considered with regards to the quota renewal are the frequency of update, and whether unused quota credits can be accumulated or banked. The latter can result in problems when users are away for a period of time and then have the perceived legitimate ability to abuse the network on their return due to surfeit of credits that have been accumulated. The sliding window method, discussed above, provides some measure of protection against this.

With regards to renewal, the period should be large enough so as to allow for averaging out of users' uneven usage during the quota period. It should not however be too long, as it is often difficult to accurately predict user network utilisation over long periods of time. Long periods also relate to a slower response time to modifications that may be introduced by the administrator. The period of renewal

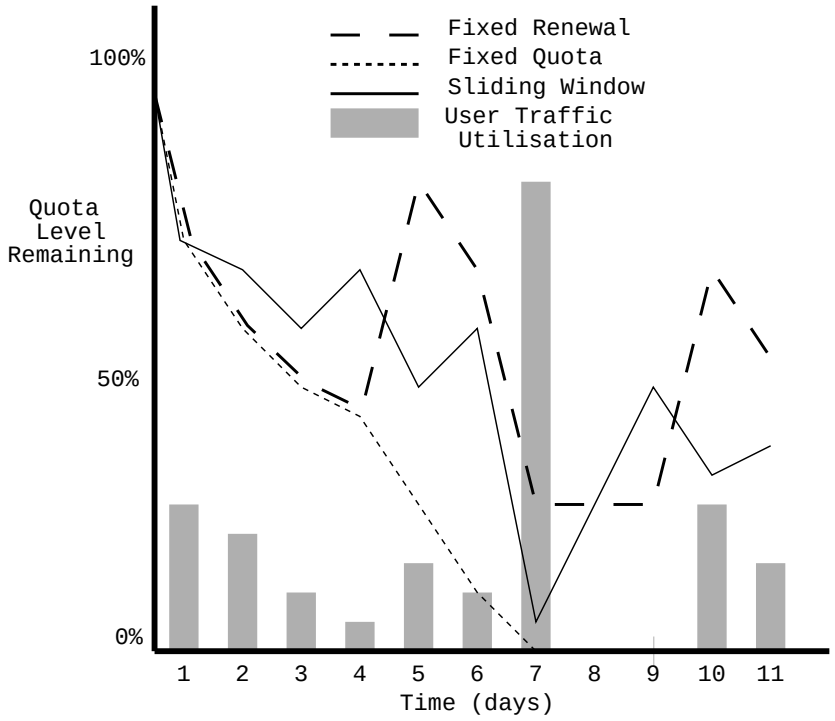


Figure 4.2: Nett effect of Renewal vs Fixed Quotas

should also take into account the quota sizing and the effect that this will have on user traffic. Figure 4.2a illustrates the comparison of the quota replenishment methods discussed.

The sliding window renewal method deals with the reduction of a user's quota due to user traffic, but compensates for this as the window moves and the quota is renewed. This effect is shown by the fact that where by day seven the user has already exhausted the fixed quota, and therefore is denied access for the remainder of the period. In the case of a quota that is renewed (in this case using the sliding window method), the quota level has dropped significantly by day seven but is replenished over the following two days. The effect of a fixed rate renewal (indicated by a dashed line) and being renewed every five days is also shown, with these periodic top-ups being the only form of renewal. It is evident that each of these methods has its benefits, and the administrator would have to select the method most appropriate to the particular situation it is being applied to.

Case Studies 3 and 4 discuss two quota based implementations and the effect they

have had on traffic at Rhodes University.

4.3.2 Case Study 3: RUCUS Proxy Web and FTP Quotas

Why Quotas?

RUCUS is a multiuser Unix machine, run by the Rhodes University Computer Users Society, supporting almost 600 users. Soon after the University implemented a quota based management model for web traffic, RUCUS users experienced a serious degradation in performance in accessing web pages. The cause of this, was that although RUCUS was a multiuser machine, it was being governed by the same IP based quota system as single user workstations were. This was compounded by the fact that certain users were performing large downloads on the system, to the detriment of the other users. The solution to this was to implement per user quotas for web traffic originating from the machine. An agreement was also reached that once this was implemented, RUCUS would be removed from the University's quota system (provided that traffic levels remained reasonable), and left to police its own bandwidth utilisation.

Implementation

The first step in managing the problem was to install some means of tracking local user traffic, and to do so a local Squid [113] caching proxy server was installed. System configurations for applications were pointed to use this cache server rather than the campus proxy server. System firewall rules were added to prevent users from attempting to circumvent the use of the local proxy server. The campus proxy was configured to reject requests originating from RUCUS and not resolving to the `squid` or root users. Squid provides detailed logs of web and FTP requests [181, part 6], along with the username of the user making the request, which is obtained by using the Unix authentication (*ident*) service [161].

Once logs were available, the next step was to develop a means of summarising and storing this data in order to build up quota information. Proxy logs are parsed on a nightly basis, and totals for the user's traffic calculated for the past day.

Calculation of totals for both web and FTP traffic passing through the proxy takes place before the daily log rotation occurs. The software that performs this is discussed in further detail in Section D.1).

In the calculation of the traffic totals, not all traffic is considered as billable. For this system it was decided by the researcher to not debit traffic from hosts on the local area network passing through the proxy server, the most likely cause of which would be as the result of mis-configured browser. As such, care needed to be taken to exclude not only traffic to the local domain (ru.ac.za), but also to other known local websites, as well as traffic resolving to the University's Class B netblock (146.231.0.0/16). Neither was 'hit' traffic (that traffic served out of either the local, or campus proxy caches[4]) billed, although the option exists to bill this at a lower rate. In essence the only traffic being billed against a user's quota is traffic which requires an external fetch by the University cache server, and thus directly impacting on the available bandwidth on the Internet access link.

After appropriate traffic has been excluded, totals for web and FTP traffic for each user are calculated, and stored in a database with a time stamp denoting the date they were inserted. A second process extracts from the database daily totals for each user over the previous 14 days and generates a user total for traffic over the period (using the sliding window method). These user total are compared against preset thresholds. Usernames with cumulative totals exceeding the thresholds set, are written out to appropriate files for the medium and high delay, and an access denied list, examples of which are shown in Section A.1.1.

The lists of usernames contained in the generated files are processed as access control lists (ACLs) by the Squid cache server. Based on these ACLs, a user's traffic is shaped by various delay pools, through the `delay_access` mechanism (see Section 4.3.3), or even denied, through the use of the `http_access` control mechanism, if the aggregated traffic exceeds a cutoff limit. The appropriate configuration options are illustrated in Figure 4.3 and a complete annotated configuration in Appendix A.1 . The configuration and operation of delay pools is discussed in Case Study 4 (Section 4.3.3).

[4]The Squid proxy classifies requests as one of a number of categories, the most common being HIT and MISS. Each of these can be one of number of subtypes, such as PARENT_HIT in the case where the parent was able to satisfy the request. [181]

Users are also informed via email that they have exceeded the lower threshold, to what degree, and the effect it is likely to have on their web traffic performance. A complete list of users with quotas over the lower cutoff threshold is also posted to the local `ru.stats` newsgroup each night. Included in this listing are users who have not yet exceeded the threshold, but are approaching it. This is to serve as a warning to users, and an example is illustrated in Figure 4.4. Users are also able to check on their quota status, and view a history of their daily usage via the personalised portal system on the RUCUS web page.

```
# log user idents for processed by the quota system
ident_lookup_access allow Rhodes
# Define ACLs using files generated by the Quota system
acl badperson ident '/usr/local/etc/squid/lists/banned'
acl lowuser ident '/usr/local/etc/squid/lists/lowusers'
acl hiuser ident '/usr/local/etc/squid/lists/highusers'
# Deny network abusers
http_access deny badperson
# Place users into delay pools
# Exclude local LAN hosts
# delay pool parameters are defined elsewhere.
delay_access 1 deny rhodes
delay_access 1 allow lowuser all
delay_access 2 deny rhodes
delay_access 2 allow hiuser all
```

Figure 4.3: Squid ACL extract for managing traffic quotas

Effectiveness

The implementation of quotas was found to be effective in substantially reducing the amount of external web traffic attributable to the RUCUS system, as users were no longer using the system as a base for indiscriminate downloads. Users who abuse the system are dealt with in a fair manner, without adversely affecting other users. The system, developed by the researcher has been functioning as a production system for eight months without problems. The majority of the web browsing is either text based (using the system web browsers such as `lynx(1)` and `w3m(1)`[5]) or file downloads. Taking this into consideration, various multimedia

[5]These are text mode web browsers that are available for several Unix platforms

This report lists RUCUS users in order of the amount of web traffic they have downloaded from RUCUS in the past 14 days. Users who download a large amount will have their web access slowed, or even denied. Users not explicitly listed in this report have downloaded less than 0.5 MB/day and fall into the no delay category. Totals are recalculated late every night, and delays are automatically enforced.

14-day total Daily average Delay

up to 28 MB up to 2 MB no delay
28 to 56 MB 2 to 4 MB slight delay
56 to 84 MB 4 to 6 MB significant delay
more than 84 MB more than 6 MB web access denied

Squid cache usage between 31/10/2000 and 14/11/2000:

User	Total (MB)	Average	Status
russell	165.42	11.82	web access denied
jockf	70.62	5.04	significant delay
ggvb	63.66	4.55	significant delay
madkev	61.84	4.42	significant delay
quark	51.03	3.65	slight delay
donovan	33.57	2.40	slight delay
nakiz	29.18	2.08	slight delay
shaun	24.77	1.77	no delay
cynic	20.49	1.46	no delay
drs	17.02	1.22	no delay
ghowell	12.47	0.89	no delay
richardp	9.98	0.71	no delay

Figure 4.4: Example of Squid notification

MIME[6] types such as video clips have also been restricted, to further conserve bandwidth.

This system is not only applicable to users. A conversion of the current system to work on IP addresses is a relatively minor change. The current Quota based system at Rhodes University operates on a two-tier level with per user (as obtained by the ident responses from clients) and a per IP quota. The software used to implement this quota system can be found on the accompanying CD-ROM further details can be found in Appendix E.

4.3.3 Case Study 4: Quota based shaping on Cisco Routers

With regards to the campus network at Rhodes University, a quota system already existed for traffic passing through the Squid caching web proxy server and has

[6]MIME is the standard for specifying content type of attachments or other data as described in RFC1512 and RFC1522.

been in successful operation for two years. However a quota system was still needed for traffic that does not pass through the web proxy, so-called non-web traffic, in order to limit the bandwidth that can be used by applications such as *Napster* [112][7], FTP or by accessing external web proxy servers. This case study describes the resulting system that was developed by the Rhodes University Information Technology division.

Design

Detailed accounting statistics are collected on a per IP basis from the gateway Internet access routers every two hours. These are then collated, and the traffic totals for each IP address averaged and compared against preset thresholds. Appropriate delay information is written to the Cisco router configuration files, which are then reloaded on the router, in order to activate the shaping. Separate quotas are maintained for incoming and outgoing traffic, as, due to the full-duplex nature of the Internet link, a current maximum of 512Kbit/second is available in either direction. Shaping of the traffic is performed by the mechanisms provided in the CISCO IOS. An example of the configuration of this is provided by Welcher [180]. Provision is also made for the exclusion of hosts, such as the core campus mail, web and proxy servers, from the shaping rules.

Implementation

This system requires an external machine to perform the data processing in addition to the router collecting the statistics. Details regarding the setup of the Cisco routers for accounting and retrieval of this information are provided in Section A.3.

One point to note is that this accounting data needs to be stored in the router RAM. Consideration should be given to the amount of RAM available when setting up

[7]The *Napster* application is not to be confused with the company of the same name which produces it. In order to destinguish between these, where the application name is used, it will be italicised. The same convention will be followed with other companies such as Imesh and CuteMX.

the accounting thresholds on the router. Figure 4.5a shows the RAM usage on the main University access router (a means of calculating the requirements is detailed in Section A.3). The RAM usage increases to an apex every two hours, when the accounting statistics are collected, and the accounting data is reset. This does however impose a penalty on the router when processing large accounting tables. Figure 4.5b shows the corresponding spikes in router CPU usage as data is collected. Figure 4.5c illustrates one of the potential problems associated with performing this monitoring - while the CPU is busy, it is unable to respond to SNMP queries[8] that were used to retrieve traffic information later used to generate the Figure (as indicated by the vertical white bars). The other interesting feature in Figure 4.5c is that this 'whiteout' only occurs during periods of high traffic, when the CPU usage rises above $\pm 60\%$, with no white bars appearing during the relative 'lull' in bandwidth utilisation between 03h00 and 06h00.

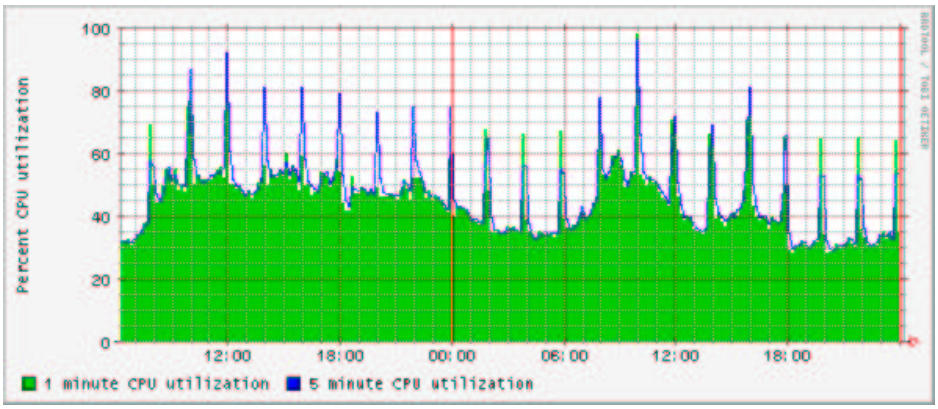
Effectiveness

Following the introduction of this system, abuse of network bandwidth by users running non-web based applications was reduced. The system allows for a user/system to download a large file but after a period of two hours the download may become throttled if it has exceeded a given threshold, as detailed in Table 4.2. For example if a user was to download a 100 megabyte file at an initial rate of 65Kbit/second, after the first hour, 29 megabytes will have been transferred. Assuming that there was no previous traffic and the quota run is scheduled at the end of the first hour, the hourly rate as averaged over the quota period is 4.8 megabytes. This causes the system to have its traffic limited to 64Kbit/sec. Following the accounting run at the end of the third hour, the average rate will be just over 14Mbytes/hour, and traffic will be limited to 32Kbit/second. The download will complete just before the end of the fourth hour. This is illustrated in Figure 4.6.

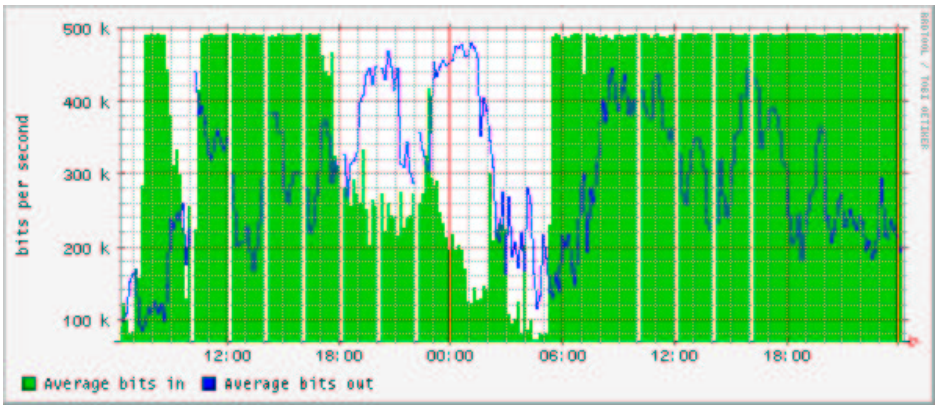
This progressive scaling of restrictions is particularly important in dealing with *Napster* [112] related abuse, as systems have their bandwidth limited or denied after the initial two hour 'free' period resulting from the quota processing interval.

[8]Under Cisco IOS, the SNMP agent also runs at a lower scheduling priority than the threads for handling traffic.

(a) Router Memory Usage



(b) Router CPU Usage



(c) Effect of increased CPU load on SNMP polling

Figure 4.5: Effect of traffic accounting on router RAM and CPU usage

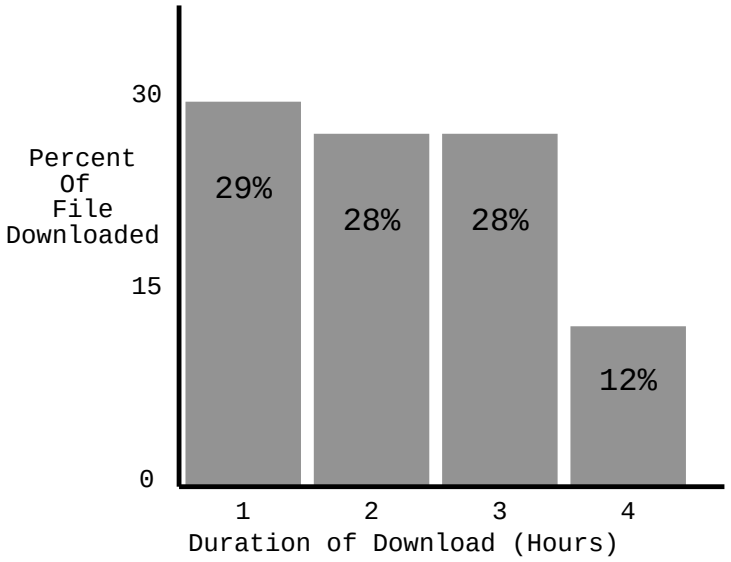


Figure 4.6: Effect of Traffic shaping on file transfers

The window periods over which statistics are aggregated are four and six hours respectively for outgoing traffic and incoming traffic. This timespan provides a heavier weighting towards incoming traffic, with a system's downloads having a more prolonged effect on the quota calculation. As an incentive to perform downloads during non-peak times, the quota system is deactivated from 01h00 to 05h00 daily.

Table 4.2: Non-web quota thresholds

Hourly Rate/hour	Daily Threshold	Limited Rate
>5MB	120MB	32Kbit/sec
3-5MB	72MB	64Kbit/sec
<3MB	<72MB	512Kbit/sec

Note: These are thresholds for incoming traffic and as such are based on a six hour sliding window period , as opposed to the four hours used for outgoing traffic. Hosts are subject their traffic shaped to the limited rate as determined by the hourly traffic rate over the window period.

4.4 Traffic Shaping

Traffic shaping allows for the manipulation and shaping of user traffic in accordance to a predetermined limit or rate. Traffic shaping is generally based on the idea of a finite queue or 'bucket' which is filled at a given rate by a pipe of a certain size, and emits traffic from a second (in most cases smaller pipe). This is known as the 'leaky bucket' concept, and is illustrated in Figure 4.7a. In most cases a bucket is used to imposes some form of delay on the traffic passing through it. This is analogous to the viscosity of the liquid passing through the bucket. Traffic is limited to a fixed maximum rate (as determined by the outflow), and provides no means of handling 'burst' flows.

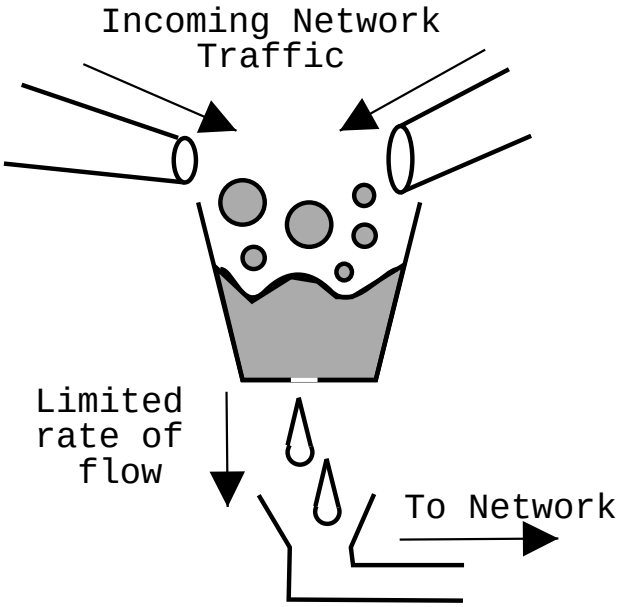
The 'token bucket' system works in a similar manner to the leaky bucket described above, but relies on a system of tokens for limiting the traffic flow, rather than a fixed size exit. Traffic is placed in a holding bucket. A second token bucket contains tokens which are deposited at a constant fixed rate, and stored up to a maximum level, after which new tokens are discarded. Traffic can however only be removed from the holding bucket if there are tokens available. For each unit of traffic, a corresponding number of tokens are removed. This system has the advantage over the leaky bucket system in that it allows for the 'bursty' nature of network traffic [131], yet still provides a means of limiting traffic as determined by the fixed token rate. Figure 4.7b illustrates the operation of a token bucket system similar to that implemented in the Squid proxy server's delay pool mechanism.

The token bucket system operates as follows:

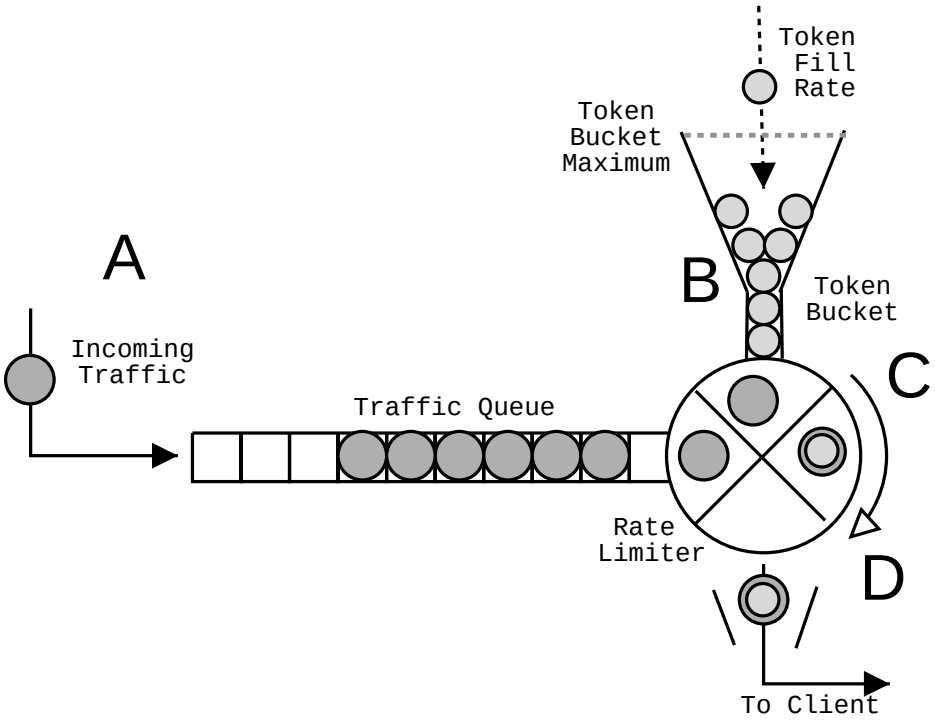
A Incoming traffic is received and broken up into units which are placed into a holding queue

B The token bucket is recharged at a given rate. Excess tokens are stored in the token bucket, up to a maximum level, after which they are discarded, this provides a cap on the maximum burst size than can be handled.

C The rate limiting mechanism operates by only removing a unit of traffic from the holding queue, when there is a token available to be removed from the



(a) Leaky Bucket



(b) Token Bucket

Figure 4.7: Leaky and Token bucket principals

token bucket in order to 'pay' for the traffic. Thus the traffic output rate is limited by the availability of tokens in the token bucket. In Figure 4.7b there are sufficient tokens to satisfy the needs of the traffic queue, and as such, the queue can be emptied.

D The traffic unit leaves the shaper and is transmitted onto the client system.

Traffic shaping can be performed by a number of different systems. These implementations of traffic shaping can be broken into three broad levels:

4.4.1 Application Level

The application being run and sending or receiving data is responsible for its own traffic shaping, usually in the form of rate limiting. This is achieved using a feedback loop to assess what the current transfer rate is, and then using techniques such as ACK limiting (the process of withholding ACK responses in order to slow traffic [127]) to adjust this rate to match the desired rate as closely as possible. The downside is that the application explicitly needs to support this option, as most applications rely on the standard congestion control mechanisms built into TCP/IP as discussed in Chapter 2, for rate limiting and flow control. User applications that usually implement features such as this are download agents such as *LFTP [97]* and *GetRight* [65]. Shaping can also be a part of server software such as the Apache webserver `mod_throttle` [160] which allows limiting of the rate of individual client connections being served. However in the case of a firewall using application level proxies, or other proxy servers such as the increasingly common caching web proxy, Squid [113] [175], an administrator can implement some form of centralised management. Application level shaping usually provides a much finer grained level of control than IP shaping mechanisms discussed below, since one is able to usually access information encapsulated within the protocol. An example of this is illustrated by the explanation of the shaping mechanism supported in the Squid caching web proxy (see Case Study 4 - Section 4.3.3)

4.4.2 IP Level

Traffic shaping is performed at a lower level than application based shaping, that of the IP transport layer, and is thus application independent. Shaping rules are usually based on a combination of the source and destination IP addresses, protocol and ports used by the hosts in the connection. Many offerings are currently available to perform this kind of traffic shaping, from the simple 'IP only' based systems such as those supported on Cisco routers [180] to more advanced offerings from companies like Packeteer [127] and Allot [9] which provide a form of hybrid system. These systems incorporate many of the advanced features of IP shaping, but with the added advantage of being able to supplement this lower level shaping by decoding many of the common application protocols such as FTP and HTTP. Packeteer has also published research into the management of TCP and UDP traffic at the IP level [123] [124]. These hybrid systems are discussed in Section 4.4.5.

An IP shaping mechanisms that has been investigated by the researcher is the *dummynet* mechanism that is provided by the FreeBSD kernel.

4.4.2.1 DummyNet

This is the traffic shaping system that is incorporated into the FreeBSD kernel and firewalling software, and is largely based on work originally done by Luigi Rizzo [152] [151] for traffic simulation. Due to its tight integration with the operating system kernel and the firewall system, the selection of traffic for shaping is very flexible. The system works by creating pipes with various configurations of pipe bandwidth, delay, probability of loss, and bucket/queue sizing. Packets are selected by a standard firewall rule and placed in a shaping pipe. Shaping is done using a bucket system, in combination with a delay factor. By limiting the rate at which packets are passed, the standard TCP congestion control mechanisms perform the actual limiting of the connection, thus requiring no modification of either client or server software. The standard TCP congestion control mechanisms as discussed in Chapter 2 are responsible for the actual limiting of transmission rates. This is particularly well-suited for a multiuser machine since the firewall

system allows one to differentiate traffic for individual users on the local machine, as this identification is done through a kernel interface rather than through an external service such as *ident* [12]. Shaping can be performed on any packets passing through the machine, whether the system is an originator, receiver, gateway, or transparent bridge[9]. A more detailed discussion on the effectiveness, and implementation of a DummyNet system can be found in Case Study 7 (Section 4.4.4).

4.4.3 Case Study 5: Implementation of traffic shaping in the Squid Caching Web Proxy

Squid Delay Pool Mechanics

Delay pools in the Squid caching proxy server are mechanisms to allow the cache administrator to specify limits on traffic passing though the cache.

The selection of traffic for specific delay pools is very flexible, as it makes use of the standard ACL mechanisms used for access control and other restrictions.

Examples of the use of delay pools include:

- as a part of a traffic quota system (as discussed in Case Study 3)
- allowing unrestricted access to certain sites but slowing down others
- limiting total HTTP traffic on a link, to prevent congestion and allow bandwidth for other services
- reducing the amount of 'casual surfing' during work hours
- limiting the bandwidth available to certain file types such as movie (asf,divx,mpeg) and MP3 audio files[10]

[9]Traffic shaping using a transparent bridge in versions of FreeBSD prior to version 4.2-STABLE of mid December 2000, will result in a kernel panic. This fault was submitted by the researcher, and a patch was committed by the code maintainer to the STABLE source tree in early December 2000.

[10]Squid currently (Version 2.3STABLE2) does not support a MIME type specific ACL. Version 2.4-DEVEL has some limited MIME based ACLs for incoming requests

Delay pools are defined in Squid as consisting of three classes:

class 1 All traffic passing though this delay pool is limited by a single bucket.

class 2 Traffic is governed by a **class 1** bucket, in addition to one of 255 individual buckets (**class 2**), determined by the final octet (bits 25-32) of the client's IP address

class 3 A more complex system consisting of an aggregate bucket, a network bucket determined by the class C portion, or third octet of the IP address, and a host bucket determined by the third and fourth octets of the address (bits 17-32).

For the classes consisting of multiple buckets, the delay imposed on a client is governed by the cumulative effect of all the buckets through which the traffic must pass.

Buckets are defined as having a restore rate, and a maximum capacity (denoted by the restore/maximum notation in bytes). This affects not only the size of the bucket (maximum size) but the rate at which information is replenished after being drained by a user request. There is also the option to configure the initial bucket level that is initialised. See the beginning of Section 4.4 and Figure 4.7b for a discussion on token bucket operation.

The restore rate and maximum size can be used configure delay pool buckets in order to create effects such as allowing for fast download of small objects, but severely limiting the speeds for larger objects. The level of the bucket can be viewed as credit which is utilised to allow the corresponding quantity of traffic to flow through to the client. By setting large buckets with low recharge rates, the administrator can provide for very quick downloading of small objects, up until the point where the bucket is exhausted. This is achieved by the stockpiling of tokens, during periods of no traffic from the client system being targeted by the delay pool – a common event as people usually pause between downloading a webpage and clicking through to another page in order to read the webpage contents.

Traffic to be passed through the delay pool mechanism is selected using the existing ACL lists used for normal access control by the Squid server. This allows for very flexible methods of traffic selection, ranging from matching based on a regular expression (eg. `\.asf$` to match all URLs ending in asf, the Microsoft video file format), to selection based on a specific user, as in Case Study 3.

An example of the slowdown and 'burst' ability described above is illustrated in Figure 4.8, where the slowdown in transfer speed is quite noticeable after the delay pool has drained, and the rest of the connection is being limited by the 4000 byte/second recharge. The squid delay pool mechanism is not as accurate as the limiting described in Case Study 7, and the oscillations around the target throughput for the average of the last twenty packets, is quite noticeable. The connection's average throughput does however exhibit a smooth decline towards the target throughput. Figure 4.9 shows a sample delay configuration as implemented for the quota system discussed in Case Study 4.1.

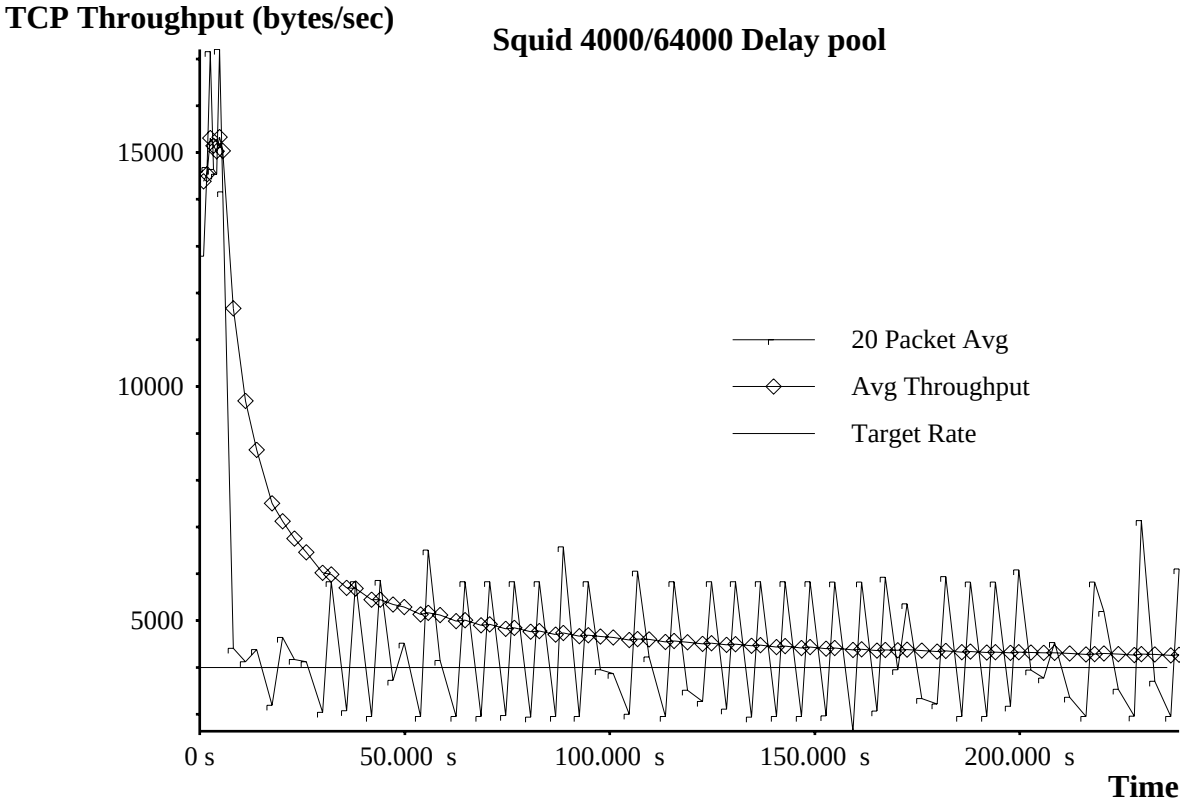


Figure 4.8: Slowdown effect of delay pools with burst rates on large files

```
delay_pools 2 # 2 delay pools
delay_class 1 1 # pool 1 class 1
delay_class 2 1 # pool 2 class 1
# place all traffic from the low users into the first delay pool
delay_access 1 allow low-user all delay_access 1 deny all
# place hi-user traffic into the second delay pool
delay_access 2 allow hi-user all delay_access 2 deny all
# limit traffic to 8KB/sec with 16KB/burst
delay_parameters 1 8000/16000
# limit traffic to a strict 2KB/sec
delay_parameters 2 2000/2000
#half fill buckets on initialisation
delay_initial_bucket_level 50
```

Figure 4.9: Sample delay pool configuration

Analysis of the traffic passing through the Squid delay mechanisms (as described in Appendix D) shows that the traffic throughput is rather bursty, most likely due to the rather course granularity of the timer used by Squid. This results in the traffic bursting at bit-rates much higher than that specified. The average throughput although higher than anticipated, does approach the desired target throughput rate. The client measured throughput rate is usually within 6.5% of the desired target rate.

Effectiveness

While not as accurate as the kernel based shaping mechanism discussed in Case Study 6 below, the delay pool feature in the Squid proxy server does have merit. The deviation from the specified rate (as shown in Figure 4.8), is not significant enough to discount this as a very effective method of bandwidth control. In addition, it allows for shaping of specific elements that are part of the HTTP application stream – which would not be possible using DummyNet.

4.4.4 Case Study 6: *Dummynet* as a bandwidth management tool

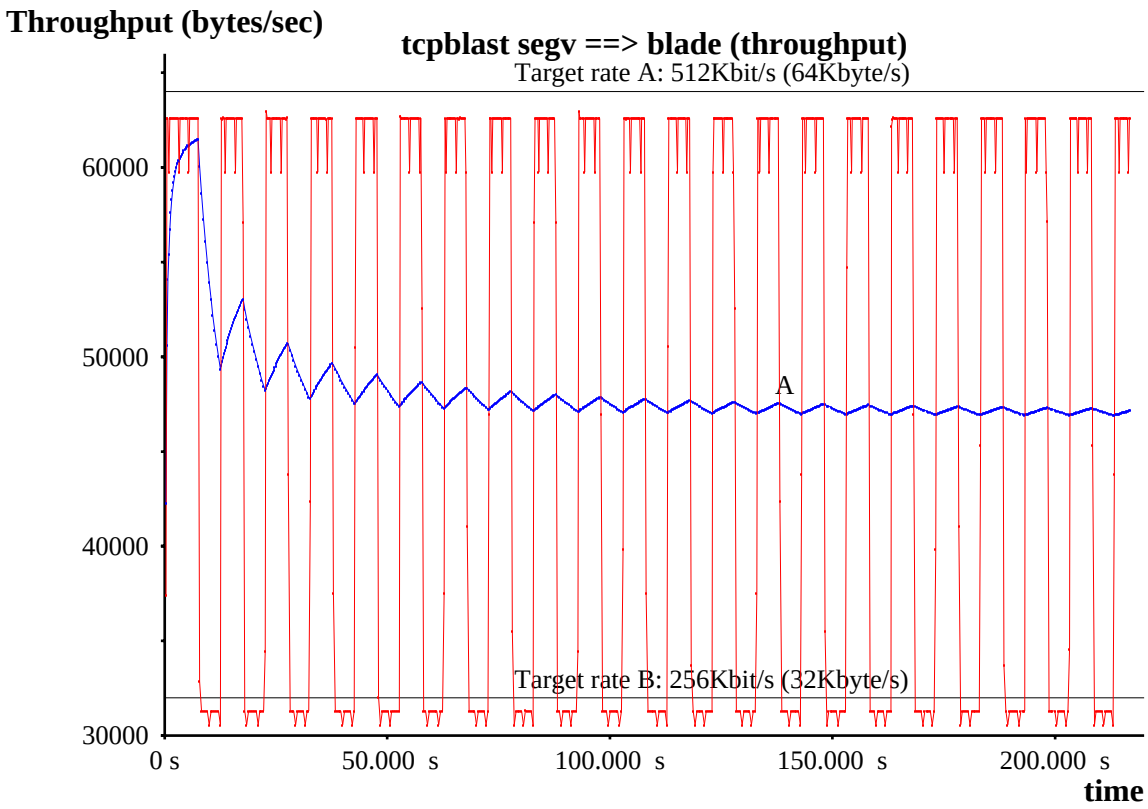
This Case study describes the Traffic management system that was implemented by the researcher on the RUCUS system.

FreeBSD's *ipfw(8)* [12] utility and *dummynet(4)* [56] are used to limit all non-privileged user traffic on RUCUS to a maximum of 20Kbyte/second each for inbound and outbound traffic not local to the Rhodes University network. All user traffic is aggregated in the dummynet pipes. This is to prevent users on the RUCUS system from taking an unfair portion of the University's Internet access bandwidth. Recent advances in the *ipfw* software allow for the limitation of a particular user to a specific bit-rate, within an overall shaped class. Class Based Queueing disciplines are a also supported in recent versions. Neither of these aspects were however investigated in this Case Study.

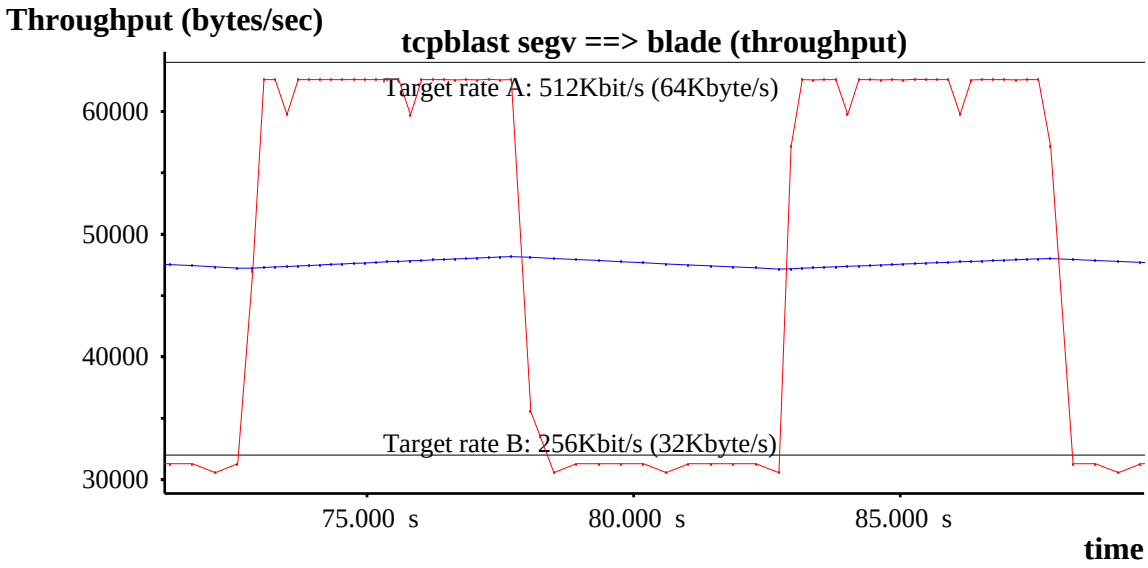
Dummynet was evaluated, and found to provide a fairly accurate means of shaping traffic. Traffic rates were shaped to within 4% of the specified rate, with rates always being less than the limiting rate. Figure 4.10{a,b} illustrate the speed and accuracy of shaping by oscillating the shaped rate between 512Kbit/sec and 256Kbit/sec, indicated by the target rate lines A and B respectively, is shown by the thinner line is the average throughput over the previous ten packets. This illustrates the rapidity with which the shaping comes into effect as is indicated by the square wave pattern produced. This rapid response is shown particularly well in Figure 4.10b. The line in the centre of the graph (as indicated by 'A') is the average throughput over the lifetime of the connection and is shown to exhibit an exponential decay over the period of measurement, with it asymptotically approaching the throughput bit-rate of 384Kbit/sec (48Kbytes/sec).

These provide shaping for user traffic, but not for performing any shaping on system users, or administrators. Services such as CVS (Concurrent Versions System), HTTP and FTP are handled by an exclude rule. Showing the approximately 10ms difference between traffic for a standard user, and a privileged user, in this case the root user. Figure 4.11 lists extracts from the configuration used on the RUCUS system details of the complete configuration can be seen in Section A.2.

The effect of the delay value indicated in this configuration can be seen in Table 4.3 (following testing by the researcher), where it is shown that the privileged user's response times are significantly faster, although not by the 10ms as expected. This discrepancy is most likely due to the fluctuations in traffic load on the Internet access link. The results from `segv` are substantially faster, and most



(a) Accuracy of DummyNet shaping



(b) Zoomed view showing transition

Figure 4.10: Example of the effect of *dummynet* shaping

likely due to the fact that the segv system has a minimal firewall ruleset and a 100Mbit/second up-link, as opposed to 10Mbit/second for RUCUS.

```
# pipe for users
# Incoming dummynet pipe
ipfw pipe 1 config bw 20KBytes/s delay 10 queue 128KBytes
# Outgoing dummynet pipe
ipfw pipe 2 config bw 20KBytes/s delay 10 queue 128KBytes
#Passing traffic though a pipe is achieved using the following
rules:
ipfw add 5000 pipe 1 ip from not 146.231.0.0/16 to
146.231.29.0/26 gid rucus ipfw add 5001 pipe 2 ip from
146.231.29.0/26 to any not 146.231.0.0/16 gid rucus
```

Figure 4.11: Selection and shaping of traffic with *ipfw*

Table 4.3: Effect of *dummynet* delays on user traffic.

User	RU LAN	ftp.is.co.za	ftp.cdrom.com
root	1.406ms	115.276ms	752.712ms
quark	1.407ms	121.413ms	754.707ms
segv	1.174ms	103.748ms	732.555ms

Note: User root is a privileged user and not subjected to shaping. quark is a normal user on the system. segv shows the measurements from another machine on the same network segment as RUCUS, where no shaping and a minimal firewall configuration is in place. Values measured are the average times for an ICMP echo (ping) for a 1000 packet test.

The University of Cape Town, South Africa, also makes use of *dummynet* to perform a bandwidth management and shaping on traffic for its approximately 15 000 users [103]. *Dummynet* is used to partition the total bandwidth into a number of different sized pipes.

4.4.5 Hybrid Systems

A number of systems which implement several advanced methods for traffic shaping are commercially available. Two of the leaders in the field are Packeteer's PacketShaper [128] system and the Allot NetEnforcer [10].

The basis of the Packeteer system is for smoothing and shaping of IP traffic, and the provision of guaranteed bandwidth for certain classes of traffic. Rather than merely placing packets in a queue, the Packeteer system makes use of several advanced features such as re-writing the TCP window advertisements to limit the transmission rate, using the standard TCP congestion control mechanism as described in Chapter 2. In addition it performs ACK clocking [123], again building on the TCP standard flow control mechanisms – where a machine will not send more than a given amount of data before an ACK for the outstanding data is received [78]. Packets are buffered by the device and released as appropriate in order to remain within the given bandwidth limit. The primary aim of the Packeteer system is to improve the efficiency of the network link, by reducing the congestion and maintaining the utilisation at a level (70%) where a balance is achieved between utilisation, and the avoidance of congestion [125]. The Allot system is based on similar principles.

For traffic shaping to be effective, placement on the network of the device performing the shaping is important, as it could have a detrimental effect on performance if implemented in the wrong place, such as between a caching proxy server and clients, rather than between the proxy and the external router. As a general rule of thumb, the device performing the shaping should be placed as close to the smaller network pipe as possible in terms of the network's logical layout. This is illustrated in Figure 4.12.

4.5 Traffic queues and Quality of Service

The traditional model for most networking devices is that packets are buffered and handled on a strictly first-in first-out sequence (FIFO queue). Packets that arrive once the queue is full are discarded (usually with the resultant issue of an ICMP Source Quench [23] [137]), or by simply dropping the packets, which causes the application to lower its transmission rate [24] [137]as the TCP/IP congestion avoidance algorithm (as described in Chapter 2) is activated due to the lost packets. Traffic Queues allow for the prioritisation of traffic based on certain criteria. Actual implementations of queueing also treat congestion in different

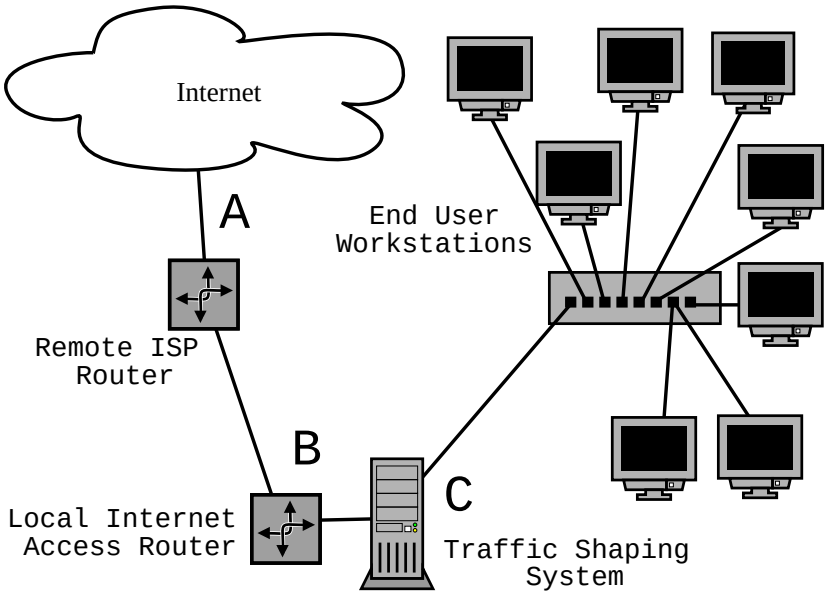


Figure 4.12: Placement of traffic shaping device on a network.

ways. The use of Queueing Disciplines is generally referred to as Class Based Queueing (CBQ). Simpler (early) implementations of queueing allowed for prioritisation of IP traffic based on the Type of Service (TOS) field in the IP Packet Headers [135] [163][11]. This field is often currently referred to as the Differentiated Service (DS) field in the IPv4 header [115]. Alternately, explicit selection of the traffic can be made by means of an ACL rule on the router. This model has been supported by Cisco routers since the mid 1990's, and has been available in the Linux stable kernel since version 2.2.0. FreeBSD has had integrated kernel support for class based queues since version 4.1 and shaping since version 2.2.8 [12].

CBQ at the simplest level consists of two components: filters and queues. Filters are used for selecting traffic and its placement in a queue. Once traffic is in a queue, the queue is responsible for decisions as to how to manage the traffic according to the appropriate queueing discipline selected.

It is important to note that these methods are really only applicable to traffic be-

[11] A similar mechanism is defined in IPv6 through the use of a Traffic Class field [42]

ing sent from a site, which in many cases constitutes a much lower use of bandwidth than the incoming traffic. Being able to perform management on the 'clickstream' (the series of requests generated by users clicking on links in web pages) for web traffic, and ACK packets for other types of connection allows for some level of traffic control. Incoming traffic can also be shaped, but if this is performed at a local router (such as that indicated at position B in Figure 4.12) it has already traversed the Internet access line, which is where the bottleneck is most likely to occur. Ideal positioning of equipment for shaping of incoming traffic would be on the far side of the access line as is shown by the placement of router as indicated by A in Figure 4.12. The problem with this, is that the network administrator usually only has access to the local router, with the remote router being fully under the ISP's control. Position C would be the ideal for the placement of a firewall or other shaping system such as those produced by Packeteer and Allot.

4.5.1 Quality of Service

Quality of Service (QoS) is used to refer to features (usually related to IP traffic) which allow a network manager to configure specifications for the delivery of services over a network along the lines of the integral QoS features in ATM networks. QoS provides the ability to divide and manage bandwidth to the extent that has previously only been possible using Frame Relay Virtual Circuits.

This is usually implemented as a series of queues on routers, since the two options for managing traffic are to either to drop or to delay packets, the latter being preferable provided the delay is not excessive. These queues are managed by one of a number of queueing disciplines which act on the traffic held in the queues, in order to effect a measure of management and implementation of QoS.

QoS is currently more frequently implemented on access and core routers on a network, but the IETF drafts for DiffServ [17] [62] look to make more widespread through the use of the first 3 bits of the TOS field in the IPv4 header. This implementation is discussed in Section 2.3, and in RFCs 2474, 2475 [115] [21]. RFC 2597 and 2598 [66] [91] expand on the use of the DS field. Details of the implementation of other QoS features into the IP protocol suite are discussed in Chapter

2.

Queueing disciplines commonly in use are described below. This list has been compiled from those supported by FreeBSD, Linux and Cisco IOS. Not all of these are available in all operating systems or network equipment.

4.5.1.1 First In First Out (FIFO)

This is the simplest queueing strategy, and is the baseline against which others aim to improve. In this model, packets are retransmitted in the order they are received, without regard to precedence.

4.5.1.2 Priority Queueing (PQ)

This is one of the oldest queueing techniques [178] [93]. A list of ACLs is used to determine traffic priority and place it in one of four appropriate queues (high/ medium/ normal/ low). The router processes packet queues from highest to lowest priority. This process is repeated for each packet to be transmitted, with lower priority queues only being serviced once higher priority queues are empty. Thus traffic in the low priority queue will only be sent when there is no other traffic pending. The problem with this is that when there is too much traffic in the high priority queue, the remaining three queues will never be serviced, resulting in packet loss due to timeouts, with the impact being increasingly significant the lower the priority. Care should thus be taken in the selection of traffic appropriate for each priority, both in terms of importance, as well as volume. This is generally suited, and intended, for low bandwidth links (ie. sub 512Kbit/sec) [37].

4.5.1.3 Fair Queueing (FQ)

A modification on the basic Priority Queueing algorithm [178], this method ensures that all queues are given some bandwidth in order to guard against bandwidth starvation, as is possible in the case of PQ. This does however exact a higher processing overhead on the system CPU [37].

4.5.1.4 Stochastic Fairness Queueing (SFQ)

This discipline was implemented in Linux kernels from version 2.2 onwards and designed for high bandwidth network connections [71], with low CPU overhead at the expense of fairness, and is a modification of the standard Fair Queueing algorithm. SFQ dynamically allocates a number of FIFO queues based on the number of flows (defined as packets with similar characteristics), with one flow per queue. These queues are serviced in a round-robin manner similar to that of Custom Queueing (see below) which allows for protection against a single traffic flow dominating the link bandwidth.

4.5.1.5 Token Bucket Filter (TBF)

This is also implemented in the Linux kernel. TBF is a simple queueing algorithm, which utilises a single queue and a bucket with a clocked token fill rate. Packets are removed from the queue at the rate of tokens are placed into the token bucket. The bucket can however accumulate tokens, which allows for rapid processing of short bursts of traffic (see Section 4.4 for discussion on the operation of token buckets).

4.5.1.6 Custom Queueing (CQ)

This method uses 17 queues for traffic, with queue 0 being reserved for system traffic, such as keep-alive packets for the interface [37] [178]. Traffic is selected in a similar manner to that used in PQ. Packets are sent from each queue on a round-robin basis. The rotation to the next queue is triggered by the traffic emitted from a queue exceeding a specified byte count. Since Packets may be larger than the remaining number of bytes until rotation, a queue can exceed its byte count, but most implementations penalise the queue by this amount on the next cycle. This indicates the importance of setting sensible MTU (Maximum Transmission Unit) sizes for traffic on the link (see Section 2.4). This method can be used to divide the bandwidth available to traffic on a relatively fair and accurate basis.

Bandwidth allocated to a queue, but that is not being used, is then shared between the remaining queues.

4.5.1.7 Weighted Fair Queueing (WFQ)

Weighted Fair Queueing provides an automated method of classifying traffic streams. Incoming packets from traffic streams are weighted based on the arrival time of the last bit, and are sorted in this weighted order for retransmission [178]. High bandwidth traffic is discriminated against in favour of lower bandwidth traffic. WFQ also makes use of the TOS field in the IPv4 header to aid in determining flow. Source/destination address pairs are also used for IP traffic. If high bandwidth connections are bursting, and the interface is exceeding a predefined congestion threshold, high bandwidth traffic is discarded. The queueing of traffic within the weighted flows is similar to that use by Fair Queueing. This is currently only supported on Cisco IOS 12.0 and later [37] and FreeBSD 4.2 upwards [56] [12].

4.5.1.8 RSVP

This is protocol developed for use on the Internet, which allows for applications to reserve bandwidth on a router and works in conjunction with WFQ and/or traffic shaping/congestion control. RSVP was originally defined in September 1997 as a proposed standard by RFC2205 [28] and RFC2208 through RFC2210 [98] [27] [189]. RFC2750 (2000) [68] updates the RSVP specification. Yavatkar *et al* have also put forward further enhancements in RFC2814 [191]. This is usually implemented on the outbound interface of a router connected to a WAN link. Applications transmit a 'PATH' message towards the receiver, containing a request for bandwidth to be reserved for this application. The receiver replies with a 'RESV' message. These messages create and maintain the path between the two hosts. This RSVP path is refreshed every 30 seconds. The timeout mechanism in the protocol allows for the prevention of resources being consumed by communications which no longer exist, as, if no PATH and RESV packets are received in a given timeframe, the reservation is deleted.

4.5.2 Congestion Avoidance

Congestion avoidance is the goal of a number of algorithms, which attempt to reduct the congestion that is experienced over a link. These algorithms all work, either by reacting to a detection of congestion on the link (due to packet loss or explicit notification), or by anticipating congestion by monitoring traffic levels. By minimising the congestion on a link, the overall performance is improved. Packet loss and delay resulting form congested links, often results in retransmissions of data by the sending host, which can further exacerbate the situation.

4.5.2.1 Random Early Detection (RED)

The Random Early Detection method of congestion avoidance is also known Random Early Discard. The principal behind this algorithm is to randomly start dropping packets from a router's buffers once a congestion threshold is reached [179] [37]. The rate at which packets are dropped is specified as a floating point value between 0 and 1, with the extremes meaning that in the event of congestion, no packets are dropped and that all packets are dropped respectively [12]. These packet drops result in the the underlying transport mechanisms between the two endpoints of the connection detecting packet loss and reacting accordingly, usually by throttling the speed of transmission. The net effect of this is to reduce the traffic waves which would otherwise be caused due to complete congestion of the link, and loss of multiple packets. TCP connections naturally assume this wave form due to the congestion control mechanisms inherent in the protocol (see Chapter 2), and as such a flow will oscillate between a base rate and a peak value [86]. After this peak rate has been reached, congestion will occur and the resultant TCP back-off will cause traffic to fall back to the median transmission rate. The use of RED allows for the slowing of a small number of flows, in order to prevent the congestion of all flows from occurring. The result of this algorithm is that better use is made of available bandwidth, as network link congestion is avoided, thus preventing a situation where all flows experience packet loss, and performing a simultaneous back-off, resulting in underutilised bandwidth [93] [71]. This is illustrated in Figure 4.13.

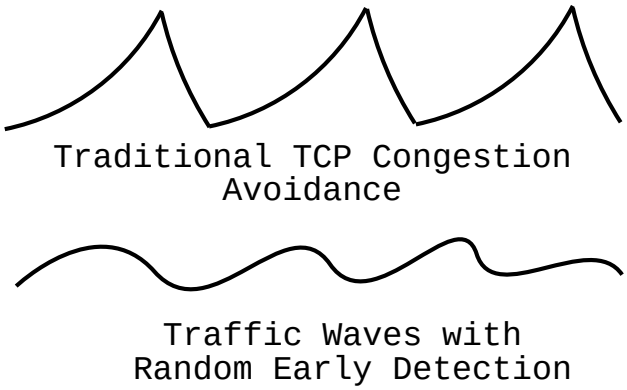


Figure 4.13: Traffic ‘waves’ and the effect of RED

4.5.2.2 Gentle Random Early Detection (GRED)

GRED is implemented in FreeBSD as a queueing discipline as a part of the dummynet(4) [56] traffic management system. The operation of this is a derivative of the RED algorithim, but it makes use of weighting in order to determine the traffic flows which will have packets discarded.

4.5.2.3 Weighted Random Early Detection (WRED)

WRED is an enhancement on the basic RED algorithm developed by Cisco Systems [37]. Using WRED, levels of precedence for IP traffic are defined, allowing for different packet types to be dropped at varying rates, such as allowing for higher drop rates in lower precedence traffic [93]. DWRED is a further enhancement, which allows for the distributed implementation of WRED across a number of Cisco Routers. This is usually implemented on core network routers.

4.5.3 Type Of Service and Differentiated Service bits

These fields are defined in the IP header, as a means by which an application can pass information to the IP transport layer as to the importance of the data contained in the IP packet. This information can be used to aid in making routing decisions where multiple routes are available, or for prioritising traffic in queues.

The development of these fields and the meaning of the values to which they can be set, are discussed in Chapter 2.

However very little actually sees these bits with the TOS level set to 0 by most applications. An analysis by the researcher of approximately 180 million packets showed that only a small proportion had the TOS/DS bits set to anything other than the default value of 0. A breakdown of the common TOS settings found is shown in Tables 2.2 and 2.3. Further details of this analysis can be seen in Appendix B.

Bearing the above mentioned congestion avoidance algorithms in mind, the buffer size on routers can also have a noticeable effect on network performance [71]. Large buffers allow for good throughput but, as links become congested, the latency of connections increases dramatically, often to the extent that packets that are held in queues have already expired, resulting in a protocol timeout and subsequent retransmission, which further aggravates the congestion situation. RED and other congestion control algorithms are particularly suited to alleviating this problem.

4.6 Active Content Control

Active control, limitation and manipulation of content provides a further means for bandwidth control. This is done at the application/protocol level by means of a proxy server. The protocol that this is most often implemented for is HTTP. The reasoning behind active content control originally stems from a desire to perform some kind of censorship or limitation of access of material, rather than to save bandwidth. For bandwidth management purposes this technique is most often used to provide some kind of limitation on ‘non-critical’ content during times of peak demand. This filtered content is most usually ‘adult-oriented’ or recreational sites. Another common reason for implementing content filtering is to remove banner advertising from web pages in order to save bandwidth, as is discussed in Case Study 7 below.

Often when this is implemented, it is construed by users to be an active form of

censorship, which it may be, and meets with resistance. This resistance can be reduced, by providing explanation as to the original motives behind the implementation. Many companies are using filtering of adult content in order to protect themselves, and decrease their legal liability in the case of being sued by employees or more likely, the recipients of email from employees, for the receipt of inappropriate material.

4.6.1 Case Study 7: 'Ad-banner' filtering and content filtering as a means of bandwidth saving.

This case study details the bandwidth savings that can be achieved by using active content filtering to block the ubiqitious banner advertisement on web pages.

The origins of banner advertising go back to the opening up and commercialisation of the Internet together with the development of the web browser that occurred in the early 1990's, prior to which it had largely been an academic and research orientated network, with a mainly textual content. Banner ads have been used to generate revenue by the display of the advertisement image on a client web page, which receives a portion of the revenue paid to the company which distributes the banner image. The use of banner advertisements has become increasingly popular in recent years with a revenue of US$2 124 million in the second quarter of 2000 [145], a 127.3% growth over the earnings for the same period in 1999.

The majority of banner advertisements can be considered a waste of bandwidth, which add little in the way of content to a web page. Research conducted by the writer assessing the make up of nearly 4000 banner advertisement images showed that many are animated and use more colours than required (4-32 would be sufficient for most cases), resulting in larger images in the region of 8-12Kbyte per banner with the largest at just under 40KB. If the web page with the banner ad is badly written (in that the HTML `<IMG>` tag lacks height and width specifiers), waiting for the banner image to be received from the advertising webserver can result in the entire page being delayed before display in the web browser, as the browser needs the download the banner image first in order to obtain its geometry

and thereafter be able to render the page for viewing.

Most banner advertisements are also non-cacheable since many are produced as output by CGI scripts instead of a link to a static image. Another option used is for the CGI script to redirect the browser to a static image (through the use of the HTTP 1.0 Status 302 - Redirect - method), but this suffers from the problem that a second HTTP request then needs to be made, with the associated TCP setup overhead. These CGI scripts are usually referenced by a unique URL, since these are often used for tracking particular sites, or even visitors. An example of an advertising service that does this is Doubleclick, which tailors the banner image served to the user viewing the site based on their past viewing and advertisement 'click' history. Output is also often marked with a `'pragma no-cache'` to prevent caching of the image by the browser or any intermediate proxy server. Most proxy servers also by default do not cache any output of CGI scripts (as identified by the .cgi extension or /cgi-bin/ in the URL path). Certain sites also use Javascript to refresh the advertising image after short time intervals. This is usually in order to generate more revenue for themselves at the expense of the viewer's bandwidth, due to the pricing model often implemented of a payment per impression. An example of this is the news page for a local newspaper group's web portal, which consumed nearly 15 Megabytes of bandwidth in a six hour period, since the image was refreshed every 30 seconds (a caching proxy server could possibly have reduced this to a single 30Kbyte request). While this may not seem an excessive amount, this was for a single workstation. If this was happening on multiple workstations, the detrimental effect on an organisations bandwidth would be quite noticeable.

Simply denying access to the advert banner image can cause 'disfigurement' of the resultant webpage, with the browser displaying its particular image for a referenced image it was unable to download, or be displayed (in this case trying to interpret the HTTP 404 Access Denied text response as image).

The solution to this is to replace the ad content with local content of the correct type (image/Java/Javascript), although this can be ignored to an extent due to the the nature of modern web browsers which identify content based on the MIME type header specified rather than file extension. Thus `image/*` can be replaced

by an image with a MIME type of `image/gif`. If this content is served from a local site, rather than being fetched from the ad-server source, the result will be a bandwidth saving and reduced latency in loading the web site for the end user. The choice of what local content to replace the banner with could be an image stating that an ad has been blocked (sized to the common 468x60 banner format as detailed in the Internet Advertising Bureau banner standards [83], and verified by the researcher). An alternative that has become more acceptable is to use a transparent 1x1 pixel image as a replacement, as this avoids the problem of disfiguring the page with a multitude of 'ad-blocked' images, as well as the problems that are experienced with resizing the block image to sizes other than the 468x60 pixels that was anticipated - the result being a badly disfigured blocky image.

The problem with using a transparent image is that certain page elements could get lost and remain completely unseen by the user. This is especially noticeable in the case of the ad-banner being contained within a `<LAYER>` tag, which is appended over the rendered page, as unless explicitly specified within the ad-banner layer, the page loses its title, being replaced by the title of "GIF Image 1x1 pixels" . The advantage is that in addition to being unobtrusive in comparison to an image stating that the ad has been blocked, the replacement image is correctly scaled by the browser to the same dimensions as the original image (assuming the page has a properly attributed `<IMG>` tag) and therefore the page retains the same layout as would have been displayed had the ad-banner been served. In the case where the `<IMG>` tag lacks `HEIGHT` and `WIDTH` attribution, having a 468x60 image will result in any columns that were less than 468 pixels wide to be widened. The use of a 1x1 image will result in the column layout being better preserved, as the column will then be as wide as the widest other element, or explicitly specified width.

By using a transparent 1x1 image, most users are unaware that ads are even being blocked. An important issue to note, is that the users can still click on the advertising image and follow the link, as it is only the image that is replaced, with the hyperlink remaining intact. In many cases the banner image has a border set which remains, leaving a coloured rectangle indicative of a link on the page.

Before any form of content filtering can be implemented, the administrator needs to be able to identify banner ads and distinguish a list of URLs which reference them. The majority of banner ads are served out by a relatively small number of large advertising companies. Many advertising banner serving websites can also be identified by host names such as `ads.company.com`, or served from a specific image directory on a webserver as in www.company.com/ads/banner1.gif. By analysing proxy server logs, and combining this with third party 'ad-banner' blocking lists available online, a list of advertising sites to be filtered can be compiled. One problem with this arises from situations such as that presented by the Akami network [3] which uses its distributed cache engines to serve both useful content and banner advertising images. As such, the entire site cannot be blocked, and a more specific regular expression will need to be developed in order to avoid filtering out the useful content (which is in the majority). The generated lists, can often be further reduced from the list or host names and URLs by using some form of regular expression (regex) to condense similar names into a single regex. The regex "`(http://ads.+|/banner?/|/ad?/)`" was determined by the researcher, and managed to match 53% of the total advertising URLs in the test sample, accounting for 17% of the traffic and 60% of the ad traffic.

Several block-lists are available on the Internet, as are a number of applications to aid in blocking advertising banners [182].

A similar approach can be taken for filtering out other content. The *SquidGuard* [14] filtering utility has several categories of 'undesirable' sites, including hacking, drug related and adult sites. This is not only limited to images. The same process can be extended to filtering content such as MP3s by replacing them with a text message, or a pre-recorded audio clip explaining that they are restricted. At Rhodes University the *SquidGuard* filtering software is used to provide some restriction of sites. However, only the images on adult related sites are replaced, leaving the text content intact. This provides a further safeguard in the case of there being a 'false-positive' match on a website, as the actual interpretable content remains for the user.

Problems do however occur, and sites can be incorrectly filtered. The redirection and filtering mechanism used should therefore allow for some type of exclusion

list. In addition it would prove more useful to users if an entire site is blocked, and that an explanatory page should be displayed instead of replacing the entire site with a 1x1 image.

This can be extended to offer even further bandwidth savings by applying redirection rules for other large files. Using this method, regular expressions can be used to automatically redirect the proxy server to a location for a file that is topologically closer than the site it may be requested from, in many cases a local mirror site. An example would be to redirect all requests for a new version of Netscape Communicator (currently approximately 20 megabytes) or Internet Explorer (55 megabytes download for version 5.5) to a local FTP or website containing the files, which would result in a substantial saving. This can be further extended to handle distributed sites such as the TUCOWS software repository [171], which has many globally distributed mirrors. Users will often end up downloading software from a site that is not the closest local mirror – an important consideration as there are often different charging mechanisms for overseas traffic. This can be transparently handled by redirecting the file downloads to the local server. Details regarding four URLs all of which reference the same file on different mirror sites is shown in Table 4.4 as measured from the researcher's server. However `tucows.is.co.za` is topologically closest to Rhodes University (which utilises Internet Solutions (is.co.za) as its primary ISP), followed by `tucows.saix.net`.

Table 4.4: Comparison of Ping times

URL	HOPS	PING Time (ms)
http://tucows.saix.net/files5/reget1.7.exe	10	193.700
http://tucows.is.co.za/files5/reget1.7.exe	7	52.333
http://www.tucows.de/files5/reget1.7.exe	17	707.500
http://www.tucows.com/files5/reget1.7.exe	21	642.376

By downloading from the closest mirror, the user achieves faster downloads, and a much lower chance of the connection being aborted due to network errors. This

can be achieved through the use of regex pattern matching and URL re-writing. For any site matching the regular expression '`http://.*tucows\..+/(.+)`' the URL can be rewritten as `http://tucows.is.co.za/($1)`, where `($1)` represents the path portion of the URL that was matched. This would result in all requests for any TUCOWS mirror being redirected to the local site. While this can provide a clean and efficient means of decreasing response times and bandwidth use, care needs to be taken that a number of criteria are satisfied:

- the chosen mirror is up to date and functional
- that there is a means for accessing the original file (this can often be achieved by appending a '?' at the end of the URL, but re-writers need to handle this appropriately.)
- users need to be informed about the procedure, and encouraged to report errors. User education about the benefits of using a local mirror is supplemented by such a strategy.
- highly popular files can be maintained on a server on the organisation LAN, but this can lead to an increased complexity in the re-write rules.

This was implemented by the writer using the Squid [113] proxy server which supports the use of a redirector. This is a helper application which is passed the URL requested by the client web browser from the proxy, and either returns a new URL, which is retrieved by the proxy instead, or a blank line, in which case the original URL is fetched. This added processing for requests can add a noticeable overhead to the total time required for the URL request, as well as an increased load on the proxy server. The increase in time taken to handle the web request from the client is in most cases offset by the performance increase gained by using a local or 'near' copy, and added advantage of which, is that the file will be held in the cache, and all subsequent referenced will be made to the same URL, thus allowing the requested content to be served directly out of the proxy cache. These impacts can in general be minimised by keeping the redirect and filtering lists as short as possible. Different redirectors also perform differently.

The selection of which URL's to redirect is determined by a configuration file. The redirector that was used for this implementation was *SquidGuard* [14] which supports filtering based on lists of sites as well as regular expressions.

Bandwidth savings

Direct bandwidth savings are difficult to quantify before implementation. Reasonable estimates can be made by looking at past access logs, and determining the amount of ad traffic. In everyday terms, there is a saving. In the case of the situation mentioned earlier in this section, banner filtering would have saved the 15MB that was downloaded in the six hour period. One variable effecting the savings, is the type of web content that users browse. Another is the accuracy of the filtering rules used for identifying banner ads, and banner ad servers. Extrapolating from this example, assuming it is a popular news site that is viewed by 50 people a day on the Rhodes University campus, if each user loaded the site up in the morning and left the browser open, an ad banner filtering system could potentially save nearly a gigabyte of traffic over an eight hour period.

4.7 Limitation of Services

Certain applications and network services such as FTP, HTTP and NNTP are traditionally known to be high bandwidth consumers. This was borne out in the traffic analysis detailed in Appendix A. In addition, peer-to-peer file sharing utilities such as *Napster* [112], *Imesh* [75], and *CuteMX* [59][12] are having an increased impact on network bandwidth. Other protocols, particular to one's site, may be identified through the monitoring processes discussed in Chapter 3. The approach to managing these network resource consumers can follow two of routes.

Filtering — Appropriate ports/IP addresses can be filtered at the firewall or external router level, either on a permanent basis, or with some form of time control with access allowed during non peak periods.

[12] These refer to the applications which are distributed by companies of the same names.

Shaping — Traffic matching rules are established and high bandwidth traffic can be constrained to within certain limits using the shaping mechanisms previously discussed. (see Section 4.4)

4.7.1 Implementing dynamic filtering as a means of traffic management

The need for traffic management can arise due to a number of factors, one of the foremost being that an organisation's Internet link is becoming saturated. Other reasons can be for security in which case the management comes close to traditional firewalling, or to prevent certain types of traffic for legal reasons. For example, in mid April 2000, the heavy metal band Metallica opened a lawsuit against Napster Inc. for the spread of pirated music [105] [111]. Included in this suit were the Universities of Southern California and Indiana, by virtue of the fact that they had not taken sufficient measures to prevent the use of *Napster* on their respective campuses [106] [104]. The 'required' blocking was difficult to implement due to the multiple servers used by the *Napster* client. The protocols discussed are known to be major consumers of bandwidth usage. The University of California, Berkeley cites a 61% use of the bandwidth by *Napster* traffic [166]. Vanderbilt University had similar figures with 45.26% of traffic being accredited to outbound *Napster* traffic and 8% incoming [172]. In South Africa, the University of Cape Town also experienced problems with *Napster* and *Imesh* saturating their Internet link [103]. These figures illustrate that the problem with bandwidth saturation is widespread even on high speed links [172] [166] [162].

Since mid 1999, a spate of applications have been developed to enable users to easily share a variety of file types over the Internet. Many of these build on the groundwork for sharing of MP3[13] media files pioneered by Napster [112], and the resultant technology implemented by a number of companies.

Soon after its release, Napster became immensely popular and, due to the sheer volume of users making use of the application, brought many institutions' Internet connections to a standstill. The nature of MP3 files are that they are large, with a

[13] MPEG 2 Layer III audio compression

typical file consuming between three and four megabytes of disk space, depending on the track length and the encoding options chosen when the file was initially compressed.

Following on from this popularity, other authors produced similar software, such as *Imesh* [75], *CuteMX* [59], *Scour Media Agent [34]*[14]and *Gnutella* [60] that were extended to potentially be able to share any file type. Of particular note is the development of *Gnutella* which utilises a protocol designed to be difficult to filter [60]. Many of these applications are also able to operate if one party in the peer-to-peer connection is behind a firewall [59] [60] [75].

These applications fall into three broad categories:

- those requiring a central server such as *CuteMX*, *Imesh*, and *Scour Media Agent*,
- the distributed server principle as implemented by Napster and OpenNap,
- the completely distributed model as implemented in *Gnutella*, where every node in the network can act as a server able to route messages on the network.

These architectures become increasingly difficult to block. The need for servers in the architectures is to enable some form of indexing and collating of files being offered by connected clients. Client search requests are processed on the server, and the results returned. This is intended to increase privacy and security for clients, as a user is unable to directly search for another user, or access their machine other than through the server which brokers the connection .

Napster uses a variety of ports and a number of servers which are tried when a connection is made. Thus, in order to block such traffic, firewall administrators would have to either block all outgoing connections, or spend time keeping up to date with server lists and block access to those. This task can be automated to an extent by parsing the server lists that are generated by services such as Napigator

[14]The Scour network was shutdown on 16 November 2000, after legal and financial trouble over copywrite infringements. [34] [159]

(`http://www.napigator.com/list.php`). This task could become quite time consuming, yet indiscriminately blocking all possible ports that could be used by such applications would soon result in legitimate services being disrupted. This is exacerbated by the fact that *Gnutella*, as a feature of its design, can use any port and allows any node on the network to act as an up-link server for new nodes joining the network. An alternative is to use an advanced means of filtering this traffic such as that provided by Packeteer [127] [128] and discussed in their publication *On Managing Controversy and Networks* [126].

These applications provide the ability to function if one of the parties is behind a firewall. In this event, in order to perform a file transfer, the party behind the firewall receives notification from the server that it is to connect to a port that is open on other users host. This negotiation is brokered and co-ordinated by the central server. Once the TCP connection has been established, data can be transmitted in either direction. This is somewhat different from the traditional model of network client software being responsible for the establishment of a connection to a server. This model fails to operate in the case where both systems are behind a firewall. In many cases a 'firewall' can be regarded as any intervening system which prohibits incoming connections , whether a firewall in the true sense, or a simple NAT[15] gateway. Concerns have been raised on the effect that the increased use of NAT gateways by users will have on these current file sharing technologies. The operation of NAT is discussed further in Section 4.11.

4.7.2 Case Study 8: A dynamic firewalling mechanism

A tool is needed that can identify and filter protocols which have been explicitly designed to be difficult to filter, such as *Napster* and *Gnutella*. This can be used for monitoring these 'unblockable' protocols, in addition to monitoring for SMTP, NNTP, FTP and HTTP traffic on non-standard ports.

Bandwidth consumed by utilities such as *Napster* and *Gnutella* are not the only cause for concern, monitoring of other network traffic of dubious intent may also be of use. Many 'warez' servers (servers containing illegal copies of software for

[15]Network Address Translation

download) make use of Internet Standard protocols such as FTP and HTTP, but run these services on non-standard ports[16]. These are usually undesirable either because of the content contained, or the lack of accounting information ('audit trail') for this traffic. This lack of audit logs is the common case where a proxy server has been deployed, and non-standard port numbers are used to avoid the proxy restrictions. Similarly there can arise a need for monitoring connections to an external proxy server which would allow users to bypass any local proxy restrictions. The same is applicable to external SMTP, POP3, IMAP and News servers. Monitoring of this sort could be regarded as a security measure – especially if the organisation has intellectual property which it wishes to prevent being disseminated via e-mail or other electronic means.

Traditional content filtering tools are unable to achieve this. Filtering of these cases is possible, but requires considerable manual effort. It is always preferable to have an automated system, with the ability to run with little intervention.

Implementation

The researcher has implemented a system using using *ngrep* [150] to gather packets off the network and apply preliminary filtering by means of packet capture logic (as implemented in the Lawrence Berkeley National Laboratory pcap library, and described in the *tcpdump(1)* man page on Unix systems [90]) and regular expressions. This output is then further processed by an encapsulating Perl script which performs logging, and generates appropriate ipfw [12] firewall rule sets for each matching class of traffic. An example of a script implementing this can be found at `http://rucus.ru.ac.za/~bvi/utils/sting/` and on the accompanying CD-ROM. This is a preliminary proof of concept, but has been tested and found to perform satisfactorily at relatively high traffic loads. A more detailed discussion of the system developed is also presented in Section D.2.

Appropriate regular expressions for protocols were generated by the writer, by examining packet dumps of a captured connection, and scanning for consistent data that can be encapsulated in a regular expression. These regular expressions are

[16]When determining what needs filtering, care needs to be taken to correctly handle the cases where legitimate traffic may run on non-standard ports. For example many legitimate webservers operate on port 8080 for historical reasons.

then used by *ngrep* for filtering general network traffic from the traffic under investigation. A useful tool for performing this examination is Ethereal [39], which provides a very useful feature of being able to produce an ASCII trace of the data transferred over the duration of a monitored TCP connection, with colour coding to differentiate the data attributable to the sender and receiver. Examples of regular expressions for matching common protocols as determined by the writer are illustrated in Table 4.5.

Table 4.5: Regular expressions matching common protocols

Protocol	Regular Expression
FTP	(^PORT .+,.+,.+,.+,) ^PASV
HTTP	(^GET .+HTTP/1) (^POST .+ HTTP/1)
Napster	FILENAME CONTAINS \".+\" MAX_RESULTS
Imesh	GET http://www.imesh.com/
Gnutella	^GNUTELLA CONNECT/

The Sting script as developed by the researcher was found to perform well under the test load, utilising less than 2% of the CPU on the test machine (Pentium III 500MHz, running FreeBSD 4.1), with the test link being run at 10Mbit/second. This should scale to operating efficiently on a lower powered machine, considering that the majority of Internet links in South Africa are in the sub 1Mbit/second range.

Feedback and monitoring

The system developed by the researcher is designed to be both flexible and extendable. Traffic does not necessarily have to be denied but can be logged for statistical or informational purposes. This data can be used for higher level decision making and capacity planning, by providing some kind of quantification of what contributes to the often significant but enigmatic ‘other’ or ‘ghost’ traffic on a network. By analysing the matches reported by the system, more permanent filters can be developed. An example of this is the collection of data about the various IP network blocks on which *Napster* servers are located. These address blocks can be filtered at router or firewall level rather than having rules dynam-

ically generated, yet the dynamic generation will still catch services running on other IP addresses and ports, easing the task of the administrator.

'Port hopping' applications such as *Gnutella* are also dealt with effectively. While the protocol is able to run on a myriad of ports in order to make it difficult to block, the actual exchange between clients remains the same and can be detected, thus effectively negating the multiple port functionality.

Caveats and further extensions

Sophisticated encryption tools which allow the tunnelling and encapsulation of IP traffic within an encrypted stream (effectively creating a VPN – Virtual Private Network) are freely available today on a variety of operating systems. Use of such encryption would negate the ability of a dynamic firewalling tool to detect and act upon nefarious traffic. Being able to encrypt a stream would still, in the majority of cases, require access to an external host on which the traffic is decrypted and forwarded to its destination. End-to-end encryption of the client-server or peer-to-peer link could be more problematic.

A firewall system such as that described above could potentially have a denial of service performed against it by a user forging malicious matching packets. Care thus needs to be taken both to limit the range of firewall rules that can be added, so as not to overflow the total number of allowable rulesets, and to ensure that rules should be expired after a given period of time since the last match.

The system as currently implemented only works effectively on protocols which have some form of repetitive identifier in plain text. Binary protocols (such as RealAudio) could theoretically be blocked using similar methods, but binary signatures would have to be found and an alternate capture program developed [155]. A possibility for this would be to use an identification system similar to that used by the *file(1)* command on Unix system, identifying file types by means of a *magic(5)* number. In summary, for the system to be successful, the administrator needs to understand the following points:

- requires access to network between clients and external router

- ideally suited to some form of firewall or transparent bridge
- protocols are matched based on regular expressions based within the protocol.
- binary protocols such as RealAudio and Microsoft Media player are more difficult but can be identified (Newer versions of *ngrep* support binary matches)

4.8 Threshold management

Threshold management is the process of reacting to events that have occurred on a network. These events are typically when certain thresholds have been crossed, most commonly in link utilisation. The general principle, is to perform some kind of corrective action once a threshold is crossed, and remains crossed for a given period. Once the levels drop below the threshold, the remedial action is stopped. This method is particularly suited to the development of automated systems, which rely on scripting to perform a first line of management on the network.

By using SNMP RMON traps and other methods [176] to detect when thresholds are reached, a system can be constructed which allows a network to dynamically respond to increasing congestion in an attempt to stabilise network performance. This can be achieved by bringing traffic shaping rules into force, or performing limitation on certain services, until such time as traffic levels drop below a given threshold. The configuration and operation of such a system on Cisco routers is described by Welcher [177].

Monitoring the number of times, duration and frequency that a link exceeds a threshold can be used to give an indication of the lines usage patterns. These can be useful for gaining an indication of when an upgrade in link capacity, or review of existing bandwidth management policy might be appropriate.

Threshold management, while more involved to design and implement, does have the benefit of allowing for a fairly relaxed bandwidth management policy, as the default is to allow relatively unlimited and unrestricted access, which is reduced

once the threshold limits are crossed. The downside of this is that users can become agitated when limitation policies come into effect. The action taken when thresholds are crossed also has to be carefully considered, as it is quite possible that the cause of the high bandwidth usage may not actually be affected by the threshold coming into effect. For example, if the first action applied once a threshold has been crossed is to start limiting web access through ad banner replacement and restriction of recreational sites, this would have no effect if the cause of the problem was a FTP transfer.

Another use for threshold management is for the activation of bandwidth on demand systems. Such systems use dial-on-demand to dynamically allocate extra bandwidth once certain thresholds on a primary line have been exceeded. The usual case is to dial up extra ISDN channels, thus providing additional bandwidth in 64KB/second blocks. Routing can also be adjusted 'on the fly' to provide for only certain traffic, such as EFT transactions, to be routed over the newly created link(s). Dynamically allocated lines are then disconnected once the threshold drops. This is an effective way of providing for infrequent spikes in traffic. This technique is supported in many of the modern router offerings from companies such as Cisco and Nortel.

4.9 Caching Proxy Servers

Caching proxy servers can serve a number of major benefits for bandwidth management. By forcing users to make use of proxy servers, or by promoting their use via an incentive scheme of sorts, an administrator is able to implement management, monitoring and control of traffic passing through the proxy servers from a central point.

Caching proxy servers are based on the premise that given a large user base, different users will request the same content. Thus if the content is stored locally, it can be returned directly form the cache, rather than having to retrieve the object from its source, resulting in bandwidth savings. Crovella and Bestavros explore the possible causes for this self-similarity of web traffic [41]. The researcher has

found that with small numbers of users, a caching proxy can make a noticeable difference.

Caching proxy servers are a logical extension to the application level proxy servers that have traditionally been part of a firewall setup, and in many cases can be integrated with an existing firewall application level proxy. In the same way that the application level proxy performs data retrieval on behalf of a client, and then passes the data onto the client, the caching proxy replicates this, but saves the data requested to disk. As such, any requests made by a client are first checked against the list of objects stored on disk, and if a match is found, the request is satisfied from the local on-disk cache rather than by fetching the object from the source, resulting in a faster response to the user request, as well as a saving in bandwidth.

Caching proxies are currently available that cache a number of protocols, two examples are:

Squid provides high performance caching for HTTP and FTP traffic. FTP traffic is only supported when it is originated by web browsers or other supporting FTP clients, which translate the users FTP request into an HTTP request that can be handled by the Squid proxy – also known as FTP over HTTP. HTTPS is also supported, but documents retrieved using this method, are passed directly to the client, and not cached on disk.

NNTPCache provides a means of caching USENET NNTP news feeds for a site, dramatically reducing bandwidth, since only articles that are read are fetched from a parent server. This can result in substantial bandwidth savings, since a full, or even partial, news feed can consume several gigabytes of network bandwidth on a daily basis.

The performance of a cache can be greatly improved through the use of some of the other techniques discussed, such as ad banner blocking, content filtering and quota systems.

Caching of HTTP content does raise a number of issues, such as the handling of dynamic content, and the removal of stale objects [175]. This issue is discussed by Gwertzman and Seltzer in their paper on web cache consistency [57]. These

issues are addressed in a number of ways. When a user requests an object from the Squid cache, a search is done of the cache database to confirm if the request can be satisfied from cached content. If the object is not held on the local cache, neighbour caches can optionally be queried [181] [183] [184], and the object retrieved. Failing this, the object must be fetched from source. The establishment of collaborative web caching networks can prove a substantial aid in reducing bandwidth use [168] [154] [190]. The effects of using the Squid proxy at Rhodes university are shown in Figure 4.14 which shows the number of client requests, and the number which are satisfied from the cache. These networks should however be carefully designed, so that the situation is avoided where it is faster to retrieve an object from source than from the peer cache in the network. Such a network has been successfully implemented by the UniNet academic network in South Africa, and on a larger scale by the Squid project [113]. Wolman *et al* have also carried out research on the performance and scaling of these co-operative cache networks [188]. With such a network in place, a number of advanced caching algorithms can be implemented, such as taking the cost of object retrieval from a number of sources into account [32].

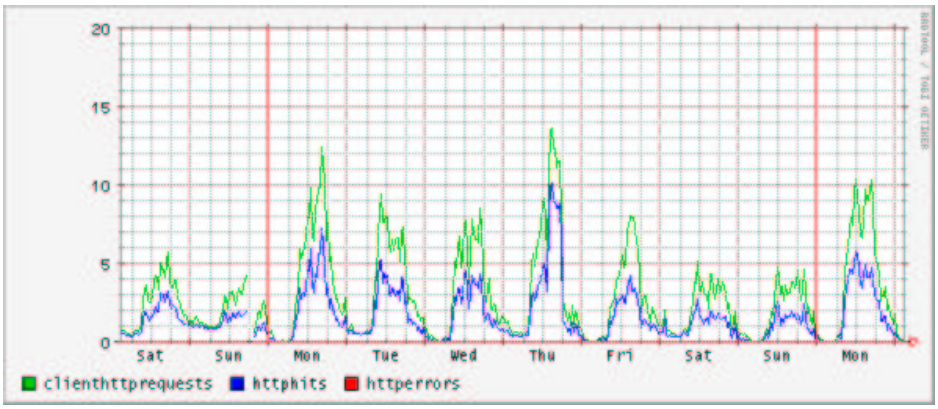


Figure 4.14: Client requests vs Cache Hits

When objects are returned by a web server, the HTTP/1.0 [16] and HTTP/1.1 [47] specifications state that a `Last-Modified` header be returned showing the last modification time of the object. The object is retrieved and stored on disk, along with all relevant headers received. Figure 4.15 shows the headers contained for an object in a Squid cache store.

If an object is found to be stored in the cache, the server sends an `If-Modified-Since` (IMS) request to the webserver querying if the object has been modified since the date of modification of the object saved in the cache. If there has been a change, the updated object is retrieved, otherwise the stored content is returned. In the case of a cache 'HIT', the object is returned to the user with a much reduced latency [95]. This method still requires a HTTP connection to the webserver to be established, but the amount of data exchanged is minimal in comparison to that which would be required to fetch the object from source. Expired objects are periodically removed from the cache by using a number of algorithms [153] [185].

```
http://www.cnn.com/virtual/2000/style/tech.css
HTTP/1.0 200 OK
Server: Netscape-Enterprise/4.1
Date: Sat, 11 Nov 2000 19:29:23 GMT
Content-Type: text/css
Last-Modified: Tue, 19 Sep 2000 14:24:31 GMT
Content-Length: 268
Accept-Ranges: bytes
X-Cache: MISS from cache.ru.ac.za
Proxy-Connection: keep-alive
<content starts>
```

Figure 4.15: Example HTTP Headers

Cache sizing is also an important factor when implementing a caching proxy server. Sufficient disk space needs to be present to provide a store for data for a reasonable period of time. Table 4.6 shows the theoretical maximum throughputs for various speed lines. Taking the example of a 512Kbit/second line, with 60% of the traffic being HTTP, that would mean that there is a maximum 3.2 gigabytes of data being transferred each day. This would mean that a 9 gigabyte cache would be able to hold three days' worth of traffic. More details on cache sizing can be found on the Squid homepage [113].

Most cache systems require that user browsers be explicitly configured to use the cache server, either by manual reconfiguration or using auto-configuration methods such as the `proxy.pac` browser auto-configuration developed by Netscape Corporation [114]. This has been supported by Netscape browsers since version

Table 4.6: Theoretical maximum throughputs for various speed network links

Bit rate	MB/day	Equivalent data
14.4K	151	105 1.44MB Disks
33.6K	354	246 1.44MB Disks
56K	590	410 1.44MB Disks
64K	657	1 CD-ROM
128K	1 350	2 CD-ROMs
256K	2 700	4 CD-ROMs
512K	5 400	8 CD-ROMs
1Meg	10 800	16 CD-ROMs
2Meg	21 600	33 CD-ROMs

2.0 and by Microsoft Internet Explorer since version 4.0. An example `proxy.pac` file is included in Appendix A (Section A.1.3).

Transparent caching is the process whereby no configuration is required for client systems. Client HTTP requests are intercepted (usually traffic bound for TCP port 80 on external hosts) at the router and redirected to the appropriate port on the cache server, which then processes the request. While this may at first seem an advantage, it does suffer from the drawback of not being able to use any form of password based authentication on the proxy server, as this collides with any authentication that may be required by websites [181]. Another is the confusion which may arise among users when error responses are displayed which were not being anticipated. Where transparent proxies do play an important role is where there are large client bases, or client systems over which the administrator has no control, such as dialup accounts at an ISP. At least one major ISP in South Africa (SAIX) is making use of transparent proxying for its dialup accounts.

4.10 Compression

The use of compression can provide a noticeable boost in the performance of a network link. The major hinderence to implementing this, is that compression is a two-way process, and as such both ends of a connection must support it. The simplest form to implement is that at the link transport level, such as that provided

by modems, or PPP. This compression is transparent to higher level protocols and applications making use of the link.

Compression at the application level can become more involved. Many applications do however support compression, including common web browsers such as Netscape Navigator/Communicator, and Microsoft Internet Explorer. A way for an administrator to reduce outgoing HTTP traffic is to make use of the compressed content as specified in the HTTP specifications [16] [47]. This can be achieved by making use of the `mod_gzip` module [148] for the Apache web server [13], produced by Remote Communications. The developers of this module claim up to 90% compression of HTML code. An advantage in this case, is that content will only be compressed if the requesting client states it is able to receive compressed content. This allows for optimisation where possible, while still maintaining compatibility. This bandwidth saving does however come at the cost of increased CPU utilisation on the webserver, as the content is compressed in realtime. Unless a web server is heavily loaded, most modern server platforms should be able to perform adequately.

This would be an ideal technology to integrate into a proxy server such as Squid. Up to the present this has not been implemented. There is however a means of obtaining compression between a proxy server and a client. The following scenario relies on the organisation having proxy servers at both ends of the network link. A compressed (and optionally encrypted) tunnel can be established between the two proxy servers using a utility such as Secure Shell (SSH) [122], which is able to provide port forwarding over this link. The systems are configured so that the local web clients make a request to the proxy server. This request is then passed on over the compressed link to the remote proxy which retrieves the web page, and returns it to the local proxy and ultimately the web browser. The advantage of this is that the compressed content utilises less bandwidth. The researcher was able to achieve compression rates down to 23% of the initial size[17] (a reduction of nine megabytes of web traffic to just over two megabytes) while using less than 7% of the CPU on a Intel Pentium III 500MHz system over a LAN. This has also been successfully tested over WAN links.

[17] This was for plain text based HTML content only, no images were included in the test sample.

Although not strictly compression, the proper design of web content using technologies such as cascading style sheets (CSS) and reducing the size of images contained by a page, can produce a noticeable saving [116]. Another area where compression can be used to reduce bandwidth is by educating users to compress attachments to e-mail messages where appropriate. Attachments such as word processor documents or spreadsheets can undergo a significant reduction in size after being compressed. Utilities such as WinZip [187] can be used to do this.

4.11 Network Address Translation

The implementation of Network Address Translation (NAT) on a network can prove useful in gaining better control of end user network usage. NAT is usually implemented in a firewall situation, where internal hosts have their Internet Address mapped to another address on the firewall system. In many cases the addresses used are from the non-routable ranges as stated in RFC1918 [147]. This option has become more popular in recent years, due to the increasing shortage of available routable IP addresses, as well as due to the increased use of firewalls. Even where a firewall is not used, client machines are required to use a machine to perform the NAT translation for them in order to be able to access external hosts on the Internet. The advantage of this is that the centralised point, provides an ideal focus for performing monitoring, as well as traffic shaping. Users are also forced to make use of local proxies in order to be able to access the Internet. The use of NAT, particularly dynamic NAT, can also enhance the security of client machines by denying incoming requests. This prevents client systems setting up services (such as FTP or Webservers) or any of the file sharing utilities mentioned in Section 4.7.2, although these may still be able to operate due to their design, which may result in abuse and consequent excessive use of network bandwidth. NAT is supported by most routers (such as products from Cisco and Nortel), and operating systems (FreeBSD and Linux), as well as embedded devices [1]. The researcher has experienced notable success in reducing bandwidth abuse by implementing NAT on networks.

4.12 Satellite Connections

The use of satellite connections as a means to supplement terrestrial data lines has become increasingly common in recent years. Satellites are able to to provide high bandwidth in most cases at lower cost than via traditional land-line [51]. This technology has enabled many home users, and users in remote areas not serviced by high-speed digital lines to obtain bandwidth which before would have been impossible [52]. The downside of this technology is the high latency that is present. This latency can have a detrimental effect as was discussed in Chapter 2. Most satellite systems are also only half-duplex, only able to transmit data from the provider to the clients dish. In order to overcome this, a land-line is used to connect to the provider for the 'upstream' connection [79]. The two halves of the logical connection are of differing bandwidths, and this can be referred to as an asymmetric connection. In many cases the satellite services is only used to stream HTTP and FTP Data connections, with the lower latency land-line used for interactive communications. This service is currently offered by two providers in Southern Africa: Infosat [80] and Siyanda [158]. The researcher has experienced success in using a system as described above. Under high load during the 2000 National Science Festival, the congestion of the outgoing traffic on the land-line proved to be the limiting factor for downloads via satellite. Never the less, this technology can prove useful as a means of either obtaining lower cost bandwidth, as a backup link, or as a part of a 'bandwidth on demand' system.

4.13 User Education

> "Men's thoughts are much according to their inclination, their discourse and speeches according to their learning and infused opinions."
>
> - FRANCIS BACON, *(1561-1626) Of Custom and Education [15]*

User education is probably one of the most effective means of long term bandwidth management. By educating users as to the benefits to themselves of adhering to bandwidth management policies, one reduces the number who might try

and subvert the system, as well as acquiring feedback as to what user requirements are. Some of the items that should be included in user education are:

- The benefits of making use of organisational proxy servers, in terms of response times, how the proxy server works, benefits with regards to any quota systems that may be in effect.
- The use of 'network close' mirrors when obtaining software. Good examples of this are the TUCOWS [171] and Metalab/Ibiblio (previously Sunsite) [74] software repositories. Although a proxy server can be configured to automatically direct users to the preferred site, the administrator is unlikely to be able to set up auto direction for all sites. By familiarising users with the practice of using mirror sites, the redirection rules on the proxy server, tend to serve as a fallback when users forget, or new users access the system. Use of a local site allows for faster downloads, has a higher chance of completing successfully, as opposed to a file being fetched over an international link.
- Quota systems and how they affect users. What quota incentives (if any) are there for using the system during non-peak periods.
- The benefit of delaying large downloads until off-peak periods, and possible means of doing this, such as a centralised download server, which users can submit requests to. These are then stored, and executed in off-peak periods.
- What other policies are in place? Are ad banners blocked? What is the benefit and effect on users of these policies? It is a reasonable assumption that if users understand the policies most, although not all, will be less likely to attempt to circumvent them.

User education is an ongoing task, which needs to be periodically revisited, and an attempt made to document policies that are in place. Without any form of user education, bandwidth management becomes a difficult task. Even with an education policy in place many users will attempt to make maximal use of the resource, often with blind disregard to others on the network. It is for this type of user that the quota systems are particularly effective.

4.14 Summation

The general aim of a bandwidth management policy should be to manage 90% of the users and traffic with 10% effort. No system is completely watertight, and there will always be users who are unco-operative and who will find a means to circumvent controls, but this should be the exception rather than the rule.

Horwitt raises a number of interesting points in her article entitled *The Politics of Policies* [70]. These include the importance of going through the correct process when designing and implementing a network management policy. Users are less likely to resist policies if they feel they are 'fair' and are able to have input into the further development of the policy. Similarly, for a policy to be successful, it requires the backing of the organisational management. When preparing to implement a policy, one way of gaining user feedback without the problems arising out of asking users to define their own policy, is to present a broad outline to the network user base, for comment.

The degree to which there is flexibility for negotiation and debate with regards to the policy proposal, depends on the organisation. Implemented policies should not be completely inflexible, as situations often arise where exceptions may need to be made for short term projects, or situations. A network management policy should also be open to evolution and development as the network grows, and new technologies are implemented within the organisation. An effective method of gauging the success of a policy is to obtain feedback from the end users, whether positive, or negative, or even suggestions for improvements.

Chapter 5

Conclusion

The primary outcome of this research has been to provide a conceptual framework of methods which network administrators can implement when managing their network bandwidth. Although a number of appropriate tools have been suggested for implementing the methods discussed, these are not the only such tools available. The tools selected represent those that were both freely available in terms of source code, and for the researcher's operating system of choice *viz.* FreeBSD. The writer has also highlighted the complementary interaction between bandwidth management methods where appropriate and, in doing so, has achieved the primary aim of the research. This work should provide a useful resource for network administrators undertaking bandwidth management on their networks, and in understanding the various methods available, and the benefits of each.

There are also a number of secondary outcomes from the research:

- Several methods for the visualisation of historical and near realtime network traffic data have been described and discussed. New methods of visualisation were also developed as part of this research. Each of these have their own particular strengths and weaknesses as discussed in Section 3.3.2. It was established that the methods discussed should be used to complement each other, in order to be able to obtain the maximum amount of information from a dataset being analysed through the visualisation process.

- The work done on three-dimensional landscape visualisations has also been promising. This provides a novel format for the display and intuitive interpretation of large amounts of data relating to network traffic. The researcher has been unable to find any literature describing the use of such a method for the visualisation of network traffic. The only similar uses which were encountered were those for the generation of 3D bar graphs for the tracking of trends in stock markets [2]. This innovation, together with the other visualisation methods assessed (see Section 3.3.2), provides the network administrator with a sound basis for the analysis of network traffic data by visual means. The tools used by the writer for the automated generation of the 3D landscape models were however only able to produce output in a relatively rough form. In addition when viewing a landscape model, the user is unable to obtain detailed information about a point. These areas in which further research can be carried out, possibly making use of the OpenGL 3D API for the construction and manipulation of the image, rather than the complex rendering process employed in this research. Such an implementation could provide a much faster feedback for the manipulation and navigation of the 3D models by users. This visualisation method should be applicable to any data sets of three or four data series. The 'fourth dimension' can be indicated through the use of colour.

- Through the use of experimental scenarios and practical, 'real world' implementations of systems it has been established that it is possible for an administrator to effectively control network traffic, and thus manage its resultant impact on the total network bandwidth. One highly effective means of control is the use of techniques such as the caching of content by proxy servers, or local web and FTP mirror sites, which bring content as close to the end users as possible (both in terms of network topology and bandwidth capacity). A further, very successful means of reducing network bandwidth requirements has been demonstrated through the shaping of network traffic. Both of these approaches can reduce the bandwidth requirements of a LAN, and also improve the usability of the often highly congested WAN links for an organisation.

- Software has been developed by the researcher to aid in the management of network traffic. These utilities were developed either in the case where no existing implementations were found to be available or where existing tools were insufficiently flexible, or were difficult to modify. The two potentially most useful systems developed are the quota management system for the Squid proxy server (Section 4.3.2), and the dynamic firewalling utility (Section 4.6.1). These systems have been developed to the extent that they are now stable, and of a quality level that they can be released for general public use.

  - Although the quota management system has been in use on the RUCUS system for nearly eight months, it has a number of features which could be enhanced. These include simplifying the configuration and providing more detailed feedback to users as to what particular downloads affected their quota, as well as the current status of their quota. The addition of these features should not be an overly large task, but fell outside the scope of this research project.
  - The dynamic firewalling tool as currently implemented by the writer suffers from the drawback of not being able to perform pattern matching on non-textual data. This was due to a limitation in the version of the *ngrep* [150] utility on which it is based, which has recently been rectified in the latest version of *ngrep*. Despite this shortcoming, it can still provide an effective tool, since the majority of Internet protocols are text based, or contain recognisable amounts of text. The process of determining the 'signatures' for non-textual data is likely to be more involved than that for text based protocols, but once determined, the principles of operation should remain unchanged.

Further outcomes of this work are:

- The development of a RMON clone tool for Unix systems which does not support the full RMON MIB. This tool is not MIB compliant, and only emulates the counters of traffic attributable to each host on a local network in

respect of bytes passed in and out. A full implementation of the standard was beyond the scope of the project, and the writer chose to only implement the portions required to facilitate further testing. This could be further extended and integrated with one of the freely available SNMP packages such as Net-SNMP (previously ucd-snmp) [64] in order to provide a RMON compliant implementation for Unix systems, such as that commonly found on routers and other network equipment.

- A number of firewall management and accounting scripts have been developed, in particular the work on using firewall accounting rules to provide a breakdown of TCP traffic on a per protocol/port basis. This work has subsequently been extended by co-workers and upgraded to operate in conjunction with the *RRDTool* [119] package. This work also served as the basis for the development of the traffic monitor system described in Case Study 2, and the traffic accounting system used on the RUCUS system as developed by a colleague.

- The database system and front-end were developed for providing long term storage and visualisation for the dataset which was used in a number of the case studies. Work was also done on the storage and optimisation of large datasets.

Any study of this kind has its limitations in terms of scope and size. Not all issues which arose could be addressed in the context of this work, and hence the following recommendations for further research arising from the work are made:

- An in-depth evaluation of the traffic shaping and queueing strategies provided by Cisco routers. These routers comprise a significant portion of the routers on the Internet and this work would therefore be beneficial to the networking community at large. This research was not possible as the writer did not have access to suitable Cisco equipment to use for experimentation. Networking equipment from other vendors should also be considered.

- An evaluation of proprietary hybrid traffic shaping systems such as those offered by Allot [9] and Packeteer [127] should be done. Building on this,

an open source system could be implemented, with the aim of providing similar functionality. Further work in this field would add greatly to the understanding of the operation of these systems. These systems were not available for the researcher to investigate further.

- The further development of the 3D visualisation system to work on a wider range of data sets, as well as providing a more flexible form of display. One possible extension that has arisen from discussions with other researchers in the Department of Computer Science at Rhodes University, is the integration of this form of visualisation into the Virtual Reality system which has been developed at Rhodes.

- The integration of intelligent agents as described by Fourie [50] could be used to enhance a Virtual Reality visualisation of a large dataset, by providing detailed information for a specific point in the landscape.

In summary, this work has achieved the aims set out in Section 1.1. As well as providing a theoretical overview for the management of network bandwidth issues, it has evaluated certain practical applications of this field and has opened up several areas of further research in this field.

Glossary

The following is a list of terms and acronyms used in the body of this document.

ACK	TCP Packet with an Acknowledgement flag set
ARPANET	Advanced Research Projects Agency Network, the predecessor to the Internet
ASCII	American Standard Code for Information Interchange, commonly used to denote plain, human readable text data.
ATM	Asynchronous Transmission Media
BCP	Best Current Practice
BDP	Bandwidth Delay Product
BSD	Berkeley Software Distribution
CAR	Committed Access Rate
CBQ	Class Based Queueing
CE	Congestion Experienced
CGI	Common Gateway Interface
CIR	Committed Information Rate also known as CAR
CSS	Cascading Style Sheets

Diginet	Digital Data Line
DNS	Domain Name System
DS	Differentiated Service
ECN	Explicit Congestion Notification
ECT	Explicit Congestion enabled Transport
EFT	Electronic Funds Transfer
FIFO	First In First Out
FQ	Fair Queueing
FTP	File Transfer Protocol
GSM	Global System for Mobile Communications
HTML	HyperText Markup Language
HTTP	HyperText Transport Protocol
HTTPS	Secure HyperText Transport Protocol
IANA	Internet Assigned Numbers Authority
IAB	Internet Activities Board
IETF	Internet Engineering Task Force
ICMP	Internet Control Message Protocol
IOS	Cisco Internetwork Operating System
IMP	Internet Message Processor, an early form of router on the ARPANET
IMS	If Modified Since
IP	Internet Protocol

IP Address	The address for a computer connected to an IP network in the form of aaa.bbb.ccc.ddd.
ipfw	IP Firewall the FreeBSD kernel firewall control utility
IPX	Inter-Network Packet Exchange, a network protocol implemented by Novell NetWare
IRC	Internet Relay Chat
ISDN	Integrated Services Digital Network
ISP	Internet Service Provide
KB	Kilobyte
Kb	Kilobit
LAN	Local Area Network
Latency	The Round Trip time for data
LBL	Lawrence Berkeley Labs
LDAP	Lightweight Directory Access Protocol
LFN	Long Fat Network
LISA	USENIX Systems Administration Conference
LTN	Long Thin Network
MAC	Media Access Control address
MB	Megabyte
Mb	Megabit
Mbit	Megabit
MIB	Management information Base

MIME	Multipurpose Internet Mail Extensions
MP3	MPEG Audio Layer III
MRTG	Multi Router Traffic Grapher
MRU	Maximum Receive Unit
MSS	Maximum Segment Size
MTU	Maximum Transmission unit
NAT	Network Address Translation
NFS	Network file System
NGO	Non Governmental Organisation
NNTP	Network News Transport Protocol
NTP	Network Time Protocol
plr	Packet Loss Rate
PMTU	Path Maximum Transmission Unit
POP	Point of Presence
POP3	Post Office Protocol version 3
PPP	Point to Point Protocol
QoS	Quality of Service
RAM	Random Access Memory
RED	Random Early Detection
RFC	Request For Comment
RMON	Remote Monitoring
RTO	Retransmission Timeout Estimator

RTT	Round Trip Time
RUCUS	Rhodes University Computer Users Society
SFQ	Stochastic Fairness Queueing
SMB	Server Message Block
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Management Protocol
SQL	Structures Query Language
SSH	Secure Shell
STD	Internet Standards Document
TBF	Token Bucket Filter
TCP	Transmission Control Protocol
TOS	Type of Service
UART	Universal Asynchronous Receiver/Transmitter
UDP	User Datagram Protocol
URL	Universal Resource Location
USENIX	Advanced Computing Systems Association
VLAN	Virtual LAN
VPN	Virtual Private Network
VRML	Virtual Reality Markup Language
WAN	Wide Area Network
WFQ	Weighted Fair Queueing
WRED	Weighted Random Early Detection

Bibliography

[1] ACCTON TECHNOLOGY CORPORATION. Cheetah iSharer3010 Internet Access Sharing Device. Online: http://www.accton.com.sg/accton/products/hubs/is3010/features.html.

[2] ADVANCED FINANCIAL NETWORK. 2c2c - the three dimensional stock market visualisation tool. Online: http://www.2c2c.com, 2000.

[3] AKAMAI TECHNOLOGIES INC. Company homepage. Online: http://www.akamai.com/, 2000.

[4] ALLEN, J. Cricket homepage. Online: http://cricket.sourceforge.net, 2000.

[5] ALLMAN, M., FLOYD, S., AND PARTRIDGE, C. RFC2414: Increasing TCP's Initial Window. Internet RFC, Sept. 1998. Experimental.

[6] ALLMAN, M., GLOVER, D., AND SANCHEZ, L. RFC2448bcp28: Enhancing TCP Over Satellite Channels using Standard Mechanisims. Internet RFC, Jan. 1999. Best Current Practice.

[7] ALLMAN, M., PAXSON, V., AND STEVENS, W. RFC2581: TCP Congestion Control. Internet RFC, Apr. 1999. Standards Track.

[8] ALLMAN (ED), M., DAWKINS, S., GLOVER, D., GRINER, J., TRAN, D., HENDERSON, T., HEIDEMAN, J., KRUSE, H., OSTERMANN, S., SCOTT, K., AND SEMKE, J. RFC2760: Ongoing TCP Research Related to Satellites. Internet RFC, Feb. 2000. Informational.

[9] ALLOT INC. Allot hompage. Online: http://www.allot.com, 2000.

[10] ALLOT INC. NetEnforcer. Online: http://www.allot.com/html/ products_netenforcer.shtm, 2000.

[11] ALMQUIST, P. RFC1349: Type of Service in the Internet Protocol Suite. Internet RFC, July 1992.

[12] ANTSILEVICH, U., KAMP, P., NASH, A., COBBS, A., AND RIZZO, L. *ipfw(8) IP firewall and traffic shaper*, Feb. 2000. included in source distribution.

[13] APACHE SOFTWARE FOUNDATION. Apache web server. Online: http://www.apache.org, 2000.

[14] BALTZERSEN, P., AND HÅLAND, L. SquidGuard homepage. Online: http://www.squidguard.org, 20000.

[15] BARTLETT, J., Ed. *Familar Quotations: a collection of passages, phrases and proverbs traced to their sourced in ancient and modern literature.*, 10th ed. Little Brown and Company, Boston, Online: http://www.bartleby.com/100/ [Out of Print], 1919.

[16] BERNERSLEE, T., FIELDING, R., AND FRYSTYK, H. RFC1945: Hypertext Transport Protocol HTTP/1.0. Internet RFC, May 1996. Informational.

[17] BERNET, Y., BLAKE, S., GROSSMAN, D., AND SMITH, A. draft-ietf-diffserv-model-04.txt: An Informal Model for Diffserv Routers. Tech. rep., July 2000. Internet Draft - Work in progress (expires January 2001).

[18] BEYSSAC, P. Bing. Online: ftp://ftp.lip6.fr/pub/networking/bing-1.0.4.tar.gz.

[19] BIERMAN, A., BUCCI, C., AND IDDON, R. RFC2895: Remote Network Monitoring MIB Protocol Identifier Reference. Internet RFC, Aug. 2000. Standards Track.

[20] BLAAUW, P. Security Administrator, Pick 'n Pay IT. Pers Comms, Dec. 2000.

[21] BLAKE, S., BLACK, D., CARLSON, M., DAVIES, E., WANG, Z., AND WEISS, W. RFC2475: An Architecture for Differentiated Services. Internet RFC, Dec. 1998. Informational.

[22] BRADEN, R. RFC955: Towards a Transport Service for Transaction Processing Applications. Internet RFC, UCLA-OAC, Sept. 1985.

[23] BRADEN, R. RFC1122: Requirements for Internet Hosts - Communication Layers. Internet RFC, IETF, Oct. 1989. Editor, Standard.

[24] BRADEN, R. RFC1123: Requirements for Internet Hosts - Application and Support. Internet RFC, IETF, Oct. 1989. Editor, Standard.

[25] BRADEN, R. RFC1379: Extending TCP for Transactions Concepts. Internet RFC, ISI, Nov. 1992.

[26] BRADEN, R. RFC1644: TTCP TCP Extensions for Transactions Functional Specification. Internet RFC, ISI, July 1994. Experimental.

[27] BRADEN, R., AND ZANG, L. RFC2209: Resource ReSerVation Protocol (RSVP) version 1 Message Processing Rules. Internet RFC, Sept. 1997. Informational.

[28] BRADEN (ED), R., ZANG, L., BERSON, S., HERZOG, S., AND JAMIN, S. RFC2205: Resource ReSerVation Protocol (RSVP) version 1 Functional Specification. Internet RFC, Sept. 1997. Standards Track.

[29] BRADNER, S. RFC2026BCP29: The Internet Standards Process Revision 3. Internet RFC, Harvard University, Oct. 1996. Best Current Practice.

[30] BRAKMO, L., O'MALLEY, S., AND PETERSON, L. Tcp vegas: New techniques for congestion detection and avoidance. In *SIGCOMM '94* (Aug. 1994), pp. 24–35.

[31] BRAKMO, L., AND PETERSON, L. Tcp vegas: End to end congestion avoidance on a global internet. *IEEE Journal on Selected Areas in Communication 13*, 8 (Oct. 1995), 1465–1480.

[32] CAO, P., AND IRANI, S. Cost-aware WWW proxy caching algorithms. In *USENIX Symposium on Internet Technologies and Systems* (Dec. 1997), pp. 193–206.

[33] CASE, J., FEDOR, M., SCHOFFSTALL, M., AND DAVIN, J. RFC1157/STD15: A Simple Network Management Protocol (SNMP). Internet RFC, May 1990. Standard.

[34] CENTERSPAN COMMUNICATIONS CORPORATION INC. Scour media exchange. Online: http://www.scour.com/.

[35] CHEREMY, R., AND KRAPF, A. IP Flow Meter. Online: http://www.via.ecp.fr/~ tibob/ipfm.

[36] CISCO SYSTEMS. Configuring Generic Traffic Shaping. Online: http://www.cisco.com/unvercd/cc/td/doc/product/software/ios120/12cgcr/qos_c/qcpart4/qcgts.htm, June 1999.

[37] CISCO SYSTEMS. Cisco IOS Release 12.0 Quality of Service Solutions Configuration Guide. Online: http://www.cisco.com/unvercd/cc/td/doc/product/software/ios120/12cgcr/qos_c/, 2000.

[38] CLARK, D. RFC813: Windows and Acknowledgement Strategy in TCP. Internet RFC, MIT, July 1982.

[39] COMBS, G. Ethereal: a network protocol analyzer. Online: http://ethereal.zing.org, 2000.

[40] CRAWLEY, E., NAIR, R., RAJAGOPALAN, B., AND SANDWICK, H. RFC2386: A Framework for QoS-based Routing in the Internet. Internet RFC, Aug. 1998. Informational.

[41] CROVELLA, M., AND BESTAVROS, A. Self-similarity in world wide web traffic: evidence and possible causes. In *ACM SIGMETRICS conference on Measurement and modeling of computer systems* (1996), ACM, pp. 160–169.

[42] DEERING, S., AND HINDEN, R. RFC2460: Internet Protocol Version 6 (IPv6) Specification. Internet RFC, Dec. 1998. Standards Track.

[43] DEPARTMENT OF COMPUTER SCIENCE, UNIVERSITY OF ARIZONA. Vegas home page. Online: http://www.cs.arizona.edu/protocols/.

[44] DEPARTMENT OF WATER AFFAIRS AND FORESTRY. The national water act - a pricing strategy for raw water use charges. Online: http://www-dwaf.pwv.gov.za/Documents/Policies/pricing.htm, Dec. 1998. Final Draft.

[45] DERI, L. Ntop. Online: http://www.ntop.org, 2000.

[46] DROMS, R. RFC2131: Dynamic Host Control Protocol. Internet RFC, Bucknell University, Mar. 1997. Standards Track.

[47] FIELDING, R., GETTYS, J., MOGUL, J., FRYSTYK, H., MASINTER, L., LEACH, P., AND BERNERSLEE, T. RFC2616: Hypertext Transport Protocol HTTP/1.1. Internet RFC, June 1999. Standards Track.

[48] FLOYD, S., AND HENDERSON, T. RFC2582: The NewReno Modification to TCP's Fast Recovery Algorithm. Internet RFC, Apr. 1999. Experimental.

[49] FLOYD, S., MAHDAVI, J., MATHIS, M., AND PODOLSKY, M. RFC2883: An Extension to the Selective Acknowledgement (SACK) Option for TCP. Internet RFC, July 2000. Standards Track.

[50] FOURIE, F., AND S. BANGAY. Using post office and postman agents to ensure reliable message delivery in distributed agent environments. In *SATNAC* (Sept. 2000).

[51] FOWLER, D. NetNews: Satellites: the new bandwidth-busters? *netWorker 2*, 5 (Dec. 1998), 5–10.

[52] FOWLER, D. NetNews: the search for bandwidth continues. *netWorker 2*, 1 (Feb. 1998), 5–8.

[53] FOX, R. RFC1106: TCP Big Window and Nak Options. Internet RFC, Tandem, June 1989.

[54] FREEBSD DOCUMENTATION TEAM. *sysctl(8) Kernel state control*, Sept. 1994. included in source distribution.

[55] FREEBSD DOCUMENTATION TEAM. *tcp(4) Internet Transmission Control Protocol*, Feb. 1995. included in source distribution.

[56] FREEBSD DOCUMENTATION TEAM. *dummynet(4) effective bandwidth manager and delay emulator*, Sept. 1998. Included in source distribution.

[57] GEWERTZMAN, J., AND SELTZER, M. World-wide web cache consistancy. In *USENIX Technical Conference* (Jan. 1996), pp. 141–151.

[58] GILFIX, M., AND COUCH, A. Peep (The Network Auralizer): Monitoring your Network with Sound. In *LISA* (Dec. 2000), no. XIV, pp. 109–118.

[59] GLOBALSCAPE INC. CuteMX homepage. Online: http://www.cutemx.com, 2000.

[60] GNUTELLA DEVELOPMENT TEAM. Gnutella homepage. Online: http://gnutella.wego.com, 2000.

[61] GRAHAM, R. FAQ: Firewall Forensics (what am i seeing?). Online: http://www.robertgraham.com/pubs/firewallseen.html, June 2000.

[62] GROSSMAN, D. draftietfdiffservnewterms03.txt: New Terminology for Diffserv. Tech. rep., Aug. 2000. Internet Draft Work in progress (expires March 2001).

[63] HALSE, G. Provisioning of Campus services using LDAP. Honours thesis, Rhodes University, Nov. 2000.

[64] HARDAKER, W. Net-snmp project. Online: http://net-snmp.sourceforge.net.

[65] HEADLIGHT SOFTWARE. GetRight homepage. Online: http://www.getright.com, 2000.

[66] HEINANEN, J., BAKER, F., WEISS, W., AND WROCLAWSKI, J. RFC2578: Assured Forwarding PHB Group. Internet RFC, June 1999. Standards Track.

[67] HEPTING, D. Whats a picture really worth? Online: http://fas.sfu.ca/~dhepting/personal/research/words/history.html.

[68] HERZOG, S. RFC2757: RSVP Extensions for Policy Control. Internet RFC, IPHighway, Jan. 2000. Standards Track.

[69] HEWLETT PACKARD. Netperf. Online: ftp://ftp.cup.hp.com/dist/networking/benchmarks/netperf/.

[70] HORWITT, E. The Politics of Policies. *Network World* (Dec. 1998). http://www.nwfusion.com/reviews/1012politics.html.

[71] HUBERT, B., AND MAXWELL, G. *Linux 2.4 Routing and Traffic Control HOWTO*, Aug. 2000. Online: http://www.ds9a.nl/2.4Routing/.

[72] HUITEMA, C., POSTEL, J., AND CROCKER, S. RFC1796: Not All RFCs are Standards. Internet RFC, Apr. 1995. Informational.

[73] HUSTON, G. RFC2990: Next Steps for the IP Qos Architecture. Internet RFC, Telstra, Nov. 2000. Informational.

[74] IBIBLIO.ORG. Metalab software repository. Online: http://www.ibilio.org/metalab/collection/index.html, 2000.

[75] IMESH.COM. Imesh homepage. Online: http://www.imesh.com, 2000.

[76] INFORMATION SCIENCES INSTITUE. RFC793STD7: Transmission Control Protocol. Internet RFC, Defense Advanced Research Projects Agency, Sept. 1981. Standard.

[77] INFORMATION SCIENCES INSTITUTE. RFC710: DOD Standard Transmission Control Protocol. Internet RFC, Defense Advanced Research Projects Agency, Jan. 1980.

[78] INFORMATION SCIENCES INSTITUTE. RFC791/STD5: Internet Protocol. Internet RFC, Defense Advanced Research Projects Agency, Sept. 1981. Standard.

[79] INFOSAT. Infosat homepage. Online: http://www.infosat.co.za/technology_overview.asp, 2000.

[80] INFOSAT. Infosat homepage. Online: http://www.infosat.co.za/, 2000.

[81] INTERNET ACTIVITIES BOARD. RFC1083: IAB Official Protocol Standards. Internet RFC, IAB, Dec. 1988.

[82] INTERNET ACTIVITIES BOARD. RFC1100: IAB Official Protocol Standards. Internet RFC, IAB, Apr. 1989.

[83] INTERNET ADVERTISING BUREAU. Banner standards. Online: http://www.banneradmuseum.com/Background.html, 2000.

[84] INTERNET ASSIGNED NUMBERS AUTHORITY. Assigned numbers. Online: http://www.iana.org.

[85] JACOBSON, V. Congestion avoidance and control. In *Symposium on Communications architectures and protocols* (1988), pp. 314–329.

[86] JACOBSON, V. Notes on using RED for Queue Management and Congestion Avoidance, June 1998. Slides produced for a presentation at NANOG 13.

[87] JACOBSON, V., AND BRADEN, R. RFC1072: TCP Extensions for Long-Delay Paths. Internet RFC, ISI, Oct. 1988.

[88] JACOBSON, V., BRADEN, R., AND BORMAN, D. RFC1323: TCP Extensions for High Performance. Internet RFC, May 1992.

[89] JACOBSON, V., BRADEN, R., AND ZANG, L. RFC1185: TCP Extensions for High-Speed Paths. Internet RFC, Oct. 1990.

[90] JACOBSON, V., LEARS, C., AND MCCANNE, S. *tcpdump(1) dump traffic on a network*, June 1997. included in source distribution.

[91] JACOBSON, V., NICHOLS, K., AND PODURI, K. RFC2598: An Expidited Forwarding PHB. Internet RFC, June 1999. Standards Track.

[92] KARRENBERG, D. tcpblast. Online: http://rucus.ru.ac.za/~bvi/utils/tcpblast.tgz (original).

[93] KENEIPP, R. Far and wide WAN service notes: QoS alternatives. *IT World.com* (Apr. 2000). Online: http://mithras.itworld.com/articles/columns/netkeneipp0421.html.

[94] KENT, C., AND MOGUL, J. Fragmentaion Considered Harmful. Tech. Rep. 87/3, DEC Western Research Labratory, Dec. 1987.

[95] KROEGER, T., LONG, D., AND MOGUL, J. Exploring the bounds of web latency reduction from caching and prefetching. In *USENIX Symposium on Internet Technologies and Systems (USITS)* (Dec. 1997).

[96] LAHEY, K. RFC2923: TCP problems with path MTU discovery. Internet RFC, dotRocket, Inc., Sept. 2000. Informational.

[97] LUKYANOV, A. V. LFTP homepage. Online: http://lftp.yar.ru/.

[98] MANKIN, A., BAKER, F., BRADEN, R., BRADNER, S., O'DELL, M., ROMANOW, A., WEINRIB, A., AND ZANG, L. RFC2208: Resource ReSerVation Protocol (RSVP) version 1 Applicability Statement some guidelines on deployment. Internet RFC, Sept. 1997. Informational.

[99] MATHIS, M., AND MAHDAVI, J. Forward Acknowledgement: Refining TCP Congestion Control. *ACM SIGCOMM 26*, 4 (Oct. 1996).

[100] MATHIS, M., MAHDAVI, J., FLOYD, S., AND ROMANOW, A. RFC1323: TCP Extensions for High Performance. Internet RFC, Oct. 1996. Standards Track.

[101] MCKENZIE, A. RFC1110: A Problem with the TCP Big Window Option. Internet RFC, BBN STC, Aug. 1989.

[102] MEIDER, W. 'A picture is worth a thousand words': From advertising slogan to American Proverb. *Southern Folklore*, 47 (1990), 207–225.

[103] MEIER, B. Network Administrator, University of Cape Town. Pers Comms, Apr. 2000.

[104] METALLICA CLUB. Metallica files suit against Napster, University of Southern California, Indiana University. Online: http://www.metallica.com/news/2000/00413a.html, Apr. 2000.

[105] METALLICA CLUB. Metallica sue Napster. Online: http://www.metallica.com/news/2000/napsterfacts.html, May 2000.

[106] METALLICA CLUB. Napster Lawsuit FAQ. Online: http://www.metallica.com/news/2000/napfaq.html, May 2000.

[107] MOGUL, J., AND DEERING, S. RFC1191: Path MTU discovery. Internet RFC, Nov. 1990.

[108] MOGUL, J., KENT, C., PARTRIDGE, C., AND MCCLOGHRIE, K. RFC1063: IP MTU discovery options. Internet RFC, July 1988.

[109] MONTENEGRO, G., DAWKINS, S., KOJO, M., MAGRET, V., AND VAIDYA, N. RFC2757: Long Thin Networks. Internet RFC, Jan. 2000. Informational.

[110] NAGLE, J. RFC896: Congestion Control in IP/TCP Internetworks. Internet RFC, Ford Aerospace, Jan. 1984.

[111] NAPSTER INC. Information about Metallica's request to disable Napster users. Online: http://www.napster.com/metallica-notice.html, May 2000.

[112] NAPSTER INC. Napster homepage. Online: http://www.napster.com, 2000.

[113] NATIONAL LABS FOR ADVANCED NETWORK RESEARCH. Squid Caching Web Proxy. Online: http://www.squid-cache.org, 2000. Software Homepage.

[114] NETSCAPE INC. Proxy client autoconfig ffile format. Online: http://home.netscape.com/eng/mozilla/2.0/relnotes/ demo/proxylive.html, Mar. 1996.

[115] NICHOLS, K., BLAKE, S., BAKER, F., AND BLACK, D. RFC2474: Definition of the Differentiated Services Field (DS Field) in IPv4 and IPv6 Headers. Internet RFC, Dec. 1998. Standards Track.

[116] NIELSEN, H., GETTYS, J., BAIRD-SMITH, A., PRUD'HOMMEAUX, E., H.LIE, AND LILLEY, C. Network performance effects of http/1.1, css1, and png. In *ACM SIGCOMM'97 conference on Applications, technologies, architectures and protocols for computer communications* (1997), pp. 155–166.

[117] ODLYZKO, A. Data networks are lightly utilized, and will stay tht way. Online: http://www.research.att.com/ãmo/doc/network.utilization.pdf, Oct. 1999.

[118] OETIKER, T. MRTG logfile format. Online: http://ee-staff.ethz.ch/˜oetiker/webtools/mrtg/ logfile.html.

[119] OETIKER, T. Round Robin Database Tool. Online: http://www.rrdtool.org.

[120] OETIKER, T. *rrdcreate(1) Man Page*, Feb. 2000. Included in source distribution.

[121] OETIKER, T., AND RAND, D. Multi Router Traffic Grapher. Online: http://www.mrtg.org.

[122] OPENBSD PROJECT. OpenSSH. Online: http://www.openssh.com/, Dec. 2000.

[123] PACKETEER INC. Controlling TCP/IP Bandwidth. Online: http://www.packeteer.com/technology/tcp_bw.pdf, Nov. 1998.

[124] PACKETEER INC. Managing UDP Traffic. Online: http://www.packeteer.com/technology/udp.pdf, Nov. 1998.

[125] PACKETEER INC. Assessing and Improving Network Efficiency with Packeteer's Packetshaper. Online: http://www.packeteer.com/technology/nwEfficiency.pdf, 2000.

[126] PACKETEER INC. On managing Controversy and Networks. Online: http://www.packeteer.com/solutions/napster.pdf, 2000.

[127] PACKETEER INC. Packeteer hompage. Online: http://www.packeteer.com, 2000.

[128] PACKETEER INC. PacketShaper. Online: http://www.packeteer.com/ products/packetshaper/, 2000.

[129] PADLIPSKY, M. RFC962: TCP-4 Prime. Internet RFC, Mitre-Bedford, Nov. 1985.

[130] PARKER, S., AND SCHMECEL, C. RFC2398: Some Testing Tools for TCP Implementors. Internet RFC, Sun Microsystems Inc., Aug. 1998. Informational.

[131] PAXSON, V., AND FLOYD, S. Wide-area traffic: the failure of Poisson modeling. In *Conference on Communication achitectures, protocols and applications* (London, Sept. 1994), ACM, pp. 257–268.

[132] PAXSON (ED), V., ALLMAN, M., DAWSON, S., FENNER, W., GRINER, J., HEAVENS, I., LAHEY, K., SEMKE, J., AND VOLZ, B. RFC2525: Known TCP Implementation Issues. Internet RFC, Mar. 1999. Informational.

[133] PLONKA, D. FlowScan. Online: http://net.doit.wisc.edu/~ plonka/FlowScan/.

[134] POSTEL, J. RFC346: Satellite Considerations. Internet RFC, UCLA-NMC, May 1972.

[135] POSTEL, J. RFC760: DOD Standard Transport Internet Protocol. Internet RFC, Jan. 1980. Editor.

[136] POSTEL, J. RFC768/STD6: User Datagram Protocol. Internet RFC, ISI, Aug. 1980. Standard.

[137] POSTEL, J. RFC792: Internet Control Message Protocol (ICMP). Internet RFC, ISI, Sept. 1981. Standard.

[138] POSTEL, J. RFC795: Service Mappings. Internet RFC, ISI, Sept. 1981.

[139] POSTEL, J. RFC879: The TCP Maximum Segment Size and Related Topics. Internet RFC, ISI, Nov. 1983.

[140] POSTEL, J. RFC1025: TCP and IP Bakeoff. Internet RFC, ISI, Sept. 1987.

[141] POSTEL, J. RFC1130: IAB Official Protocol Standards. Internet RFC, IAB, Oct. 1989. Editor.

[142] POSTEL, J. RFC1140: IAB Official Protocol Standards. Internet RFC, IAB, May 1990. Editor.

[143] POSTEL, J. RFC1200: IAB Official Protocol Standards. Internet RFC, IAB, Apr. 1993. Editor.

[144] POSTEL, J., AND REYNOLDS, J. RFC959: File Transfer Protocol. Internet RFC, ISI, Oct. 1985.

[145] PRICE WATERHOUSECOOPERS. IAB Internet Advertising Report: 2000 second quarter summary. Online: http://www.iab.net/reports, Sept. 2000.

[146] RAMAKRISHNAN, K., AND FLOYD, S. RFC2481: A Proposal to add Explicit Congestion Notification (ECN) to IP. Internet RFC, Jan. 1999. Experimental.

[147] REKHTER, Y., MOSKOWITZ, B., KARRENBERG, D., DE GROOT, G., AND LEAR, E. RFC1918/BCP5: Address Alloaction for Private Internets. Internet RFC, Feb. 1996. Best Current Practice.

[148] REMOTE COMMUNICATIONS INC. mod_gzip apache web server Internet Content Acceleration Module. Online: http://www.remotecommunications.com/apache/mod_gzip/, Oct. 2000.

[149] REYNOLDS, J., AND BRADEN, R. STD1: Internet Official Protocol Standards. Internet RFC, IETF, Aug. 2000. Editors, Standards Track.

[150] RITTER, J. ngrep homepage. Online: http://www.packetfactory.net/Projects/Ngrep/, 2000.

[151] RIZZO, L. IP Dummynet. Online: http://www.iet.unipi.it/~luigi/ip_dummynet/.

[152] RIZZO, L. Dummynet: a simple approach to the evaluation of network protocols. *ACM Computer Communication Review 27*, 1 (Jan. 1997), 31–41. Online: http://www.acm.org/sigcomm/ccr/archive/97/jan97/ccr-9701-rizzo,pdf.

[153] RIZZO, L., AND VICISANO, L. Replacement policies for a proxy cache. Technical Report RN/98/13, University College London, Department of Computer Science, 1998.

[154] RODRIGUEZ, P., AND BIERSACK, E. Bringing the Web to the network edge: Large Caches and Satellite Distribution. In *MONET Special issue on Satellite-based information services* (Jan. 2000).

[155] ROESCH, M. Snort homepage. Online: http://www.clark.net/~roesch/security.html, 2000.

[156] SHENKER, S., PARTRIDGE, C., AND GUERIN, R. RFC2122: Specification of Guarenteed Quality of Service. Internet RFC, Sept. 1997. Standards Track.

[157] SIMPSON, J. B., Ed. *Simpson's Contemporary Quotations*. Houghton Mifflin, Boston., Online: http://www.bartleby.com/63/ [Out of Print], 1988.

[158] SIYANDA. Siyanda Broadband Satellite Communications. Online: http://www.siyanda.co.za.

[159] SMITH, T. Scour to stop Napster style movie sharing. *The Register* (Nov. 2000). Online: http://www.theregister.co.uk/content/archive/14789.html.

[160] SNERT.COM. mod_throttle homepage. Online: http://www.snert.com/Software/mod_throttle/, 2000.

[161] ST. JOHNS, M. RFC1413: Identification Protocol. Internet RFC, US Department of Defense, Feb. 1993.

[162] STENGER, R. Campuses seek compromise over popular bandwidth hog. *CNN.com* (Apr. 2000). Online: http://www.cnn.com/2000/TECH/computing/03/01/napster.ban/.

[163] STEVENS, W. *TCP/IP Illustrated: The Protocols*, vol. I. Addison-Wesley, Jan. 1994.

[164] STEVENS, W. *TCP/IP Illustrated: TCP for transactions, HTTP, NNTP and the Unix Domain Protocol*, vol. III. Addison-Wesley, Jan. 1996.

[165] STEVENS, W. RFC2001: TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms. Internet RFC, NOAO, Jan. 1997. Standards Track.

[166] TALCOTT, S. Restriction of Internet access at UCBerkeley to continue. *Daily Californian* (Mar. 2000). Online: http://lists.fsu.edu/pipermail/nolenet/2000March/002241.html.

[167] TCPDUMP. Tcpdump/libpcap Homepage. Online: http://www.tcpdump.org.

[168] TEWARI, R., DAHLIN, M., VIN, H., AND KAY, J. Beyond Heirarchies: Design Considerations for Distributed Caching on the Internet. Tech. Rep. TR98-04, University of Texas, Austin, 1998. Online: http://www.cs.utexas.edu/users/dmcl/papers/ps/TR9804.ps.

[169] TREND MICRO INC. WebManager homepage. Online: http://www.antivirus.com/products/webmanager/, 2000.

[170] TSAOUSSIDIS, V., BADR, H., GE, X., AND PENTIKOUSIS, K. Energy/Throughput Tradeoffs of TCP Error Control Strategies. In *IEEE Symposium on Computers and Communications* (2000), IEEE.

[171] TUCOWS INC. Tucows software repository. Online: http://www.tucows.com.

[172] VANDERBILT UNIVERSITY. Napster and bandwidth usage. Online: http://www.vanderbilt.edu/resnet/napster.html, 2000.

[173] VOROBYEV, V. Trafshow. Online: ftp://ftp.nsk.su/pub/RinetSoftware/.

[174] WALDBUSSER, S. RFC2819/STD59: Remote Network Monitoring Managment Information Base. Internet RFC, Lucent Technologies, May 2000. Standards Track.

[175] WANG, J. A survey of web caching schemes for the internet. *ACM Computer Comunication Review 29*, 5 (Oct. 1999), 36–46.

[176] WELCHER, P. RMON. Tech. rep., Mentor Technologies, Oct. 1996. Online: http://www.mentortech.com/learn/welcher/papers/rmon.htm.

[177] WELCHER, P. Threshold Manager. Tech. rep., Mentor Technologies, Mar. 1997. Online: http://www.mentortech.com/learn/welcher/papers/thresh.htm.

[178] WELCHER, P. QoS (Quality of Service) Features. Tech. rep., Mentor Technologies, Oct. 1998. Online: http://www.mentortech.com/learn/welcher/papers/qos1.htm.

[179] WELCHER, P. Quality of Service, Part II. Tech. rep., Mentor Technologies, Nov. 1998. Online: http://www.mentortech.com/learn/welcher/papers/qos2.htm.

[180] WELCHER, P. Quality of Service, Part III. Tech. rep., Mentor Technologies, Nov. 1998. Online: http://www.mentortech.com/learn/welcher/papers/qos3.htm.

[181] WESSELS, D. *Squid Frequently Asked Questions*, 2001. Online: http://www.squid-cache.org/Doc/FAQ/.

[182] WESSELS, D. Squid: Related software. Online: http://www.squid-cache.org/related-software.html, Jan. 2001.

[183] WESSLES, D., AND CLAFFY, L. RFC2186: Internet Cache Protocol (ICP) version 2. Internet RFC, NLANR/UCSD, Sept. 1997. Informational.

[184] WESSLES, D., AND CLAFFY, L. RFC2187: Application of Internet Cache Protocol (ICP) version 2. Internet RFC, NLANR/UCSD, Sept. 1997. Informational.

[185] WILLIAMS, S., ABRAMS, M., STANDRIDGE, C., ABDULLA, G., AND FOX, E. Removal policies in network caches for worldwide web documents. In *Appliations, technologies, architectures, and protocols for computer communications* (1996), pp. 293–305.

[186] WILLIAMS, T., AND KELLEY, C. Gnuplot: An interactive plotting program. Online: http://www.gnuplot.org, Dec. 1998.

[187] WINZIP COMPUTING INC. WinZip homepage. Online: http://www.winzip.com/, 2000.

[188] WOLMAN, A., VOELKER, G., SHARMA, N., CARDWELL, N., KRLIN, A., AND LEVY, H. On the scale and performance of cooperative web proxy caching. In *17th ACM symposium on operating system principles* (1999), ACM, pp. 16–31.

[189] WROCLAWSKI, J. RFC2209: Tthe use of RSVP with IETF Integrated Services. Internet RFC, MIT LCS, Sept. 1997. Standards Track.

[190] YANG, J., WEI, W., AND MUNTZ, R. Collaborative web caching based proxy affinities. In *International conference on Measurements and modeling of computer systems* (2000), pp. 78–89.

[191] YAVATKAR, R., HOFFMAN, D., BERNET, Y., BAKER, F., AND SPEER, M. RFC2814: SBM (Subnet Bandwidth Manager): A protocol for RSVP-based admission control over IEEE 802style networks. Internet RFC, May 2000. Standards Track.

Appendix A

Example Configurations

A.1 Squid

The following squid configuration file is a stripped down version of the configuration file used on the RUCUS system, and contains examples of content filtering, delay pools and http_access control. For details on the various tags detailed in the file below, the reader is referred to the Squid documentation, and in particular the example `squid.conf` file that ships with the squid source distribution. Where appropriate, some description has been provided to explain the function of a particular tag.

```
#$Id: squid.conf,v 1.30 2000/11/21 19:30:56 bvi Exp $
#
# Example Squid configuration file
# For further details as to what options do,
# look at the squid.conf file provided with squid


# NETWORK OPTIONS
# ----------------------------------------

# Network addresses we listen on
http_port rucus.ru.ac.za:3128
http_port 127.0.0.1:3128
```

```
http_port 192.168.231.1:3128

# Force all traffic to come from RUCUS
tcp_outgoing_address 146.231.29.2


# OPTIONS WHICH AFFECT THE NEIGHBOUR SELECTION
# ----------------------------------------------------
# Peer Caches that can be queried
cache_peer wwwproxy.ru.ac.za    par-
ent  3128    3130    default
cache_peer diablo.cs.ru.ac.za   sibling 3128    3130
cache_peer segv.ru.ac.za        sibling 3128    3130

# force squid not query peers in the case of
# local ru.ac.za addresses
cache_peer_domain wwwproxy.ru.ac.za !ru.ac.za
cache_peer_domain segv.ru.ac.za !ru.ac.za
cache_peer_domain diablo.cs.ru.ac.za !ru.ac.za

# Use this to detect when peers (esp. siblings) die
dead_peer_timeout 120 seconds

# directly fetch from source
# (through parent if need be)
# urls containing these strings
hierarchy_stoplist cgi-bin ?

# Do not cache the output of CGI's and the like
acl QUERY urlpath_regex cgi-bin \.cgi \?
no_cache deny QUERY


# OPTIONS WHICH AFFECT THE CACHE SIZE
# -----------------------------------------------
# how much to use for holding objects in ram
cache_mem  8 MB

# The maximum object size held in the cache.
```

```
maximum_object_size 4096 KB

# the minimum object size held in cache.
minimum_object_size 0 KB


# LOGFILE PATHNAMES AND CACHE DIRECTORIES
# -----------------------------------------------
#
# where the cache is to be stored
cache_dir ufs /usr/local/squid/cache 200 16 256

# log file locations
cache_access_log /usr/local/squid/logs/access.log
cache_log /usr/local/squid/logs/cache.log
cache_store_log /usr/local/squid/logs/store.log
cache_swap_log /usr/local/squid/cache/swap.log
useragent_log /usr/local/squid/logs/useragent.log

# Important for most squid log processing tools
emulate_httpd_log off

# A customised MIME table for squid to use.
mime_table /usr/local/etc/squid/mime.conf

# Optionally log the mime headers for data
log_mime_hdrs off

# Set what should be stored in log files
# and at what level of  detail
debug_options ALL,1

# dont bother resolving IP's - speeds operation up
log_fqdn off

# Provides a way for anonymising client data
#client_netmask 255.255.255.255
```

```
# OPTIONS FOR EXTERNAL SUPPORT PROGRAMS
# -----------------------------------------
# These would be enabled in order to make use
# of redirectors such as SquidGuard
# redirect_program /usr/local/bin/squidguard
# redirect_children 5

# not be a wanted effect of a redirector.
#redirect_rewrites_host_header on

#  TAG: redirector_access
#  If defined, this access list specifies which
#  requests are sent to the redirector processes.
#  By default all requests are sent.
#  Provides for optimisation.

#  Authentication (currently not used)
#authenticate_program none
#authenticate_children 5
#authenticate_ttl 3600
#authenticate_ip_ttl 0


# OPTIONS FOR TUNING THE CACHE
# ------------------------------------
#limit the max request size from clients
request_header_max_size 10 KB

#  TAG: request_body_max_size (KB)
request_body_max_size 100 KB

#  TAG: reply_body_max_size (KB)
reply_body_max_size 25 MB

# How often to refresh cache content.
# Squid Defaults:
refresh_pattern ^ftp:           1440    20%     10080
refresh_pattern ^gopher:        1440    0%      1440
refresh_pattern  .               0       20%     4320
```

```
# Expire objects form the cache older than this
reference_age 6 months

# allow for squid to per-
form some magic on aborted downloads in an effort to
# save bandwidth
quick_abort_min 16 KB
quick_abort_max 16 KB
quick_abort_pct 95

# when handling range requests, only return
# what the client asks fordo not pre-fetch
# the file up to that point
range_offset_limit 0 KB


# ACCESS CONTROLS
# ---------------------------------

# Access Control lists used by http_access
# and other  commands
#Defaults:
acl all src 0.0.0.0/0.0.0.0     # Any machine
acl manager proto cache_object
acl localhost src 127.0.0.1/255.255.255.255
acl SSL_ports port 443 563
# these are ports that squid can connect to
# multiple declarations of ACLs with the same names
# append the data to the acl - better readability
acl Safe_ports port 80 81 21 443 563 70 210
acl Safe_ports port 280         # http-mgmt
acl Safe_ports port 488         # gss-http
acl Safe_ports port 591         # filemaker
acl Safe_ports port 777         # multiling http
acl CONNECT method CONNECT

#local RUCUS Stuff
```

```
acl snmppublic snmp_community public

acl rucus src 146.231.29.2/32
# RUCUS and its vhosts
acl rucus-subnet src 146.231.29.0/26
acl kaos src 146.231.29.8/32 192.168.231.0/24
acl bvi-home src 146.231.97.74/32
acl rhodes src 146.231.0.0/16

acl peer-squid src 146.231.31.122/32 146.231.26.70/32
acl root ident root
# These are used in traffic shaping
# see examples later on.
acl badperson ident
   ‘‘/usr/local/etc/squid/lists/banned’’
acl lowuser ident ‘‘/usr/local/etc/squid/lists/lowusers’’
acl hiuser ident ‘‘/usr/local/etc/squid/lists/highusers’’

# Authentication using the Proxy_AUTH mechanism
acl auth_ok proxy_auth ‘‘/usr/local/etc/squid/lists/authlist’’

# IP based vhosts on RUCUS
acl vhosts ident ‘‘/usr/local/etc/squid/lists/vhost’’

# File based blocklists for unwanted content
# (see examples in Appendix A)
acl multimedia urlpath_regex -i
  ‘‘/usr/local/etc/squid/lists/block-mm’’
acl images url_regex -i
  ‘‘/usr/local/etc/squid/lists/block-images’’


# Actual HTTP access controls
# Define what is allowed access
http_access allow manager localhost
http_access deny manager
http_access deny !Safe_ports
http_access deny CONNECT !SSL_ports
```

```
# deny specified types unless re-
quested by the root user
http_access deny rucus multimedia !root
http_access deny rucus images !root

http_access deny badperson
http_access deny vhosts
http_access allow localhost
http_access allow rucus
http_access allow kaos
http_access allow bvi-home
http_access allow peer-squid
http_access allow auth_ok
http_access deny rhodes

# deny anything we havent explicitly allowed above
http_access deny all

# Allow peers to query us
icp_access allow peer-squid
icp_access deny all

# only allow clients matched by these rules to
# use this Squid to fetch MISS data
miss_access allow rucus
miss_access allow kaos
miss_access deny rhodes

# The Realm for our squid authentication
proxy_auth_realm RUCUS Squid

# perform ident lookups on all rhodes hosts so as to
# see who is trying to bypass University restrictions
ident_lookup_access allow  rhodes
ident_lookup_access deny all


# Other Parameters
#---------------------------------------
```

```
#
# Append this domain to all hostnames without a .
# www ==> www.ru.ac.za
append_domain .ru.ac.za

# No Need since all requests are coming
# from the local machine
forwarded_for off

# use DIRECT access for FTP requests
acl rhodes-servers dst 146.231.0.0/255.255.0.0
acl ru-servers dstdomain ru.ac.za
acl FTP proto FTP

always_direct allow FTP
always_direct allow SSL_Ports

# ..and a parent for stuff outside of Rhodes
never_direct deny ru-servers
never_direct deny rhodes-servers
never_direct allow all

# DELAY POOL PARAMETERS
# all require DELAY_POOLS compilation option
# --------------------------------------------

#################### RUCUS DELAY STUFF###############
delay_pools 2           # 2 delay pools
delay_class 1 1         # pool 1 class 1
delay_class 2 1         # pool 2 class 1

delay_access 1 allow lowuser all
delay_access 1 deny all

delay_access 2 allow hiuser all
delay_access 2 deny all

delay_parameters 1 8000/16000 # 8K fill , 16K burst
delay_parameters 2 2000/4000  # 2K fill ,  4K burst
```

```
# Initial level that delay buckets should be set at
delay_initial_bucket_level 50
```

A.1.1 Example User and host lists

These are files contain usernames to which various delay pool effects are applied. The lists always contain the user nobody, as squid requires a file based ACL to have at least one entry. These entries correspond with the output shown in Figure 4.4. The AUTO-GENERATED header is used as a reminder to administrators that these files should not be edited as changes will be lost when the lists are re-created.

hidelay

```
#AUTO-GENERATED :significant delay
nobody
jockf
ggvb
madkev
```

lowdelay

```
#AUTO-GENERATED :Users in low delay
nobody
quark
donovan
nakiz
```

banned

```
#AUTO-GENERATED :Users banned for abuse
nobody
russell
```

Authentication file

```
bvi:VD7qCU5fElxj2
guy:pzm1prho.5DbA
nbm:GieLXE7bHj6pM
```

A.1.2 Content Type Blocklists

The following files are uses as a file based ACLs to filter out content to users of the RUCUS squid. the system is text based hence content of the following mime types `image/*`. `video/*` and `audio/*` cannot be viewed, and is therefor blocked. External file lists are used in order to provide for a cleaner `squid.conf`.

block-images

```
# Images BLOCKLIST
# $Id: block-
images,v 1.1 2000/06/09 13:18:58 bvi Exp $
\.gif$
\.png$
\.jpg$
\.jpeg$
```

block-mm

```
# Multimedia BLOCKLIST
# $Id: block-
mm,v 1.1 2000/06/09 13:18:41 bvi Exp $
\.mp?$
\.ra$
\.wav$
\.avi$
\.ram$
\.a.x$
\.mov$
```

A.1.3 Example `proxy.pac`

This is the automated proxy configuration file used at Rhodes University.

```
// Automatic Proxy Configuration File
// Last update:  19990611
// Location:  http://www.ru.ac.za/proxy.pac
function FindProxyForURL(url,host)
{
 // Is it one of the supported protocols ?
 if( (url.substring(0,5)!="http:") &&
     (url.substring(0,4)!="ftp:") &&
     (url.substring(0,7)!="gopher:") )
  return "DIRECT";

 // Is it a plain hostname ?
 if(isPlainHostName(host))
  return "DIRECT";

 // Is it the 'localhost' alias ?
 if( shExpMatch(host,"localhost.*") ||
     shExpMatch(host,"127.0.0.1") )
  return "DIRECT";

 // Is it on the local network?
if( shExpMatch(host,"*.ru.ac.za") ||
    shExpMatch(host,"*.rhodes.ac.za") ||
    shExpMatch(host,"scifest.on.web.za") ||
    shExpMatch(host,"schoolfest.on.web.za") ||
    shExpMatch(host,"www.highwayafrica.org.za") ||
```

```
    shExpMatch(host,"146.231.*.*") )
 return "DIRECT";

 // We passed all exception rules, so...
 // Return a cache appropriate to our subnet.
if (isInNet(host, "146.231.40.0", "255.255.248.0"))
  return "PROXY 146.231.128.8:3128; " +
         "PROXY cache3.ru.ac.za:3128";
else
  return "PROXY 146.231.128.8:3128; " +
         "PROXY cache3.ru.ac.za:3128";
}
```

A.2 IPFW Example

These firewall rules are an extract of appropriate rules from the RUCUS firewall configuration , and illustrate the setup of a traffic shaping system as described in Section 4.3.3. This is **NOT** a complete firewall configuration, and implements **NO** security measures.

A.2.1 Traffic Shaping

```
# 40000  traffic shaping
# 40nnn  End the search for things which shouldn't be shaped
# incoming from RU
add 40000 allow ip from 146.231.0.0/16 to 146.231.29.0/26
# outgoing to RU
add 40001 allow ip from 146.231.29.0/26 to 146.231.0.0/16
# IRC server connections, and outgoing mail
```

```
add 40002 allow tcp from any 6667,8000,smtp to 146.231.29.0/26
add 40003 allow tcp from 146.231.29.0/26 to any 6667,8000,smtp
# don't slow the wheels
add 40004 allow ip from any to 146.231.29.0/26 gid wheel
add 40005 allow ip from 146.231.29.0/26 to any gid wheel
# Traffic coming from kaos system (VPN)
add 40006 allow ip from 192.168.231.0/26 to any
add 40007 allow ip from any to 192.168.231.0/26
# do not affect any ICMP
add 40008 allow icmp from any to any

# Create the pipes
# Incoming pipe for users
pipe 1 config bw 20KBytes/s delay 10 queue 128KBytes
# Outgoing pipe  for users
pipe 2 config bw 20KBytes/s delay 10 queue 128KBytes
# separate pipe for other services
pipe 3 config bw 32KBytes/s delay 1000 queue 128KBytes

# 41nnn  Shape traffic
# avoid pipe 3 traffic going in pipe 1
add 41000 skipto 41020 tcp from 146.231.29.0/26 ftp,ftp-
data,http,https to any
add 41001 skipto 41021 tcp from any to 146.231.29.0/26 ftp,ftp-
data,http,https
# shape any other external user traffic
add 41010 pipe 1 ip from any to 146.231.29.0/26
add 41011 pipe 2 ip from 146.231.29.0/26 to any
# shape external FTP and HTTP separately
add 41020 pipe 3 tcp from 146.231.29.0/26 ftp,ftp-
data,http,https to any
add 41021 pipe 3 tcp from any to 146.231.29.0/26 ftp,ftp-
data,http,https
```

A.2.2 Traffic Accounting

```
# 20000  Counters
#=============================================================
#
# For section 20000, a rule number ending in
# 0 counts incoming traffic
# 1 counts outgoing traffic
# 20nnn HTTP/HTTPS
#Example:
#add 20nn0 count tcp from any to <host> http,https

#add 20nn1 count tcp from <host> http,https to any
```

```
# incoming rucus http
add 20000 count tcp from any to rucus.ru.ac.za http,https
# outgoing rucus http
add 20001 count tcp from rucus.ru.ac.za http,https to any
# incoming sport http
add 20010 count tcp from any to sport.ru.ac.za http,https
# outgoing sport http
add 20011 count tcp from sport.ru.ac.za http,https to any
# incoming soc http
add 20020 count tcp from any to soc.ru.ac.za http,https
# outgoing soc http
add 20021 count tcp from soc.ru.ac.za http,https to any
# incoming on.web.za http
add 20030 count tcp from any to on.web.za http,https
# outgoing on.web.za http
add 20031 count tcp from on.web.za http,https to any
# incoming rmr http
add 20040 count tcp from any to rmr.ru.ac.za http,https
# outgoing rmr http
add 20041 count tcp from rmr.ru.ac.za http,https to any
# incoming src http
add 20050 count tcp from any to src.ru.ac.za http,https
# outgoing src http
add 20051 count tcp from src.ru.ac.za http,https to any
# incoming moria.org http
add 20060 count tcp from any to moria.org http,https
# outgoing moria.org http
add 20061 count tcp from moria.org http,https to any
# incoming acts.org.za http
add 20070 count tcp from any to acts.org.za http,https
# outgoing acts.org.za http
add 20071 count tcp from acts.org.za http,https to any
# incoming debating.org.za http
add 20080 count tcp from any to debating.org.za http,https
# outgoing debating.org.za http
add 20081 count tcp from debating.org.za http,https to any
# incoming saints.za.net http
add 20090 count tcp from any to saints.za.net http,https
# outgoing saints.za.net http

add 20091 count tcp from saints.za.net http,https to any


# 21nnn  FTP (not easy to count passive data connections)
# incoming from external servers
add 21000 count tcp from any ftp,ftp-data to rucus.ru.ac.za
# outgoing to external servers
add 21001 count tcp from rucus.ru.ac.za to any ftp,ftp-data
# incoming to rucus' site
add 21010 count tcp from any to ftp.rucus.ru.ac.za ftp,ftp-data
# outgoing from rucus' site
```

```
add 21011 count tcp from ftp.rucus.ru.ac.za ftp,ftp-data to any


# 22nnn  IRC
# incoming from external servers
add 22000 count tcp from not 146.231.29.0/26 8000,6667 to ru-
cus.ru.ac.za
# outgoing to external servers
add 22001 count tcp from ru-
cus.ru.ac.za to not 146.231.29.0/26 8000,6667
# incoming from ZAnet clients
add 22010 count tcp from any to rucus.zanet.net 6667
# outgoing to ZAnet clients

add 22011 count tcp from rucus.zanet.net 6667 to any
# incoming from LagNet servers
add 22020 count tcp from any 8000,6667 to lagnet.rucus.ru.ac.za
# outgoing to LagNet servers
add 22021 count tcp from lagnet.rucus.ru.ac.za to any 8000,6667
# incoming from LagNet clients
add 22030 count tcp from any to lagnet.rucus.ru.ac.za 6667
# outgoing to LagNet clients

add 22031 count tcp from lagnet.rucus.ru.ac.za 6667 to any


# 23nnn  CVSup
# incoming from CVSup servers (cvsupin)
add 23000 count tcp from any cvsup to rucus.ru.ac.za
# outgoing to CVSup servers (cvsupin)
add 23001 count tcp from rucus.ru.ac.za to any cvsup
# incoming from CVSup clients (cvsupd)
add 23010 count tcp from any to rucus.ru.ac.za cvsup
# outgoing to CVSup clients (cvsupd)

add 23011 count tcp from rucus.ru.ac.za cvsup to any


# 24nnn  Squid and ICP
# incoming TCP
add 24000 count tcp from wwwproxy.ru.ac.za 3128 to rucus.ru.ac.za
# outgoing TCP
add 24001 count tcp from rucus.ru.ac.za to wwwproxy.ru.ac.za 3128
# incoming ICP
add 24010 count udp from wwwproxy.ru.ac.za 3130 to rucus.ru.ac.za
# outgoing ICP

add 24011 count udp from rucus.ru.ac.za to wwwproxy.ru.ac.za 3130



# 25nnn  SMTP
# incoming for outgoing mail
```

```
add 25000 count tcp from any smtp to 146.231.29.0/26
# outgoing for outgoing mail
add 25001 count tcp from 146.231.29.0/26 to any smtp
# incoming for incoming mail
add 25010 count tcp from any to 146.231.29.0/26 smtp
# outgoing for incoming mail

add 25011 count tcp from 146.231.29.0/26 smtp to any


# 26nnn  MUD
add 26000 count tcp from not 146.231.0.0/16 2000 to any  # incom-
ing MUD traffic

add 26001 count tcp from any to not 146.231.0.0/16 2000  # outgo-

ing MUD traffic
# 29nnn  Summary counters
add 29000 count ip from any to any in  # total incoming
add 29001 count ip from any to any out # total outgoing
# total incoming from core switch
add 29010 count ip from any to any in via rl0
# total outgoing to core switch
add 29011 count ip from any to any out via rl0
# total incoming from kaos
add 29020 count ip from any to any in via rl1
# total outgoing to kaos
add 29021 count ip from any to any out via rl1
# don't count RFC1918 addresses as external
add 29030 skipto 30000 ip from any to 192.168.231.0/26
add 29031 skipto 30000 ip from 192.168.231.0/26 to any
# incoming external traffic counters
add 29040 skipto 30000 ip from not 146.231.0.0/16 to any
# outgoing external traffic counters
add 29041 skipto 30000 ip from any to not 146.231.0.0/16
# from RU to rucus
add 29050 count tcp from not 146.231.29.0/26 to 146.231.29.0/26
# from rucus to RU
add 29051 count tcp from 146.231.29.0/26 to not 146.231.29.0/26
```

A.3 Cisco Router Configuration

The following is an extract of appropriate portions of a Cisco router configuration file with traffic shaping enabled.

A.3.1 Traffic Accounting

Traffic accounting can be set up on most Cisco routers running IOS 10.0 or higher [36]. Activating traffic accounting on an interface results in a table with the format described below (Table A.1) being created. This table stores traffic details for each unique source and destination host pair. This table can consume a large amount of router memory, and so the threshold value specified should be within the routers memory limits. The design and mode of network access should also be taken into account when deciding on a threshold value. If all internal clients access the Internet through an NAT gateway or application level proxy, the host count is likely to my much lower, and consequently the threshold value can be lowered. A rough indication of the memory required can be determined using the following formula:

$$Threshold\,value \times ((2 \times host\,size) + (2 \times counter\,size))$$

For the example configuration below, this would calculate to

$$16000 \times ((2 \times 4) + (2 * 16) = 640000$$

Since these values are in Bytes, the conversion shows that it use 640Kbytes of RAM. The values chosen are the 4 bytes for each host (the size of an IP address), and 16 bytes for each counter. The threshold value of 16000 is what is currently being used at Rhodes University with approximately 4 000 active hosts on the network.

Table A.1: Cisco IP accounting format

Record	Source IP	Destination IP	Traffic IN	Traffic Out
...	146.231.26.70	196.4.160.12	598210	6790
1562	146.231.29.2	196.231.41.132	1000	43242
...	207.25.71.26	146.231.128.8	76521	300

The following commands can be entered in the configuration mode for each interface that one wishes to activate IP accounting on.

```
interface Ethernet0
!enable accounting on the interface
ip accounting
! set the accounting limit to 16000 src\dst host pairs
ip accounting-threshold 16000
```

Accounting data can be retrieved by means of the show IP accounting command, which produces output similar to the following

```
GF-1#show ip accounting
Source Destination Packets Bytes
207.159.128.2  196.25.141.130 1    72
146.64.8.4     196.25.141.134 29   7080
207.159.128.10 196.25.141.130 1    72
198.41.3.38    196.25.141.134 1    307
146.231.128.6  196.25.141.130 1    65
146.231.128.8  196.25.141.130 1    65
196.22.194.129 196.25.141.134 1    243
198.41.3.101   196.25.141.134 6    1121
64.208.37.3    196.25.141.135 11   832
146.231.29.35  196.25.141.134 7    598
146.231.29.2   196.25.141.134 109  72276
196.25.69.143  196.25.141.134 7    1190
204.216.27.3   196.25.141.134 1    273
146.231.26.70  196.25.141.135 1217 1093608
```

This output can then be parsed and processed by an external program.

A.3.2 Delay Configuration

Traffic shaping on Cisco routers [37] [36] can be configured by setting up appropriate access-lists to match the traffic to be shaped. For each interface on which shaping should occur, a traffic-shape group should be established, defining the

shaping parameters, and the access-list to be used to determine the traffic to be processed. An example setup is shown below.

```
! Set the interface
interface Ethernet0
!Clear the current shaping setup.
no traffic-shape group <access-list>
! Add a new setup
access-list 184 permit tcp 146.231.24.200 0.0.0.0 any
access-list 184 permit udp 146.231.24.200 0.0.0.0 any
traffic-shape group 184 8192 2048 8192
```

The `traffic-shape group` command takes parameters in the format:

`traffic-shape` group `<access-list> CIR (bps) [Bc (bits) [Be(bits)]]`

access list	the access list to be applied in order to match traffic
CIR	Committed Information Rate - the bitrate that traffic is shaped to (the CIR)
Bc	burst-size - the number of sustained bits that can be transferred per interval. This defaults to (CIR/8)
Be	excess-burst-size - the maximum number of bits that can exceed the burst size in the first interval of congestion.

The *exec mode* command `'show traffic statistics'` can be used to show the current state of traffic-shaping statistics, an example of which is presented in Figure A.1. This Figure shows the number of packets and bytes delayed and which access-list rules they were delayed by. Also shown is the queue depth, the maximum depth to which packets for a particular access-list were queued. The size of the is dependant on the particular router configuration. Once a queue is full, packets are dropped. By analysing performance data such as this, the administrator can see the effect of the access-list shaping, and also gain an indication of whether queue sizes may need to be increased.

```
Access Queue Packets Bytes    Packets Bytes  Shaping
I/F  List   Depth Delayed Delayed  Active
Et0  192    35    22154   13711656 10870   6800944 yes
Et0  195    44    3325    2234213  1381    1087680 yes
Et0  194    15    20743   1483961  10061   719804  yes
Se0  184    59    16671   13847014 8335    6922755 yes
Se0  182    0     0       0        0       0       no
```

Figure A.1: Example Traffic Shaping Statistics

Appendix B

An analysis of IP traffic on the Rhodes University network

This study was undertaken between the 29th of November 1999 and the 8th of December 1999. IP packet headers were captured on the monitor system (as described in Case Study 2 3.1.4). These results were then processed and summarised. The total size of the captured packet headers amounted to nearly 16 gigabytes. The summary data for this study contained on the accompanying CD-ROM. The summary of the traffic data collected presented below, is restricted to the top 15 ports for TCP and UDP ordered both by total byte-count, and by traffic volume as measured by the packet-count. The researcher has become aware of several anomalies in the data which came to light during analysis. These may have been due to problems with the initial process of data capture, or the processing software which was developed. However due to time constraints, this experiment was not able to be repeated.

Ports were classified according to their official IANA designation [84]. Robert Graham's Firewall Forensics FAQ [61] also proved useful.

B.1 TCP Traffic

B.1.1 Source Ports

The top values revealed in the analysis of the most popular TCP source ports as shown in Tables B.1 and B.2 held few unexpected results. HTTP followed by Squid Cache responses and FTP data transfers were shown to be the most popular ports both in terms of the quantity of data transferred, as well as the packet count. Traffic attributable to the relatively widespread use of Napster on campus is shown in 4th and 5th place in terms of the data transferred. These places are lower down in the packet count ranking, most likely because of Napster's stream oriented design where there are large data transfers as opposed to HTTP where the size of content varies greatly. The placing of SMTP in 4th place in the packet ranking and absence from the size ranking (37th place) is most likely due to the high number of ACK packets. NNTP is also shown to be a notable user of bandwidth. The anomalies of positions 7 to 14 are most likely one-off transfers – most likely movie files considering their byte counts – since their packet count is very low. Table B.2 provides a more realistic view as to what the popular protocols are.

B.1.2 Destination Ports

Email traffic is shown to be the top of the list in the size ranking shown in Table B.3 , with Napster traffic being placed 2nd and 4th, and NNTP in 3rd place. The packet based ranking (Table B.4). The unidentified ports in Table B.3 are most likely the result of Passive FTP transfers. An item of interest in Table B.4 is the exponential decline in the packet count for the ports listed. As with the source ports, the packet count provides a useful indication of the popularity of protocols on a network link.

Table B.1: Top 15 TCP source ports by total traffic size

TCP Port	Size (bytes)	Protocol
80	16 632 688 441	HTTP
3128	6 680 628 717	Squid cache
20	5 724 392 677	FTP Data
6699	2 644 888 554	Napster
6688	1 211 968 314	Napster
443	700 333 148	HTTPS
1954	520 791 452	
1944	520 532 452	
1936	519 328 107	
1937	515 518 197	
1938	514 306 980	
1940	512 663 669	
1968	507 522 638	
7998	36 8273369	
119	31 704 598	NNTP

Table B.2: Top 15 TCP source ports by packet count

TCP Port	# of packets	Protocol
80	22 337 624	HTTP
3128	9 316 334	Squid Cache
20	4 135 212	FTP Data
25	3 462 634	SMTP
6699	3 135 489	Napster
443	2 056 190	HTTPS
119	1 719 850	NNTP
6688	1 253 003	Napster
21	1 241 986	FTP Command
22	1 046 450	Secure Shell (SSH)
6700	948 392	Napster
6667	827 677	Internet Relay Chat (IRC)
53	498 220	DNS
5999	383 559	Cvsup
1023	371 944	

Table B.3: Top 15 TCP destination ports by total traffic size

TCP Port	Size (bytes)	Protocol
25	3 473 280 837	SMTP
6700	1 190 952 828	Napster
119	1 081 382 433	NNTP
6699	750 965 565	Napster
1438	689 478 255	
3353	687 270 052	
3754	681 246 592	
3324	679 982 305	
1617	653 253 097	
3873	619 836 939	
80	553 681 610	HTTP
3264	524 431 317	
3258	518 663 612	
3266	518 237 440	
3259	518 085 685	

Table B.4: Top 15 TCP destination ports by packet count

TCP Port	# of packets	Protocol
80	18 534 461	HTTP
3128	8 786 619	Squid cache
25	4 447 237	SMTP
6699	2 525 931	Napster
20	2 476 540	FTP Data
443	2 157 603	HTTPS
119	1 802 664	NNTP
6700	1 314 608	Napster
21	1 194 467	FTP Command
22	1 173 581	Secure Shell (SSH)
6688	892 002	Napster
6667	839 422	Internet Relay Chat (IRC)
53	543 662	DNS
5999	537 491	Cvsup
1438	479 099	

B.2 UDP Traffic

B.2.1 Source Ports

The three source ports which can positively identified are DNS, Squid ICP queries, and the Network Time Protocol (NTP). The relative order of these is the same on both the size and packet count rankings. The researcher after a preliminary examination, has been unable to identify the other ports present. Of particular concern is the traffic generated on port 1021. This could however have been due to an error when the traffic was originally captured. Analysis of the data was only carried out some time after the initial capture, and so the writer was unable to verify the source of these packets.

Table B.5: Top 15 UDP source ports by total traffic size

UDP Port	Size (bytes)	Protocol
1021	42 949 711 109	
824	38 657 925 265	
53	1 173 855 967	DNS
1036	799 070 274	
3130	420 855 484	Squid ICP
6901	45 907 695	
2000	42 393 378	
6970	24 244 555	
123	16 411 208	NTP
2267	14 776 262	
5000	13 829 884	
1024	11 070 824	
2949	10 199 997	
520	9 710 707	
213	7 134 721	

Table B.6: Top 15 UDP source ports by packet count

UDP Port	# of packets	Protocol
53	8 172 357	DNS
3130	5 073 881	Squid ICP
6901	985 024	
1036	848 141	
123	352 998	NTP
1024	300 946	
4923	225 828	
4196	165 426	
724	152 126	
5000	126 831	
814	122 656	
2268	119 708	
3695	118 412	
4193	114 646	
1304	114 225	

B.2.2 Destination Ports

The ordering of identifiable destination ports is similar to that of the source ports, with DNS being in first position in both Table B.7 and Table B.8. Squid ICP queries are second in the packet based ordering, and 14th in terms of size. The ports listed in positions two to 13 were unable to be attributed to any particular protocol by the researcher.

B.3 ICMP

The vast majority of the ICMP packets analysed, were of type 3 (destination unreachable). Most of the allowable ICMP type and code permutations [163, p71] were observed.

Table B.7: Top 15 UDP destination ports by total traffic size

UDP Port	Size (bytes)	Protocol
53	39 261 881 201	DNS
39106	4 294 969 207	
39108	4 294 968 366	
39109	4 294 968 089	
28700	4 294 967 952	
39105	4 294 967 812	
28701	4 294 967 803	
28698	4 294 967 781	
28699	4 294 967 539	
28697	4 294 967 448	
39107	4 294 967 288	
52302	799 120 806	
3130	443 432 283	Squid ICP
6970	86 060 230	
6901	55 860 193	

Table B.8: Top 15 UDP destination ports by packet count

UDP Port	# of packets	Protocol
53	9 273 293	DNS
3130	5 073 817	Squid ICP
6901	983 354	
52302	845 123	
123	357 304	NTP
1024	240 208	
111	175 544	Sun RPC
6970	159 978	
724	151 809	
5000	126 826	
814	122 515	
614	113 312	
1304	104 499	
4193	99 099	
427	95 170	

B.4 Ranges of port usage

An investigation was also carried out as to the number of ports used for both TCP and UDP out of the possible 65535 available ports. It was found that almost the entire port range was used. It should be noted, that the majority of traffic is carried by a small portion of the total ports.

Protocol	Total # of ports	Total % used
UDP		
Source	59951	91.4
Destination	59450	90.7
TCP		
Source	64830	98.9
Destination	64723	98.7

B.5 Type of Service Field

Only a small portion of the packets captured made use of the TOS field with anything other than the default setting of all zeros. Evidence was found of use of the Explicit Congestion Notification (see Chapter 2). Use of this field in the traditional sense was found to be more prevalent, than the newly proposed Differentiated Services implementation.

Appendix C

Visualisation Tools and Techniques

This appendix provides more information about software used to perform the visualisations.

C.1 Simple Graphing

The initial tool that was used to perform this visualisation was MRTG [121], and later *RRDTool* [119]. Both of these make use of a lossy data storage format [118]. A tool was developed by the researcher that did not suffer from this problem. The result was the bwplot system which is implemented using a combination of PHP and Perl, to provide a web based browser of historical traffic data. This system is also discussed in Case Study 1 (Section 3.1.2). The source code for this system is contained on the accompanying CD-ROM.

C.2 Layered Graphs

RRDTool [119] was used for this. The script used by the author to produce the layered images as depicted in Figure 3.11 is shown below:

```
#!/bin/sh
```

```
WWWDIR=’/www/data/bvi’
RRDDIR=’/usr/local/monitor/data/rrd’
/usr/local/bin/rrdtool graph $WWWDIR/protocol-in-d.gif\
-w 430 -c CANVAS#dddddd --alt-autoscale-max \
-t ’’Rhodes Internet Traffic - per Protocol -Incoming’’\
-v ’’Bytes per second’’ \
DEF:total=$RRDDIR/total-in.rrd:input:AVERAGE \
DEF:tcp20=$RRDDIR/protocol-in.rrd:tcp20:AVERAGE \
DEF:tcp25=$RRDDIR/protocol-in.rrd:tcp25:AVERAGE \
DEF:udp53=$RRDDIR/protocol-in.rrd:udp53:AVERAGE \
DEF:tcp80=$RRDDIR/protocol-in.rrd:tcp80:AVERAGE \
DEF:tcp119=$RRDDIR/protocol-in.rrd:tcp119:AVERAGE \
DEF:tcp3128=$RRDDIR/protocol-in.rrd:tcp3128:AVERAGE \
DEF:tcp8000=$RRDDIR/protocol-in.rrd:tcp8000:AVERAGE \
DEF:tcp8080=$RRDDIR/protocol-in.rrd:tcp8080:AVERAGE \
DEF:icmp=$RRDDIR/protocol-in.rrd:icmp:AVERAGE \
CDEF:www=tcp80,tcp8080,tcp8000,+,+ \
AREA:total#000000:’’Total’’ \
AREA:tcp3128#ff00ff:’’cache’’\
STACK:www#ff0000:’’www’’ \
STACK:tcp20#ffff00:’’ftp’’ \ STACK:tcp25#00ffff:’’smtp’’ \
STACK:udp53#ffffff:’’named’’ \
STACK:tcp119#00ff00:’’nntp’’\
STACK:icmp#0000ff:’’icmp’’ \
/usr/local/bin/rrdtool graph $WWWDIR/protocol-out-d.gif
```

Further details on the operation of *RRDTool* can be found in its accompanying documentation.

C.3 Colour Maps

This was one of the more involved pieces of software developed, and also served as a base for the work done on the 3D landscape visualisation. Data for a month

was extracted from the database that was used in Case Study 1 (Section 3.1.2). This was then reformatted, and processed in order to produce a grey scale image, the shading values of which were arrived at by mapping the data value for a particular point, divided by the maximum theoretical value (8192) and then multiplying by 255 and taking the absolute value. This gave a 256 colour pallette. In order to produce the false colour maps, a coloured palette was created. A utility developed by the writer read in the original grey scale image, and replaced the grey scale pallete with the coloured one, and output a new image. These scripts are included on the CD-ROM.

The contour map was produced using the *GnuPlot* mathematical modelling software [186]. The version used in the research (3.7.1), has a number of bugs, and is not ideally suited to this kind of function.

C.4 Three-Dimensional Landscapes

This proved the most challenging tool to design and develop. The input into the generation system was the processed datafile that the grey scale colourmap was produced from. This was then used as a heightfield input in a *PovRay* ray-tracing script. which produced a static view of the landscape with lighting. An enhancement on this was to split the image into four logical hightfields in order to produce the coloured strata as shown in Figure 3.14b. The problem with this is that the image took four times as long to render.

In order or produce the animations, a further *PovRay* script was developed which included dynamic camera, on a time based flight-path over the landscape. The flight-path was predetermined, and was not easy to change, which ruled out the possibility of ‘interactive’ user feedback. This script produced a number of images, which were then compiled into a single MPEG1 video file. On the researchers system (PIII 500MHz) it took approximately 40 minutes to produce a 30 second video clip in this manner. The advantage was that the entire process could be scripted form the database extraction to the generation of the video clip.

In an attempt to produce better quality models, and animations, 3D Studio Max

was used. The grey scale colour map was used to produce a heightfield mesh which was then rendered producing results such as that in Figure 3.15. In order to add a false colour mapping (as in Figure 3.16), the colour gradient map was used as a skin over the mesh produced by the grey scale image. The false colour map could not be used directly, as the software did not correctly interpret the pallette values for the mesh generation. This software was also used to generate the VRML model which together with some sample animations can be found on the accompanying CD-ROM.

Appendix D

Software Developed

This appendix details some of the design and implementation issues with regards to the two major software utilities that were developed during the course of this research. These tools can be found on the accompanying CD-ROM.

D.1 Squid Quota Management

The major design philosophy behind this tool was to maintain both simplicity and flexibility to aid for future enhancement. The decision to use a database as a backend, was influenced by this. The SQL functionality is used to perform much of the number manipulation and calculation, leaving the front end script to handle the actual updating of quotas and the squid configuration.

The tool is broken into two scripts. The first is responsible for the parsing of the Squid log files in order to calculate the per user totals for the previous day. The handling of what constitutes 'billable' traffic is done by this script. When it has processed the data, the user totals are inserted into the database, along with the date. A second script is responsible for the calculation and implementation of the quotas. A SQL query is passed to the database and a list of users and their totals over the previous 14 days is returned. This list is then parsed, and appropriate data written out to the quota control files. This script also has the option of expiring old user data from the database.

D.2 Dynamic firewalling

The dynamic firewalling utility is a Perl script which processes data captured off the network by the *ngrep* [150] utility. A number of configuration options are available for selection expressions and operational parameters. IP traffic can be selected by passing a 'pcap logic' statement to the *ngrep* program.

When a packet matches the pcap expression, and one of the regex rules which *ngrep* has been passed, it prints the matching data, together with information about the source and destination IP address:port pairs. This information is parsed by the Perl script in order to determine what action should be taken according to the configuration. If a firewall rule is required to be generated, appropriate parameters are passed to a subroutine, which inserts the rule into the system firewall table. A number of security features are also implemented to prevent against denial of service attacks.

Appendix E

CD-ROM Contents

The CD-ROM which accompanies this work comprises a number of sections which are elaborated below. Included are some of the data files used in the research and the raw results of some of the experiments carried out. The source code for the tools developed by the researcher are also available.

The contents of the Thesis, tools and videos directories are also avaliable online via the researcher's homepage: `http://rucus.ru.ac.za/~bvi/research/msc/files`.

Data/

This directory contains the data that was used in the research, and the experiments. Not included is the raw IP data captured as part of the bandwidth study, due to its size. The summary files that were produced, are however included.

bannerads/ A collection of nearly 4000 banner ad images which were collected, along with some analysis of their size and makeup.

bw_test_files/ A collection of packet traces of traffic generated by the researcher during tests on traffic shaping with *dummynet* and via Squid.

dummynet/ A collection of packet traces and graphs gathered while performing testing on the *dummynet* implementation in FreeBSD

packetdumps/ The summary results obtained from the processing of the *tcpdump* capture files used in the analysis of the Internet Traffic at Rhodes University. The original packet captures were in excess of 16 gigabytes, and could therefore not be included.

squid_delay/ Packet capture traces of the effect that Squid delay pools have on traffic.

Figs/

This directory contains the figures that are used in the body of this work, both in encapsulated level 2 Postscript (eps) format, as well as the originals that this was was created from.

Other/

Thesis/

This directory contains an electronic copy of the thesis in Adobe PDF and Postscipt format in addition to the BiBTEX bibliography file containing the references used in this work.

Tools/

bwclean/ The scripts for processing the data collected in Case Study 1, and used for the visualisations discussed in Chapter 3.

data/ Data input files files

bwclean4.pl The parser for the data files that produces summary output

feedbwdb.pl The script that takes the output produced above and inserts them into the database

bwplot/ This is the Web based front-end (bwplot) to the database system as described in Case Study 1.

bwprobe.php3 The PHP script which produces the front-end

bwplot.cgi The Perl script which generates the image from the database

gnuplot/ This directory contains example scripts and output from the GnuPlot utility

monitor/ These are scripts for manipulating data gathered by the monitor system discussed in Case Study 2.

povray/ This directory contains the scripts for generating the 3D landscapes

datafiles/ - The plain text data files form the database

data_gsimages/ - The grey scale images for the data files

images/ examples of the output

misc/ bits and pieces

dat2png.pl converts data file to grey scale PNG image

dat2xpm.pl converts data file to grey scale xpm

dat2xpmC.pl Produces an xpm image with colour palette

dopal.pl replaces the xpm palette with a given colour palette

extract5.pl extracts a months data from the database

flyby1.param Parameter file for the MPEG encoder

fullmonty.sh the master script which controls the generation of the animations

heightfield.pov the PovRay Scene file

rmon/ The script what was developed as an RMON clone

squidquota/ These are the two scripts which are used to implement the Squid quota system

calcdelay.pl responsible for the calculation of the appropriate delays for users, and the generation of the delay_pool ACL files

sushi.pl the parser for the squid log files which inputs the daily summary data into the database

runsquid.sh a control script for the running of a quota system

sting/ These are files related to the development of the Sting dynamic filtering tool

caps/ Various packet captures used in reverse engineering the protocols

datalog/ A very early implementation

docs/ Notes on the operation of the protocols and ideas for filtering

misc/ miscellaneous bits and pieces

sting/ The sting development directory (including several old versions)

sting-0.1.tar.gz the distribution tarball

Video/

A number of sample animations are contained in this directory .

images/ high quality static images taken from the animations

barry's traffic.max the 3D Studio Max model for producing the AVI animations

october99-set3-Iframeonly.mpg High quality MPEG animation for October 1999

october99-set3.mpg - smaller lower quality version of the above

september99-set2-Iframeonly.mpg - High quality MPEG animation for September 1999

september99-set2.mpg - smaller lower quality version of the above

fly1.mpg - a much slower version of the September animation

landscape.wrl - A VRML model of the landscape.

sept_colour_flybye(pass1).avi - the high quality colour gradient animation