

An Investigation Into End User Bandwidth Control Methodologies & A Comparison of Cisco & Linux Traffic Shaping On A LAN-To-WAN Gateway.

Dissertation for BSc Honours Degree (Network Management)

Author:	Gregory J. McMullin
Student ID:	17017449
Tutor:	Iain Hughes
Hand-in Date:	23rd April 2010
Project Type:	Investigation

Table of Contents.

<i>Acknowledgements.</i>	i
<i>Abstract.</i>	ii
1.0 Introduction.	1
1.1 Aims & Objectives.	1
1.2 Clarification Of Project Scope.	2
1.3 Elucidation Of Problem.	2
1.4 Why Is End User Bandwidth Control Necessary?	3
1.5 Research Methodology.	3
1.5.1 Relevant Published Research.	3
1.5.2 Experimentation Methodology.	4
2.0 Network Traffic Management & Control.	5
2.1 Network Traffic.	5
2.1.1 The Transmission Control Protocol (TCP).	5
2.1.2 The User Datagram Protocol (UDP).	7
2.1.3 The Internet Protocol (IP).	7
2.2 Quality of Service (QoS).	8
2.2.1 Integrated Services (IntServ).	8
2.2.2 Differentiated Services (DiffServ).	9
2.3 Traffic Shaping.	10
2.3.1 The Leaky Bucket Algorithm.	10
2.3.2 The Token Bucket Algorithm.	11
2.4 Traffic Policing.	11
2.5 Queuing and Scheduling.	12
2.5.1 The First In First Out (FIFO) System.	12
2.5.3 Priority Queuing (PQ).	13
2.5.4 Weighted Fair Queuing (WFQ).	13
2.5.5 Class Based Queuing (CBQ).	13
3.0 Investigation Into Traffic Shaping Methods.	13
3.1 Bandwidth Control & The Hierarchical Network Model.	13
3.1.1 At The Access Layer.	14
3.1.2 At The Distribution Layer.	14
3.1.3 At The Core Layer.	15
3.2 Real World Traffic Shaping Implementations.	15
3.2.1 GNU Linux.	15
3.2.2 Cisco IOS.	16
3.2.3 Mikrotik RouterOS.	16
3.2.4 Blue Coat Packetshaper.	16
3.2.5 Arbor Networks / Ellacoya.	16
3.2.6 Other Networking Hardware Manufacturers.	16
3.3 Chosen Methods & Reasoning.	17
4.0 Overview Of Tools & Methods.	17
4.1 Overview Of Cisco.	17
4.1.1 Cisco Shaping Methods.	17
4.1.2 Configuring Class Based Shaping.	18
4.1.3 Cisco Hardware To Be Used.	21
4.2 Overview of Linux.	21
4.2.1 GNU Linux.	21
4.2.2 Ubuntu Linux.	21

4.2.3 Linux Traffic Control (tc).	22
4.2.4 Class Based Queuing (CBQ) On A Linux Platform.	23
4.2.5 Hierarchical Token Bucket (HTB).	23
4.2.4 Configuring Linux For Networking.	24
4.2.5 Hardware Implemented.	26
4.3 What Will Be Measured?	26
4.3.1 TCP Throughput.	26
4.3.2 UDP Throughput.	26
4.3.3 Network Latency.	27
4.3.4 Jitter.	27
4.3.5 Packet Loss.	27
4.3.6 Out Of Order Arrivals.	27
4.3.8 Percentage CPU Utilisation.	27
4.3.9 Percentage Memory Utilisation.	28
4.4 Measurement Tools Considered.	28
4.4.1 Iperf.	28
4.4.2 Q-Check.	28
4.4.3 RUDE & CRUDE.	28
4.4.4 ping.	28
4.4.5 top.	29
4.4.6 show proc cpu / show proc memory.	29
4.4.7 Chosen Measurement Tools.	30
5.0 Experimental Investigation Methodology.	30
5.1 Objectives Of Investigation.	30
5.2 Implemented Network Topologies.	31
5.3 Traffic Shaping Scenarios.	32
5.4 Testing Methodology.	32
5.4.1 TCP Troughput.	33
5.4.2 UDP Troughput, Jitter & Packet Loss.	33
5.4.3 Network Latency.	34
5.4.4 Router CPU & Memory Utilisation.	34
5.5 The Importance Of Obtaining Baseline.	34
6.0 Experimental Results.	35
6.1 What Is Expected?	35
6.2 Analysis Of Recorded Data.	35
6.2.1 Analysis Of The Effect of Shaping On TCP Throughput.	35
6.2.2 Analysis Of The Effect of Shaping On UDP Throughput.	36
6.2.3 Analysis Of The Effect of Shaping On UDP Jitter.	37
6.2.4 Analysis Of The Effect of Shaping On UDP Packet Loss.	38
6.2.5 Analysis Of The Effect of Shaping On UDP Packet Delivery.	39
6.2.6 Analysis Of The Effect of Shaping On CPU & Memory Use.	40
6.2.7 Analysis Of The Effect of Shaping On Network Latency.	42
7.0 Conclusions & Recommendations.	42
7.1 Conclusions Drawn.	42
7.2 Recommendations To ISPs and Network Managers.	43
7.2.1 Is there A Clear Winner?	44
7.2.2 Ease Of Configuration.	44
7.2.3 Cost Of Implementation.	45
7.2.4 Cost To Upgrade.	45
7.2.5 Is It Supported?	45

7.2.6 Total Cost Of Ownership (TCO).	45
7.2.7 To Conclude.	46
8.0 Critical Evaluation.	46
8.1 Were The Objectives Of The Report Satisfied?	46
8.2 Evaluation Of Experimental Results.	47
8.3 Do Any Areas Require Further Research?	48
8.3.1 System Monitoring Methods.	48
8.3.2 Testing For Scalability.	48
8.3.3 Other Bandwidth Control Methods.	48
8.3.4 Further Investigation For ISPs.	49
8.4 Evaluation Of The Project.	49
8.5 Evaluation Of The Project Management.	49
References & Bibliography.	51
List Of Figures & Tables.	55
Appendices.	56
Appendix I – iPerf.	56
Appendix II – Q-Check.	57
Appendix III – iperf configurations.	60
Appendix IV – Linux Traffic Shaping Scripts.	61
Appendix V – Cisco Traffic Shaping Scripts.	63
Appendix VI – Linux tc shaping methods	67
Appendix VII – Raw Data From The Cisco Platform.	69
Results For Topology One Baseline.	69
Results For Scenario A.	70
Results For Scenario B.	70
Results For Scenario C.	71
Results For The Topology Two Baseline.	72
Results For Scenario D.	73
Results For Scenario E.	74
Appendix VIII – Raw Data From The Linux Platform.	76
Results For The Topology One Baseline.	76
Results For Scenario A.	77
Results For Scenario B.	77
Results For Scenario C.	78
Results For The Topology Two Baseline.	79
Results For Scenario D.	80
Results For Scenario E.	81
Appendix IX – Original Project Specification.	83
Appendix X – The Contents Of The Supplied CD.	84

Acknowledgements.

Firstly I would like to thank my project supervisor Iain Hughes who's knowledge and experience has steered me down the right path on more occasions than I care to mention. I would also like to thank all the other members of the Sheffield Hallam Faculty who's teaching over the past three years has helped with the production of this report.

I would also like to thank everyone at ask4™ for their support. In particular Chris Bridle for the loan of his Cisco router and his constant interest in my progress. Thank you also to Jonathan Burrows for giving me the go ahead to use some of the business' equipment when my own failed. Also thank you to Matt Eley, Jody Botham and Jamie Lavington for understanding that my dissertation has taken precedence over turning up to work in the past few weeks.

A big thank you goes out to Stephen Philip Riley for volunteering his time to proof read a report so far out of his own field of expertise. I hope this doesnt melt your brain. And also to Kier Halewood for proof reading via sattelite. Duncan Farnworth is worth a mention as his help with linux scripting was a real time saver. Cheers Dunc.

Finally thank you to my beautiful wife Christine who has put up with my pacing and muttering over the past 6 months. Thank you for proofreading and for having a better understanding of grammar than I do.

To anyone I have forgotten I apologise, rest assured I am eternally grateful.

Abstract.

As modern Internet Service Providers (ISPs) attempt to deliver higher internet connection speeds different methods of delivery are used. Some ISPs are providing network connectivity by connecting a multi tenanted building's Local Area Network (LAN) to their core network. This method of broadband delivery requires a method of end user bandwidth control.

This report investigates various methods of end user bandwidth control that may be attractive to such an ISP. It also compares the implementation of Cisco and Linux traffic shaping methodologies on a simple network topology. The suitability of these methods to provide a traffic shaping service for ISPs is evaluated and the effect of these shaping methods on the network traffic is also measured.

The results obtained provide proof of concept that both the Linux and Cisco platforms are capable of shaping traffic in the manner that an ISP would require. They also show that the network traffic is not effected detrimentally by the implementation of traffic shaping.

1.0 Introduction.

High speed broadband connectivity is rapidly becoming an essential utility (Cellan-Jones, 2009). Currently most customers receive broadband using existing copper telecommunication cables. However modern ISPs are not limited to providing connectivity over the legacy telephone networks. This traditional delivery method delivers acceptable downstream connection speeds but upload speeds are limited to a fraction of the download speed. Many new residential buildings are being built pre-wired with a Category-5e or Category-6 Local Area Network (LAN). This allows for an Internet Service Provider (ISP) to provide internet connectivity to each individual residential unit. If the ISP chooses to connect a high bandwidth line from the building to their core network then the end users can achieve download and upload connection speeds far in excess of most telecom cable connections. As the raw internet connection speed of any machine connected to the LAN will be comparable to the speed of the line connected to the building it is important that the ISP is able to implement some kind of end-user speed limiting method.

1.1 Aims & Objectives.

This report aims to investigate and discuss possible methods of end user connection speed limiting at the edge router of an Ethernet LAN. The two most feasible methods have been identified and a proof of concept network topology has been created and tested for these most viable implementations. The testing methodology ensures that the methods implemented do in fact shape the required traffic to the requisite limit. It also checks that the traffic is not negatively affected by the traffic shaping. Various attributes of the data including, but not limited to, throughput, latency and Round Trip Time (RTT) have been measured and compared. Real world attributes such as cost of implementation, ease of configuration and maintenance will be taken into account in order to provide a set of recommendations to any ISP who is considering setting up a traffic shaped multi host LAN (See section 6.4).

A more defined set of objectives are as follows:

- Research possible network traffic management technologies and methods.
- Provide an overview of traffic management technology and techniques.
- Research previous work on this topic and investigate past experimentation and implementation.
- Build upon this work and devise an experimental methodology.
- Identify possible methods of end user traffic management.
- Select the most viable methods for testing.
- Implement a lab based test network for chosen methods.
- Implement a testing methodology.
- Identify a number of scenarios to create on the test network.
- Investigate each chosen traffic management method's effect on network traffic in each of the chosen scenarios.
- Analyse and interpret the collected data.
- Investigate other attributes of the methods that have been tested.
- Compare the results of the investigation across the methods investigated.
- Produce a set of recommendations.
- Identify areas of the topic that require further investigation.

- Critically evaluate the investigation, experimentation, report and own performance.

1.2 Clarification Of Project Scope.

This project is only concerned with the end-user traffic control aspect of an ISP implementation at a Multi Tenant Building (MTB). It is expected that there will be a number of other areas that would require detailed research before a real world implementation could be realised. These areas are likely to include:

- End user security.
- Network security.
- End user authentication.
- Network address translation from the Internet to the private LAN.
- Contention ratios.
- Network Quality of Service (QoS).

This project does not intend to investigate any of the areas listed above although it is noted that they should be investigated in any future research.

1.3 Elucidation Of Problem.

Consider an ISP who has the contract to supply internet access to a block of flats with multiple tenants. If the building is pre wired for Ethernet connectivity the ISP can simply create a switched LAN within the building and connect this LAN to their own network backbone as shown in Fig.1.1. This report was inspired by a local ISP called ask4™ who use a Linux based system of their own design to control network traffic. The original goal of this report was to discover if Cisco systems hardware could adequately replace the Linux machines. It was soon realised that the scope of the project would need to be widened to investigating multiple implementations of traffic shaping and control.

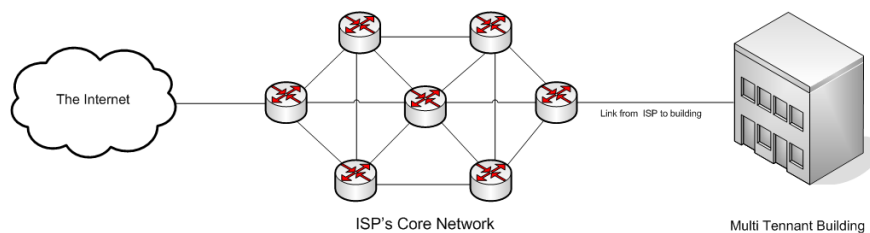


Fig.1.1

The investigated situation.

ask4™'s approach to controlling data is to implement traffic shaping at the building gateway router. This logical approach applies the traffic control as near as possible to the source of the traffic (see fig.1.2). This report aims to investigate alternative methods that an ISP such as ask4™ could use and will compare the two most feasible implementations to discover their effects on network traffic.

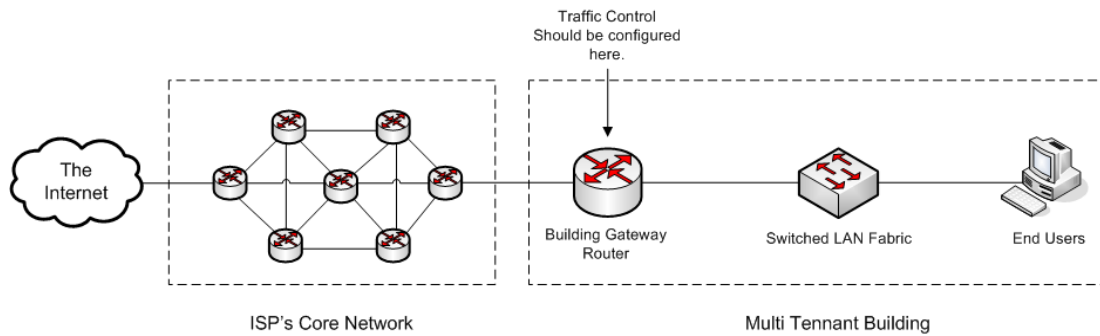


Fig.1.2

Controlling traffic at the building.

1.4 Why Is End User Bandwidth Control Necessary?

There are several reasons for an ISP to limit the customer's connection speed. Foremost being the ability to create a product with a pre-defined connection rate that the customer can purchase. For example the ISP may wish to sell a low speed starter package of 1Mbps for one price and a higher speed connection of 10Mbps for a higher price. It is also in the interest of the ISP to be able to control the data flowing from the customer to their network and then onto the internet as a single customer using bit-torrent could easily saturate a 100Mbps link. For most lower tier ISPs their largest cost is moving traffic off their network and onto the internet via transit providers. An effective end-user connection speed management solution would help data management in all these situations.

1.5 Research Methodology.

1.5.1 Relevant Published Research.

There have been a number of previous investigations into Traffic control. However, many of these investigations examine different areas such as Quality of Service (QoS) and controlling traffic once it has left the originating LAN. This report aims to investigate end user traffic shaping and as such many of the reports concerned with QoS are not specifically relevant.

One QoS thesis that does actually provide some relevance was published by Sheffield Hallam University in 2009 and is entitled "*An investigation into quality of service and its effect on internet application traffic*" (Bridle, 2009). The report itself investigates how network traffic is affected by a QoS implementation. The report looks at User Datagram Protocol (UDP) and Transport Control Protocol (TCP) throughput, Latency, Jitter and packet-loss and records how QoS affects each of these attributes of network traffic. The subject matter of Bridle (2009) may not be wholly relevant to this report but the testing methodology outlined within certainly is. His approach was to construct a small test network and then use Iperf to test TCP and UDP throughput across it with various QoS implementations. This approach gave excellent quantitative results and should be easily modified to work with traffic shaping.

In 2001 a technical report entitled "*Traffic shaping precision test*" (Ubik, 2001) was written by Sven Ubik and was published by CESNET. The report was concerned with how well a Cisco Systems series 7500 router performed basic traffic shaping on a fast Ethernet

interface. The report examined the effect of Cisco's traffic shaping method on various UDP streams generated on a test network and also measured the load on the routers processors as the traffic was being shaped. Ubik concluded that Cisco's traffic shaping implementation *"was working, in most cases, with precision sufficient for practical purposes"* (Ubik, 2001). It appears that Ubik (2001) has very similar aims to those of this report, however the experimentation that was performed has some differences to those planned herein. Some of the identified problems with the methodology outlined in Ubik (2001) are:

1. The experiment does not take into account TCP traffic.
2. Traffic is only tested flowing in one direction.
3. Only tests throughput no concession given to latency or jitter.

As well as building upon the experiment performed in Ubik (2001) this report also aims to address the flaws in its methodology and thus gain a wider understanding of how shaping methods affects network traffic.

Another thesis that provides some relevant material has the title *"A Linux based LAN to internet traffic shaping and IP accounting solution"* (White, 2006) and was published by Sheffield Hallam University. It investigates how to build and configure a Linux machine to act as a fully featured network router. A section of the report deals with using the traffic control (tc) program within Linux to limit the connection speeds of several end users. The report tests and proves that the Linux tc traffic shaping implementation works as expected, but does not really investigate the effect that the shaping has on the traffic nor does it investigate the load on the machines processor as shaping is taking place. White (2006) does not set out to investigate Linux shaped traffic shaping, the report has a broader scope and thus the traffic shaping section does not provide a large amount of detail. Some of the problems identified in the experimentation methodology of White (2006) are:

1. The tests are performed over the internet. - Not a controlled environment.
2. The end user clients that are shaped were virtual machines implemented on a single host operating system. This approach does not give a true reflection of network usage. It would be more realistic to use actual hosts on a switched network.
3. Throughput testing methodology is centred around downloading large files from a web server. This method does not provide adequate control. Results could be skewed by external factors.
4. The throughput tests were limited to TCP/IP traffic. UDP traffic should also be examined.

Despite these drawbacks White (2006) does give a balanced and informed overview of the Linux traffic shaping implementation. This report aims to build on the groundwork put in place by White (2006) and improve the testing methodology. White (2006) also determines that traffic shaping is an implementation of QoS and as such QoS should be investigated and relevant features of QoS may be considered for use in the experimentation section.

1.5.2 Experimentation Methodology.

The reports discussed in section 1.5.1 performed experimentation in order to fully understand what is happening to the network traffic. This report hopes to implement a

testing methodology based on Bridle (2009) that will build upon the methodologies implemented in Ubik (2001) and White (2009).

One of the main aims of this report is to investigate the affect on network traffic of a traffic shaping implementation. Ubik (2001) and White (2006) have already partially completed this objective. The experimentation methodology that will be delivered in section 5 will use a topology similar to that laid out by Bridle (2009), that of an "end user" PC and a "server" PC that connect though a router. TCP and UDP bandwidth tests will then be run in both directions across the network with the ultimate goal of determining average values for Latency, Throughput and Jitter whilst the traffic is being shaped at different levels.

A method of measuring processor and memory usage for each traffic control method will also be devised. This will hopefully provide an insight into how hard the systems are working. From this information it will be possible to extrapolate how well the implementations will scale.

2.0 Network Traffic Management & Control.

2.1 Network Traffic.

This section aims to give a background overview of the networking protocols that will be monitored in the experimentation section of this report. This section does not aim to give a comprehensive overview. If you wish to learn more please refer to *Computer Networks* by Andrew S. Tanenbaum (2003) which goes into far greater detail.

2.1.1 The Transmission Control Protocol (TCP).

TCP is described as "*the work horse of the internet*" (Tanenbaum, 2003). By design it provides a reliable end to end connection for traffic to pass through, even though the network it is working upon is an inherently unreliable medium. This leads to TCP being described as a connection orientated protocol and means that before any data can be transmitted, a TCP connection has to be created between the sender and the receiver.

TCP is also a reliable protocol. Reliable in this instance means that any data transmitted over TCP will be delivered accurately, in the correct order and without errors. Due to the nature of switched inter-networks packets sent form a source to a destination can arrive out of sequence. TCP deals with this by assigning a sequence number to each packet. These sequence numbers ensure that the destination can accurately re-order any packets that arrive out of sequence. The procedure for this is as follows:

1. Source transmits packet to destination over TCP connection.
2. Destination receives packet and checks for errors. If the packet is not corrupted then the destination sends an acknowledgement (ACK) packet back to the source.
3. If the source does not receive the ACK packet after a defined period of time it assumes that the packet has been lost or was corrupted and so re-sends the packet.

4. When the packets have been received by the destination TCP checks the sequence numbers and re-orders the packets if necessary.

This reliability means that TCP has an advantage over UDP when it comes to delivering data over inter-networks. TCP also has a number of features that are directly relevant to traffic shaping such as flow control and congestion control.

2.1.1.1 TCP Congestion Control.

As TCP is a connection orientated protocol there is a line of communication in place between the source and the destination. The ACK packets sent by the destination inform the source that the packet has arrived and also how long it took to get there. If the source does not receive these ACK packets after transmitting it assumes that there is congestion on the link and thus "backs off" sending data. Essentially TCP allows the transmission speed of the data to be reduced. Traffic shaping implementations can take advantage of this by deliberately dropping packets and thus leading the source machine to drop its transmission rate. How this is achieved will be discussed in a later section of this report.

2.1.1.2 TCP Flow Control.

Another attribute of the TCP is its ability to perform flow control. Consider a situation where the sender of the data is sending at a rate which is in excess of what the destination machine can process. Eventually the receiving machine will begin to drop packets because it can not handle the arriving data frames. There are two ways in which flow control can be implemented:

1. **Feedback based flow control** - In which the receiving machine sends information back to the sending machine, giving it permission to continue sending. Or providing information about how it is coping. (Tanenbaum, 2003)
2. **Rate based flow control** - In which the sending protocol has a mechanism to limit the rate at which the sender can send data. This needs no feedback from the receiver. (Tanenbaum, 2003)

Both these methods will lead to the data rate of the sending machine to be reduced. Either to the rate at which the destination machine can service packets or to an arbitrary rate set by the sending protocol. TCP uses what is known as a "Sliding Window" to control flow, without going into too much detail this allows the receiving machine to specify how much data it can receive.

2.1.1.3 TCP Congestion Avoidance Mechanisms.

TCP is able to reduce its speed of transmission when congestion is detected on the network. This is achieved by devices employing congestion avoidance algorithms on their interfaces that drop packets and thus cause TCP to retransmit at a slower speed using the sliding window flow control mentioned earlier. The two main congestion avoidance algorithms are:

- **Random Early Detection (RED)** - Random early detection works by dropping packets from any of the outgoing traffic flows at random as the interfaces buffer reaches its fill limit. This means that all TCP flows passing through the interface will reduce in flow speed. RED can be said to be a fair

method as it maintains equity in the traffic that is discarded. (Ferguson & Huston, 1998)

- **Weighted Random Early Detection (WRED)** - WRED attempts to build some unfairness into the dropping of packets. Unfairness is important if QoS is to take place on the flows as it allows DiffServ to take place (see section 2.2.2). WRED uses RED to discard packets introduces the ability to assign each flow a "weight" so that more (or less) packets can be dropped from a particular flow. Thus by "classifying" a flow with a particular weight a transmission rate can be set for it.

2.1.2 The User Datagram Protocol (UDP).

UDP is a connectionless protocol which means it allows data to be sent without a connection having to be established first. UDP does not check to see if the destination is ready, it simply transmits the data. Data is encapsulated in datagrams not bit streams like TCP. UDP is considered to be an unreliable protocol, which means any data sent over UDP is not guaranteed delivery, instead it offers a "best effort" delivery. The UDP datagrams may be dropped by any of the intermediate networking devices if they are congested. No mechanism exists for the sending machine to be notified of any such drops. Thus there will be no re transmission. Where TCP can re-order out of sequence packets, UDP can not re order out of sequence datagrams, this is because the UDP header (See fig 2.1) does not have a sequence number field and therefore the destination will not know in what order they are supposed to be.

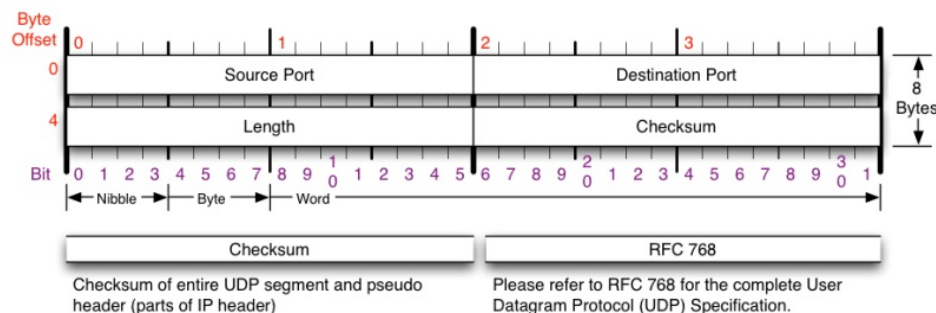


Fig.2.1

The UDP Header (Lyon, 2009)

UDP has uses in a number of applications such as client-server interactions and multimedia. (Tanenbaum, 2003). Some examples of UDP use are:

- Voice over IP (VOIP).
- Remote Desktop (RDP).
- Streaming Audio and Video.

It is important that any traffic shaping method implemented is able to limit the maximum transmission speed of both TCP and UDP.

2.1.3 The Internet Protocol (IP).

TCP and UDP are transport protocols which means they are responsible for encapsulating data into a medium which can be transmitted onto the internet. It is IP's responsibility to

route this data across the networked fabric of the internet. It is able to do this as all hosts on an IP network are assigned an IP address. IP is another unreliable protocol that only promises a "best effort" delivery. In this respect it is similar to UDP.

2.2 Quality of Service (QoS).

End-user traffic shaping can be considered to be a method of QoS. Traditionally there have been two ways of achieving QoS. Management and control and Over-provisioning (Tanenbaum, 2003). From an ISP point of view over-provisioning is something to avoid, primarily because it is not financially economical. Instead ISPs must look to manage and control the connection speeds of their end users. To achieve this they may use QoS traffic shaping techniques. The two main methods of QoS delivery standardised by the IETF are Integrated and Differentiated Services.

2.2.1 Integrated Services (IntServ).

IntServ is a QoS architecture standardised by the Internet Engineering Task Force (IETF) which documents how QoS is to be implemented on inter-networks. Its main purpose was to provide the ability to *"support real-time as well as the current non-real-time service of IP."* (IETF, 1994). IntServ was created so that streaming multimedia would not overwhelm available internet bandwidth. IntServ is implemented on every router along the path the data needs to flow. Resources can be reserved in advance so that traffic is able to flow efficiently. The protocol that makes IntServ possible is known as The Resource ReSerVation Protocol or RSVP.

2.2.1.1 The Resource ReSerVation Protocol (RSVP).

RSVP provides *"provides receiver-initiated setup of resource reservations for multicast or unicast data flows, with good scaling and robustness properties."* (IETF, 1997). RSVP does this by communicating with devices along the path the traffic needs to take and reserving the bandwidth and CPU time needed to service the packets at the desired rate. Once every device along the path has reserved the necessary bandwidth the source device can begin transmitting. (Cisco, 2010) This method ensures that the packets travelling from source to destination have enough resources at each step of the journey thus they are dealt with quickly and efficiently at each hop. Fig.2.2 illustrates RSVP in action.

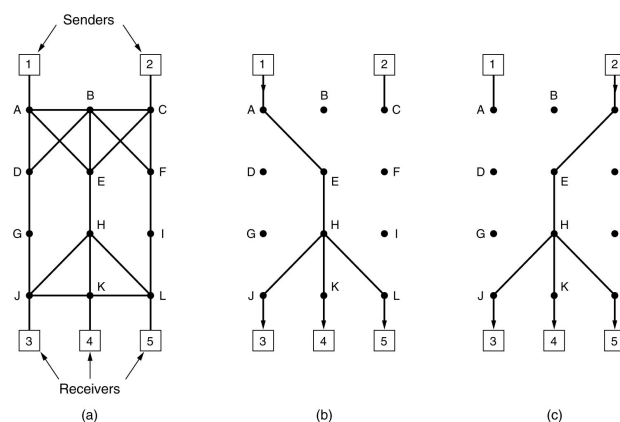


Fig.2.2

RSVP. (a) Host 3 requests a channel to host 1. (b) Host 3 then requests a second channel to host 2. (c) Host 5 requests a channel to host 1. (Tanenbaum, 2003)

This flow reservation method is a very good way of ensuring that QoS is delivered to the flows that need it. However when scaled to internet sized networks there is the possibility that there will be thousands if not millions of flows passing through an individual router. As each router keeps a per-flow state in its memory it makes routers vulnerable to memory overflow. This could lead to data collisions, packet loss and eventually crash the device (Tanenbaum, 2003). Hence, it is said that IntServ does not scale to larger networks well.

2.2.2 Differentiated Services (DiffServ).

As discussed in section 2.2.1.1 IntServ has problems when scaled to internet sized networks. This is where the second IETF standardised QoS architecture, Differentiated Services or DiffServ comes in.

DiffServ is a simpler approach to QoS which is implemented within the router itself without needing to set up the whole traffic path in advance and without having to reserve flow bandwidth (Tanenbaum, 2003). DiffServ is often referred to as a class-based standard meaning that instead of using traffic flows it uses traffic classes i.e. *"Packets are classified and marked to receive a particular per-hop forwarding behaviour on nodes along their path"* (IETF, 1998).

Classification means that specific rules can be set up on each intermediate router. This paired with Diffserv's ability to mark packets in the "Type of Service" field of the packet header means that different packets can be given different levels of service at different routers. Some packets, for example streaming video, may be given priority and will be forwarded as soon as they are received. Other packets may be constrained by a shaper that will reduce their flow rate. This corresponds well to the situation that is being investigated by this project.

Traffic can be classified at many layers of the OSI networking model. The following criteria may be used to identify a traffic flow and apply it to a queue or shaping mechanism:

- OSI Level 1 - The Physical Interface, sub interface, PVC or Port.
- OSI Layer 2 - MAC address, Class of service (CoS) bits, VLAN identification.
- OSI Layer 3 - Source or destination IP Address, DiffServ code point (DSCP), IP Precedence.
- OSI Layer 4 - TCP or UDP ports.
- OSI Layer 7 - Application signatures or URLs in packet headers or payloads. (Szigeti & Hattingh, 2005)

In the investigated situation the traffic will be classified either by the Source/Destination IP Address or the source/destination MAC address. The forwarding behaviour will be done at the first hop i.e. the LAN edge router. Customers traffic speed will be shaped as it enters or leaves the building LAN. This class based approach means that traffic can be sent and received at different rates. In our example customers requiring a 10Mbps download speed would be placed in a different class to customers who only require a 1Mbps download speed. When these packets arrive at the LAN-edge router DiffServ will take the responsibility of shaping each class of packet to their corresponding agreed data rate.

2.3 Traffic Shaping.

Traffic shaping is concerned with controlling the rate of data being transmitted by a device. The IETF definition of a Traffic Shaper is:

"Shapers delay some or all of the packets in a traffic stream in order to bring the stream into compliance with a traffic profile. A shaper usually has a finite-size buffer, and packets may be discarded if there is not sufficient buffer space to hold the delayed packets."
(IETF, 1998)

ISPs could choose to use a traffic shaping mechanism to limit their end users' internet connection speeds. Traffic shaping works at the server side of the client server relationship therefore it can be implemented at the LAN-edge router for incoming and outgoing user traffic. Shaping is a possible method for setting up a Service Level Agreement (SLA) between an ISP and a customer. Two methods of traffic shaping, the "leaky bucket" and "token bucket" algorithms will be discussed in the following sections.

2.3.1 The Leaky Bucket Algorithm.

Consider a metal bucket with a small hole drilled in the bottom. As the bucket is filled water will begin to leak from the small hole in the bottom. No matter how fast the bucket is filled the rate that the water leaks out will always be constant. This theory can also be applied to packets. Fig.2.3 shows the leaky bucket in operation.

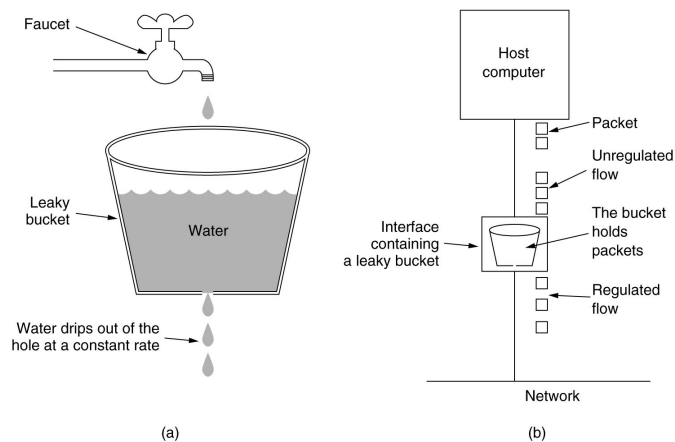


Fig.2.3

The Leaky Bucket Algorithm

(a) A leaky bucket with water. (b) A leaky Bucket with Packets (Tanenbaum, 2003)

The leaky bucket can be set up on a network interface, the bucket can be thought of as a finite packet queue. When a packet arrives at the interface it is placed into the queue. After a period of time, depending on the set size of the queue, it will be serviced by the interface and passed onto its destination. This action of the packet entering the queue and waiting around smoothes the outgoing flow. If the packet arrives at the queue and the queue is full then the packet is simply dropped. The leaky bucket algorithm implements a "single server queuing system with a constant service time" (Tanenbaum, 2003). This is not the most efficient way of managing network traffic as there is no way to increase the traffic speed mid flow,

the flow rate for the leaky bucket is static. This is where the token bucket algorithm comes in.

2.3.2 The Token Bucket Algorithm.

The token bucket algorithm is more advanced than the leaky bucket algorithm. Where the leaky bucket algorithm can only shape traffic to a particular rate the token bucket algorithm allows traffic to increase in speed for small time periods. The token bucket algorithm is able to do this as it is built around the leaky bucket model but with added tokens. In order for a packet to pass through the leaky bucket it must capture a token. Once a token has been taken by a packet it is destroyed meaning that for more packets to pass through more tokens must be generated. The traffic speed is now dependant on the amount of tokens in the bucket. If more tokens are created the transmission rate of the queue will increase, likewise if tokens are created at a slower rate then the transmission rate will be reduced. If a host does not transmit for a period of time then the bucket can fill up to the maximum amount of tokens, this allows the host to burst its transmission rate when it eventually does start transmitting. In addition, the leaky bucket algorithm never discards packets instead when the bucket fills up it begins to discard transmission tokens. Fig.2.4 shows the token bucket algorithm in action.

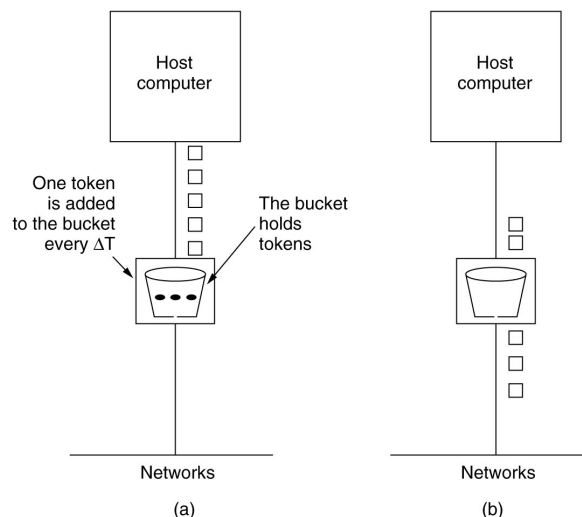


Fig.2.4

The token bucket algorithm.

(a) Before. (b) After. (Tanenbaum, 2003)

2.4 Traffic Policing.

Traffic Policing is usually employed to make sure that traffic that is transmitted conforms to an SLA. ISPs can use traffic policing to ensure that a customer does not send more data onto a link than they are allowed to do. Traffic policing in combination with traffic shaping ensures that traffic transmission rate never exceeds a specific pre-agreed limit. Fig.2.5 shows how policing and shaping affect the flow of traffic.

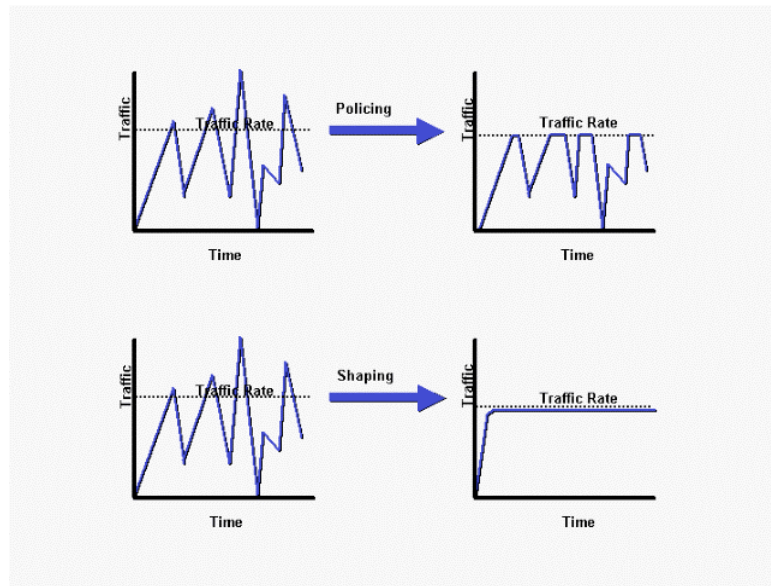


Fig.2.5

Comparing shaping and policing (Cisco, 2005)

As demonstrated policing simply drops any traffic that exceeds a specific rate, whereas shaping smoothes the entire flow out. It is for this reason that shaping, and not policing will be considered for testing in this report.

2.5 Queuing and Scheduling.

"*Queuing is the logic or ordering packets in linked output buffers [and] Scheduling is the process of deciding which packet to transmit next*" (Szigeti & Hattingh, 2005). Traffic shaping implementations rely on queuing and scheduling systems to be able to alter the speed of the traffic flows. Fig.2.6 shows an example of a simple queuing and scheduling system.

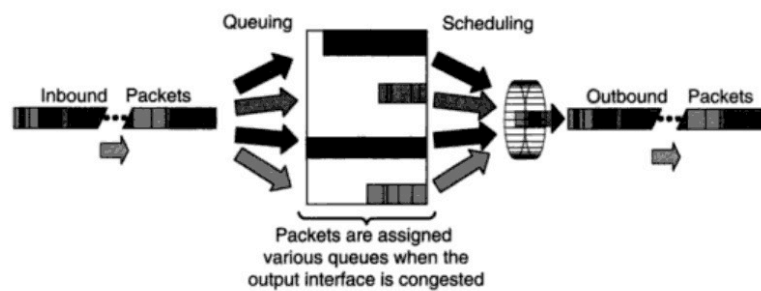


Fig.2.6

A Generic Queuing Example (Szigeti & Hattingh, 2005)

More information about queuing, scheduling and QoS as whole can be found in the Book End-to-end QoS Network Design: Quality of service in LANs, WANs, and VPNs by Tim Szigeti and Christina Hattingh (2005).

2.5.1 The First In First Out (FIFO) System.

The simplest queuing and scheduling system is a FIFO queue. The first packet that joins the queue is the first one to be transmitted. This works well when the number of packets in the queue is less than the maximum possible. However when the queue is full any subsequent packets are discarded. FIFO queuing works well when the number of packets in the queue

is less than the maximum as all packets are serviced and the flow is smoothed. However a FIFO system has only one queue and thus when the queue is full any subsequent packets are dropped.

2.5.3 Priority Queuing (PQ).

PQ consists of four queues defined as high, medium, normal and low priority. The high priority queue is serviced first and once it has been emptied the medium queue is serviced and so on until all the queues have been emptied. The main problem with PQ is that if there is a constant high priority flow no other queue will ever be serviced. Scheduling in PQ is done on the strict priority method. (Szigeti & Hattingh, 2005)

2.5.4 Weighted Fair Queuing (WFQ).

WFQ generates a queue per traffic flow on the interface. It then divides the bandwidth of the interface between the number of flows fairly. A weight is then added to each flow which allows each flow to be reduced or increased depending on their calculated priority. (Szigeti & Hattingh, 2005) Priority is calculated by flow volume with low volume flows given priority over high volume. (Ferguson & Huston, 1998)

2.5.5 Class Based Queuing (CBQ).

CBQ was designed to resolve the resource starvation problem of PQ. Up to 16 queues can be defined in class form the 4 in PQ. Each queue can be assigned a priority. Each queue is served by the server in a round robin fashion and also the amount of traffic that is drained from each queue on each pass can be defined. These improvements to PQ provides increased fairness. (Ferguson & Huston, 1998) However due to this implementation CBQ does not offer strict priority to real time traffic flows that PQ does. (Szigeti & Hattingh, 2005)

3.0 Investigation Into Traffic Shaping Methods.

This section will investigate and discuss several feasible methods for end user bandwidth control on an Ethernet LAN.

3.1 Bandwidth Control & The Hierarchical Network Model.

The hierarchical network design model is a model implemented by Cisco Systems to help the design and construction of large scale inter-networks. It uses a three tiered hierarchy:

- The backbone (core) layer that provides optimal transport between site.
- The distribution layer that provides policy-based connectivity
- The local-access layer that provides workgroup/user access to the network (Cisco, 2010b)

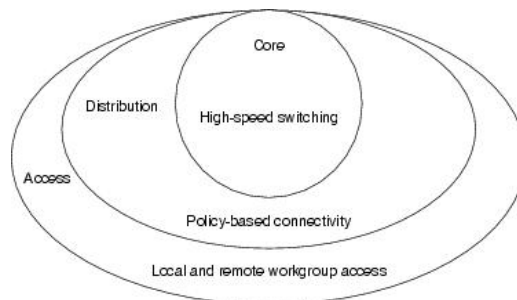


Fig.3.1

The Hierarchical Internetwork Design Model. (Cisco, 2010b)

Bandwidth management can take place at any or all of the three levels. The next sections will discuss how.

3.1.1 At The Access Layer.

It is theoretically possible to limit the end users connection speed on their device itself. Either by installing software that shapes their incoming and outgoing bandwidth or by physically configuring their hardware to limit connection speed. However these methods are far from ideal. The ISP would need access to every device that connects to its network in order to configure the hardware or install the software. What is to say that the customer would not simply reverse the changes? There is also an ethical consideration, the device belongs to the customer not the ISP so does the ISP have any right to alter its workings? Probably not. This method is flawed on a number of levels and should not be considered for testing or implementation.

The next device in the network chain that bandwidth management could be implemented is the access switch. The switch that the customer connects to is the property of the ISP, so the ISP has full control over its configuration. It should be relatively simple to identify which switch-port the customer is attached to and to configure it to limit the connection speed. In Cisco switches policing can be applied to both incoming and outgoing traffic on a switch port (Cisco, 2005) meaning that the incoming and outgoing traffic can be limited to an arbitrary rate. This sounds like the perfect solution to our problem. However it does have some drawbacks. In a large MTB there may be hundreds of customers, and over the whole network there may well be thousands of customers. Each individual switch port would have to be configured, this could be a very time consuming process. Also if a customer wanted to upgrade or downgrade their connection speed the switch configuration would need to be altered. For these reasons it is obvious that this method does not scale well and thus should not be considered as a method to test.

The access layer is not the level where bandwidth management should be implemented. Implementation of policies like bandwidth control and QoS are best implemented at the distribution layer. (Cisco, 2010b)

3.1.2 At The Distribution Layer.

If we follow the path of the data onto the next device we will find ourselves at the distribution layer. In most internetworked LANs the distribution layer begins at the LAN-edge router. According to the Cisco hierarchical network model this is where policy decisions need to be made. So how could this be done?

The LAN-edge router will have at least two interfaces, one facing the local LAN and one facing the rest of the ISPs core network. A traffic shaping method can be implemented on each of these interfaces. If a classful method is implemented then, multiple rates can be defined. The end users can then be added to one class on each interface either by use of their IP address or MAC address. As long as the shaping method is configured correctly the end user's traffic will be shaped. Thus their connection speed will be limited. The beauty of this method is that both the incoming and outgoing traffic will be modified with traffic shaping and as you can see from Fig.2.5 shaping is a far smoother method of controlling flow than policing.

This is the method that is implemented and tested in this report.

3.1.3 At The Core Layer.

The core layer is in used for high speed data delivery, service policy implementations should not be made at this level. Therefore no investigation will be made into core layer end-user traffic management methods.

3.2 Real World Traffic Shaping Implementations.

The following sections cover possible traffic shaping implementations that are available on the open market that an ISP may wish to use.

3.2.1 GNU Linux.

GNU Linux is an open source operating system that has the ability to perform traffic shaping .The Linux kernel includes a number of programs that, when configured correctly, can turn a Linux machine into a powerful, fully featured router. The program that performs the traffic shaping is called "traffic control" but is widely referred to as "tc".

The Linux Advanced Routing and Traffic control How-To pages at lartc.org state a that the following tasks are possible using a machine configured for Linux routing:

- *"Throttle bandwidth for certain computers*
- *Throttle bandwidth TO certain computers*
- *Help you to fairly share your bandwidth*
- *Protect your network from DoS attacks*
- *Protect the Internet from your customers*
- *Multiplex several servers as one, for load balancing or enhanced availability*
- *Restrict access to your computers*
- *Limit access of your users to other hosts*
- *Do routing based on user id, MAC address, source IP address, port, type of service, time of day or content."* (Hubert, 2004)

These abilities mean that Linux should be able to be configured to satisfy the requirements of this report.

3.2.2 Cisco IOS.

Cisco IOS is the proprietary operating system installed on Cisco Systems networking hardware. The IOS software is extremely extensible and can be configured to perform many tasks. These include a fully featured QoS solution that allows for both policing and shaping of traffic. A Cisco router running IOS would be able to be configured to act as a traffic shaper and thus would satisfy the requirements of this report.

Cisco are considered to be the market leaders in networking devices.

3.2.3 Mikrotik RouterOS.

Mikrotik RouterOS is a commercial product based on the Linux 2.6 kernel. It can be installed on a PC and that PC will then be able to act as a fully featured router with features including:

"routing, firewall, bandwidth management, wireless access point, backhaul link, hotspot gateway, VPN server and more." (Mikrotik, not dated)

Its QoS features, specifically its bandwidth control mechanisms, seem ideally suited to the task at hand.

3.2.4 Blue Coat Packetshaper.

Previously known as Packeteer the Blue Coat Packetshaper is hardware appliance that performs traffic monitoring, shaping and compression (Bluecoat, 2010). It is an enterprise level device that provides the ability to perform traffic management for an entire network. Blue Coat produce a range of packetshaper machines. Their entry level device is the Packetshaper 900 which can handle up to 5000 TCP streams at a time and has the ability to handle up to 256 traffic classes. (Blue Coat, 2009)

3.2.5 Arbor Networks / Ellacoya.

Previously known as Ellacoya, Arbor Networks produce carrier class broadband optimisation devices. Their e30 and e100 machines have the ability to perform deep packet inspection (DPI). This allows ISPs to classify traffic up to the application layer. This means that ISPs can use these devices to manage high traffic applications such as peer to peer file sharing. These devices can also be used to manage subscriber bandwidth like so:

"You can proactively manage bandwidth demand and prioritize it according to how and when it's used, and by whom. As a result, you can fully optimize network performance in off-peak and peak periods." (Arbor Networks, 2009)

These devices provide true ISP level management capabilities, they are designed to work at the distribution layer of the ISPs network and provide traffic management solutions for thousands of subscribers.

3.2.6 Other Networking Hardware Manufacturers.

There are other manufacturers of networking equipment, such as Hewlett Packard and Netgear who have their own internal operating systems on their routers. These manufacturers also provide the ability to perform QoS and thus bandwidth management.

3.3 Chosen Methods & Reasoning.

This report aims to compare two of the methods outlined above. Unfortunately it is not possible to obtain Bluecoat Packetshaper or Arbor Networks Ellacoya appliances for testing. These carrier class devices retail at multiple thousands of Pounds so obtaining examples to investigate is not feasible at this time.

As the Mikrotik Router OS solution is based on the Linux kernel it seems counter productive to compare this to another Linux distribution. Under further research it seems that Mikrotik Router OS is simply a more user friendly front end for the Linux networking suite. For these reasons it is decided not to pursue testing of the Mikrotik Router OS. Any further investigations into traffic shaping methods may wish to revisit Mikrotik Router OS.

Cisco Systems are considered the market leader in router OS design and implementation, it stands to reason that their implementation should be chosen ahead of other manufacturers. So this leaves Cisco and Linux as the two traffic shaping methods that will be tested and evaluated. There have already been investigations into these two methods of traffic shaping (see section 1.5.1) so any further research will be these in mind.

4.0 Overview Of Tools & Methods.

This section will provide an overview of the tools and methods used to perform the network testing detailed in Section 5.

4.1 Overview Of Cisco.

Cisco IOS is the operating system installed on Cisco Systems networking hardware. It can be configured by the use of its Command Line Interface (CLI) in which a series of commands can be imputed to program the router. These commands allow the device to be configured to perform multiple tasks. IOS supports a number of methods of traffic shaping. These are discussed in the following section.

4.1.1 Cisco Shaping Methods.

4.1.1.1 Generic Traffic Shaping (GTS).

GTS is the default method of traffic shaping found on Cisco routers. It uses a token bucket to smooth outgoing traffic flows to a particular bit-rate. GTS is configured on a per interface basis and can use Access Control Lists (ACLs) to classify what traffic should be shaped. Weighted fair queuing (WFQ) is the only queuing method supported in GTS (Cisco, 2010c). Fig.4.1 shows GTS in action.

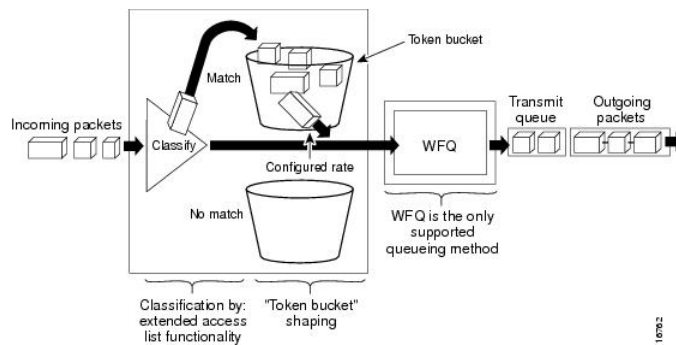


Fig.4.1

How GTS works (Cisco, 2010c)

4.1.1.2 Class Based Shaping (CBS).

Class based shaping allows GTS to be applied on particular classes of traffic. GTS itself can only be configured by the use of ACLs but as CBS can use traffic classes this provides increased flexibility in configuration. CBS can be configured to use CBWFQ as its queuing mechanism. Up to 64 classes can be configured and specific bandwidth allocations can be assigned to each class. (Cisco, 2010c)

CBS also allows for an average or peak traffic rate to be configured for shaping. This means that bandwidth can be allocated more efficiently as it allows for more data to be sent if the bandwidth is available. (Cisco, 2010c)

4.1.1.3 Distributed Traffic Shaping (DTS) & Frame Relay Traffic Shaping (FRTS).

DTS and FRTS are the final two methods of traffic shaping that can be implemented on Cisco IOS these methods are not being considered for implementation in this investigation so will not be discussed. However any further study into traffic shaping may wish to investigate these methods further.

4.1.1.4 Which Method Will Be Implemented?

A choice has to be made between using GTS and CBS as the method to be investigated. In the scenario that is being tested shaping needs to be achieved on a per IP or per MAC address basis. This can be achieved by both GTS and CBS. GTS can perform this only by the use of ACLs and CBS by the use of traffic classes. In a real world scenario it is more advantageous to implement a CBS system as this would allow for greater flexibility as traffic can be classed on a large number of attributes. For this reason CBS will be chosen as the Cisco shaping method to be tested.

4.1.2 Configuring Class Based Shaping.

QoS and thus traffic shaping is performed on Cisco IOS in a modular fashion, meaning that there are several layers of configuration that must be accomplished before completion. These layers are as follows:

1. Creation of a traffic class using the `class-map` CLI command.
2. Attaching the traffic class to a traffic policy and a corresponding shaping method using the `policy-map` CLI command.
3. Enforcement of the traffic policy on the required interface using the `service-policy` CLI command. (Cisco, 2010d)

4.1.2.1 The `class-map` Command.

To configure a traffic class the `class-map` command is used. It has two lines. The first names the class the second defines what traffic the class matches. The first command has the following syntax:

```
Router(config)#class-map [match-all | match-any] class-name
```

- `class-map` is the IOS command used to create the traffic class
- `[match-all | match-any]` defines which of the following match criteria should be followed. `match-all` means all the match commands have to be followed. Whereas `match-any` means the traffic will be considered part of the class if any of the match commands are met.
- `class-name` is the name given to the traffic class.

After the traffic class has been created and named the match command(s) are configured. They have the syntax:

```
Router(config-cmap)#match <match-identifier>
```

- `match` is the IOS command that defines what traffic will be added to the traffic class
- The `<match-identifier>` can be a number of things. Traffic can be matched on an ACL, the destination MAC or IP and many more.

As the configuration that will be implemented in this report will use ACLs to match traffic to certain hosts, the match command will be as follows:

```
Router(config-cmap)#match access-group {access-group | name access-group-name}
```

- `access-group` defines that an access control list will be used to match traffic.
- `access-group` will be the access control list number.
- or if the access list is a named access list the `name access-group-name` modifier will be used. (Cisco, 2010e)

4.1.2.2 Access Control Lists (ACLs).

As ACLs are to be used to classify the traffic it is important to examine the configuration syntax of ACLs in Cisco IOS. In this scenario extended ACLs are used as the source and destination addresses can be specified.

```
Router(config)#access-list [access-list-number | access-list-name] [permit | deny] <protocol> <source> <destination>
```

- `access-list` is the IOS command that creates an access list.
- `[access-list-number | access-list-name]` defines the number or name of the access list.
- `[permit | deny]` states whether the specific traffic should be permitted or denied.
- `<protocol>` defines the protocol of the traffic IP, TCP or UDP.
- `<source>` defines the source IP address of the traffic.
- `<destination>` defines the destination IP address of the traffic. (Cisco, 2010f)

An example of the ACL format that will be used in this report follows:

```
Router(config)#access-list 100 permit ip 10.0.0.2 0.0.0.0 any
```

Where 0.0.0.0 is a wildcard mask that defines traffic will be controlled to a single host.

4.1.2.3 The `policy-map` Command.

The traffic class that has been defined by the ACL and the `class-map` command must then be attached to a traffic policy. It is the policy that configures that nature and type of shaping that will be implemented on the traffic. The first step is to create the policy map using the `policy-map` command. The syntax of this command is as follows:

```
Router(config)#policy-map policy-map-descriptor
```

- `policy-map` is the command that configures a policy-map.
- `policy-map-descriptor` is the name given to the policy-map for descriptive purposes

Once the policy is named a traffic class can be assigned to it using the `class` command:

```
Router(config)#class class-name
```

- `class` is the command that defines which class will be attached to this policy.
- `class-name` is the name of the previously configured traffic class.

There is now a traffic class assigned to the policy. The final step in configuring a policy map is to define what the policy does. As the scenario calls for traffic shaping to be configured the following command will be used:

```
Router(config-pmap-c)# shape {average | peak} mean-rate [burst-size [excess-burst-size]]
```

- `shape` is the IOS command used to define traffic shaping.
- `{average | peak}` defines whether traffic will be shaped to an average or will be allowed to burst at a peak rate. In the scenario that is to be investigated traffic will be shaped at an average rate.
- `mean-rate` is the rate that traffic will be shaped to in bits per second (bps)
- `[burst-size [excess-burst-size]]` is concerned with setting the maximum burst size and excess burst size. Both of which will be configured in bps. (Cisco, 2010e)

An example of the above full policy map that may be used in this scenario is as follows.

```
Router(config)#policy-map traffic_to_server
Router(config)#class traffic_from_host_A
Router(config-pmap-c)#shape average 2000000
```

4.1.2.4 The `service-policy` Command.

The final step in configuring class based shaping on a Cisco router is to enforce the configured policy onto an interface. This is done using the `service-policy` command. Firstly an interface must be chosen:

```
Router(config)#interface <interface ID>
```

- `interface` the IOS command to enter interface configuration mode.
- `<interface ID>` defines which interface is being configured

Then the service policy is implemented on the interface:

```
Router(config-if)#service-policy output policy-map-name
```

- `service-policy` is the command used to configure the policy on the interface.
- `output` specifies that the policy should be applied to outbound traffic.
- `policy-map-name` is the name of the policy map configured previously. (Cisco, 2010e)

For example:

```
Router(config)#interface fa0/1
Router(config-if)#service-policy output traffic_to_server
```

Once the router had been configured traffic from the desired sources should be shaped.

4.1.3 Cisco Hardware To Be Used.

The hardware that will be configured is a Cisco 2600 series router. It is a 2621 model which has two Fast Ethernet ports, a 50Mhz RISC processor and 32MB of RAM. (Cnet, 2010). This is legacy hardware that has now been marked as “end of life” by Cisco themselves.

4.2 Overview of Linux.

4.2.1 GNU Linux.

GNU Linux is an open source operating system built upon the Linux kernel which was originally developed in 1991 by Linus Torvalds (linux.org, 2008). Linux is a versatile operating system that has a number of uses in enterprise. Linux can be configured in an Enterprise environment as a server operating system or a desktop operating system but most importantly for this project it can also act as a router operating system. The Linux kernel contains a number of programs that facilitate networking and since kernel 2.4 Linux contains the ability to manage and control networking traffic using the `tc` command (Hubert, 2004). With the correct configuration it should be possible to use a normal desktop PC to manage and shape network traffic as efficiently as a proprietary stand alone appliance such as a Cisco router.

4.2.2 Ubuntu Linux.

Ubuntu Linux is a free Linux distribution developed and distributed by Canonical Ltd. Version 9.10 of the Ubuntu software is utilised in this report. Ubuntu was chosen as it has a large amount of freely available documentation and has been used previously by the author. Ubuntu is also considered to be a very mature and stable distribution which can deliver high performance. QoS implementation and thus traffic shaping can be performed by Ubuntu by use of the traffic control program that is present in the Linux Kernel.

4.2.3 Linux Traffic Control (tc).

This report does not aim to investigate the mechanisms of Linux traffic control in any more depth than the queuing and shaping mechanisms employed. If a knowledge of the fundamentals of Linux traffic control is required then section 2.1 of Leonardo Balliache's "*Differentiated Service on Linux HOWTO*" (Balliache, 2003) should be read, as it provides a good introduction to the topic.

Tc is the Linux program used to manage and control network traffic. tc is part of the "iproute2" software package which is a collection of programs which can control the TCP/IP networking within the Linux system (Linux Foundation, 2009).

tc, as its name suggests is concerned with the control of traffic on a TCP/IP network. By using this tool it is possible to implement multiple QoS policies on traffic passing through the machines interfaces. It is worth mentioning at this point that the traffic can only be controlled on the outgoing interface of a device, incoming data streams can not be affected by tc. This is not much of a problem for this report's implementation as it is possible to shape traffic flows at both interfaces of the machine thus providing limitation on the hosts upload and download speed. As shown in Fig.4.2.

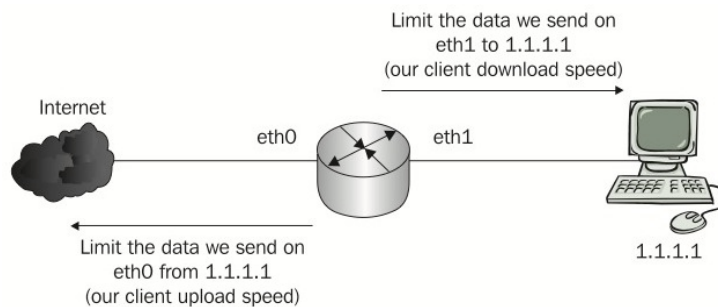


Fig.4.2

A simple Network (Gheorghe, 2006).

tc is able to control traffic by using queuing disciplines (qdiscs) implemented on a network interface. Some of the qdiscs will be familiar from section 2.5. The qdiscs implemented in the Linux kernel are listed below:

- *"FIFO (pfifo and bpfifo): The simplest qdisc, which functions by the First In, First Out rule. FIFO algorithms have a queue size limit (buffer size), which can be defined in packets for pfifo or in bytes for bpfifo.*
- *pfifo_fast: The default qdisc on all Linux interfaces. It's important to know how pfifo_fast works; so we'll explain it soon.*
- *Token Bucket Filter (tbf): A simple qdisc that is perfect for slowing down an interface to a specified rate. It can allow short bursts over the specified rate and is very processor friendly.*
- *Stochastic Fair Queuing (SFQ): One of the most widely used qdiscs. SFQ tries to fairly distribute the transmitting data among a number of flows.*
- *Enhanced Stochastic Fair Queuing (ESFQ): Not included in the Linux kernel, it works in the same manner as SFQ with the exception that the user can control more of the algorithm's parameters such as depth (flows) limit, hash table size options (hardcoded in original SFQ) and hash types.*

- *Random Early Detection and Generic Random Early Detection (RED and GRED): qdiscs suitable for backbone data queuing, with data rates over 100 Mbps.*" (Gheorghe, 2006)

These qdiscs are inherently classless in their simplest implementation they effect all traffic and treat all traffic the same. In order to have different traffic controlled in a different way than other traffic then classification needs to be configured. There are two classful qdiscs that can used to classify traffic and then perform shaping based on the traffics class. Class based queuing (CBQ) and Hierarchical Token Bucket (HTB).

4.2.4 Class Based Queuing (CBQ) On A Linux Platform.

CBQ is similar to the class based shaping discussed in section 4.1.1.2 of this report. Both systems classify traffic and then make decisions on how to treat the traffic based on a policy that is enforced. Section 2.5.5 explains how CBQ works.

CBQ on Linux is implemented in a Hierarchical fashion. Every interface has a root qdisc implemented upon it which has direct communication with the kernel. A child class is enforced on the root qdisc. This child class can have further child classes, and these child classes have their own qdiscs. These child qdiscs can then have further child classes, known as leaf classes (Gheorghe, 2006). As shown in Fig.4.3.

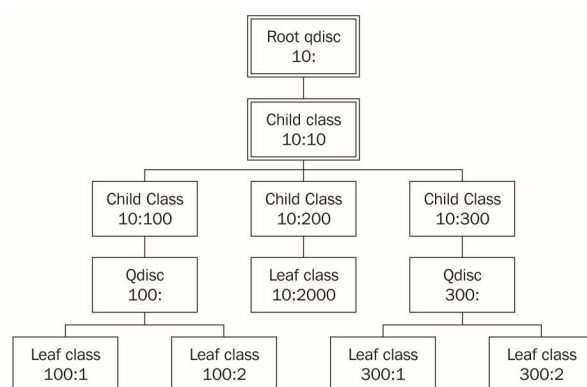


Fig.4.3

The hierarchical qdisc/class relationship on a Linux Interface (Gheorghe, 2006).

Each child class has a qdisc attached to it for scheduling the packets within that class. So essentially the leaf qdiscs queue and schedule their traffic, which is then queued and scheduled by the child qdiscs and then finally by the root qdisc. This process allows for traffic to be shaped to specific levels of service. The CBQ qdisc "*is the most complex qdisc available, the most hyped, the least understood, and probably the trickiest one to get right*" (Hubert, 2004) for these reasons that an alternative method was developed. That method is Hierarchical Token Bucket or HTB.

4.2.5 Hierarchical Token Bucket (HTB).

HTB was developed by Martin Devera in order to provide "*a more understandable, intuitive and faster replacement for the CBQ qdisc in Linux*" (Devera, 2002). HTB is a refinement of CBQ it uses the same hierarchical structure shown in Fig.4.3. HTB introduces the concept a token bucket (see section 2.3.2) to the class based shaping mechanism and thus introduces fine grained control. Traffic can be classified using the child class/leaf class structure, then

queued using the corresponding qdiscs. When the packets eventually move up the hierarchy and reach the root qdisc they enter the token bucket and the data flows are successfully shaped.

The improvements that HTB brings over CBQ are as follows:

- Finer control over traffic (Brown, 2006)
- Where CBQ uses link idle time calculations to perform shaping. Instead it uses a token bucket. (Hubert, 2004)
- HTB is easier to configure. CBQs configuration is complicated even in simple implementations. (Hubert, 2004)
- Due to this relative simplicity of configuration HTB scales to larger implementations a lot easier than CBQ (Devera, 2002).

It is for these reasons that HTB will be implemented in the Linux shaping experiment defined later in this report.

4.2.4 Configuring Linux For Networking.

This section provides information as to how the Linux operating system can be configured to act as a router and traffic controller.

4.2.4.1 Enabling Routing.

Assuming that the machine to be configured has at least two networking interfaces it can be configured to route IP traffic.

The first step is to ensure that both interfaces are configured with static IP addresses on different subnets.

Set eth0 to have the IP address 10.0.0.1:

```
$ sudo ifconfig eth0 10.0.0.1
```

Set eth1 to have the IP address 192.268.0.1

```
$ sudo ifconfig eth1 192.168.0.1
```

The next step is to enable IP routing between the two interfaces using the command by enabling IP forwarding.

```
$ sudo sh -c "echo 1 > /proc/sys/net/ipv4/ip_forward"
```

An edit must also be made to the `/etc/sysctl.conf` file. These two lines must be added:

```
net.ipv4.conf.default.forwarding=1
net.ipv4.conf.all.forwarding=1
```

Once these configuration steps have been completed. The device will route IP traffic.

4.2.4.2 Configuring Shaping.

Shaping is configured using the `tc` command. The following example is based on the simple network outlined in Fig.4.2 and shows how a single host would be shaped to have a synchronous 1Mbps connection. `tc` uses three commands to achieve this:

- `tc qdisc` - which governs the queuing disciplines
- `tc class` - which governs traffic classes
- `tc filter` - which governs how `tc` identifies what data to shape

On the outbound (or upload) interface `eth0` the first step is to configure a root `qdisc`. This is done using the command:

```
Router$ tc qdisc add dev eth0 root handle 1: htb
```

This command creates a HTB `qdisc` on `eth0`, it is designated root and has the handle 1. The next step is to define a child class for this root `qdisc`. This is done with the following command:

```
Router$ tc class add dev eth0 parent 1:0 classid 1:1 htb rate 100Mbit
```

The class is created on interface `eth0` and its parent entity is defined as 1:0 i.e. the root `qdisc`. The class is given the class ID of 1:1 and is defined as a 100Mbps HTB class. The 100Mbps is the speed of the interface. It is recommended that the child class of the root `qdisc` should always be configured to match the speed of the interface.

The root `qdisc` and class are now configured. The next stage is to create a further child class that will be shaped to the speed required:

```
Router$ tc class add dev eth0 parent 1:1 classid 1:10 htb rate 1Mbit
```

This command adds a further child class on `eth0` its parent is the previous child class 1:1 and assigned the handle 1:10 it is a HTB class and will shape traffic to 1Mbps. Referring back to Fig.4.3 shows that the next command must create a `qdisc` for the child class. This is done as follows:

```
Router$ tc qdisc add dev eth1 parent 1:10 sfq quantum 1514b perturb 15
```

A new `qdisc` has been created as a child of the 1:10 class, it uses a SFQ `qdisc`. The `quantum`, and `perturb` commands relate to the SFQ discipline.

The final configuration step is to classify what traffic will be shaped by the `qdisc`. To do this the `tc filter` command is employed as follows:

```
Router$ tc filter add dev eth0 protocol ip parent 1:0 prio 5 u32 match ip src 1.1.1.1 flowid 1:10
```

This filter is added to the eth0 interface. The IP protocol is defined along with the parent root qdisc 1:0. The match command in this statements catches any traffic coming from the 1.1.1.1 host and assigns it a flow ID of 1:10. Once this command has been entered any traffic coming from 1.1.1.1 will be shaped to a limit of 1Mbps as it leaves eth0.

In order to shape the downstream traffic coming from the internet to 1.1.1.1 another HTB qdisc needs to be implemented on eth1. These are the commands that will enable this:

```
Router$ tc qdisc add dev eth1 root handle 2: htb
Router$ tc class add dev eth1 parent 2:0 classid 2:1 htb rate
100Mbit
Router$ tc class add dev eth1 parent 2:1 classid 2:10 htb rate 1Mbit
Router$ tc qdisc add dev eth1 parent 2:10 sfq quantum 1514b perturb
15
Router$ tc filter add dev eth1 protocol ip parent 2:0 prio 5 u32
match ip dst 1.1.1.1 flowid 2:10
```

Notice that the root qdisc's handle is now 2:. This is because you can not have two qdiscs on the same system with the same identifier as this will cause problems. Also notice that the match statement in the tc filter now matches a destination address rather than source address. This address could easily be substituted for a range of IP addresses allowing a number of hosts to be shaped.

More information about configuring tc to shape network traffic can be found in Chapter 3 of the book: "*Designing and Implementing Linux Firewalls and QoS using netfilter, iproute2, NAT, and L7-filter*" by Lucian Gheorghe (2006).

4.2.5 Hardware Implemented.

The shaping experiment will be performed using machine with a 2.8Ghz Pentium 4 processor and 1Gb of RAM. Two 10/100 network cards have also been installed.

4.3 What Will Be Measured?

This section lists the networking properties that will be measured and documented in the following experimental sections.

4.3.1 TCP Throughput.

TCP throughput is a measure of how much TCP traffic is passed on the link over a certain period of time. It is usually measured in bits per second (bps). This will be an important measurement as it will give the clearest indication as to whether the traffic is being shaped and how accurate the shaping is.

4.3.2 UDP Throughput.

Same as TCP throughput but with UDP traffic instead. This measurement will again give information about how well the shaping mechanisms perform. If possible UDP streams of differing bit-rates will be used to gauge what difference if any the shaping has upon them.

4.3.3 Network Latency.

Network latency is the time taken for a packet to traverse the network. It is usually defined as the Round Trip time (RTT) of the packet i.e. the time taken for a packet to be sent to its destination and a response received by the sender. Latency can be measured using a simple ping test which provides a measurement of the RTT though "*the use of the ICMP protocol's mandatory ECHO_REQUEST datagram to elicit an ICMP ECHO_RESPONSE from a host or gateway.*" (linux.die.net, Not Dated) Latency is affected by a number of different factors such as packet transmission speed, packet propagation speed and the affects on the packet of any connected network devices

4.3.4 Jitter.

Jitter is defined as "*The variation (i.e. standard deviation) in the packet arrival times*" (Tanenbaum, 2003). It can be calculated by analysing the time taken for a number of packets to arrive at their destination. It is important to have a low jitter value when streaming video or audio, such as VOIP. Fig.4.4 shows an example of a low jitter result and a high jitter result:

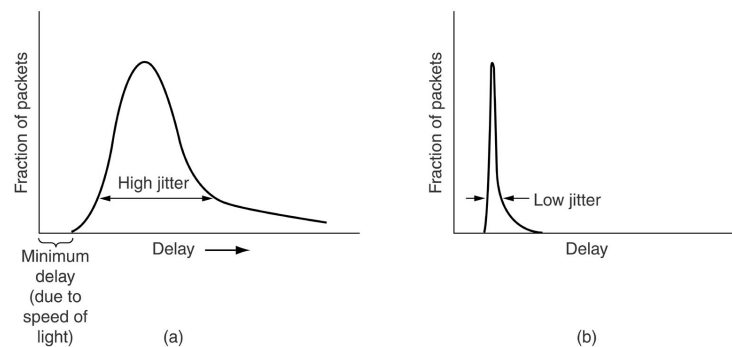


Fig.4.4

Jitter. - (a) High Jitter. (b) Low Jitter (Tannenbaum, 2003).

4.3.5 Packet Loss.

Packet Loss can be defined "as an error that occurs when data networks are overly congested. When pieces of data ("packets") are unable to be transmitted, they are sometimes "thrown out" by the network. Packet loss may or may not be disruptive to the recipient of the data, depending on the severity of loss." (Harvard, Not Dated)

4.3.6 Out Of Order Arrivals.

In a networked system it is possible for packets to arrive in a sequence that is different from the sequence in which they have been transmitted. TCP has a sequence number in its header and thus has the ability to re-order packets upon delivery (IETF, 1981). UDP does not contain a sequence number in its header and thus does not have the ability to re-order packets upon delivery (IETF, 1980).

4.3.8 Percentage CPU Utilisation.

This measurement will identify what percentage of the system's CPU is in use and thus give an idea of how hard the system is working.

4.3.9 Percentage Memory Utilisation.

This measurement will indicate how much of the systems memory is in use and will again give an idea as to how much load is being placed on the system.

4.4 Measurement Tools Considered.

In order to measure the attributes discussed in the previous section several tools are required. This section will discuss those tools and the reasoning behind why they have been chosen or rejected.

4.4.1 Iperf.

Iperf is an open source program that has the ability to measure maximum UDP and TCP bandwidth utilisation, jitter and packet loss (Iperf, 2008). Iperf allows for very detailed configuration, some of the most useful features of iperf are listed below:

- TCP bandwidth testing can be performed with various window sizes.
- UDP bandwidth testing can be performed with streams of various bit-rates.
- When testing with UDP streams iperf calculates the jitter of the connection as well as any packet loss.
- Iperf has the ability to transmit test traffic over a precise end user defined timescale.
- Precise throughput checks can be made at set intervals throughout any test.
- Iperf can act as a client or a server on each machine.

More information about Iperf can be found in Appendix I.

4.4.2 Q-Check.

Q-Check is a piece of software produced by Ixia which shares many similar features with iperf. Q-Check has the ability to:

- Measure latency and RTT
- Simulate traffic flows across a network.
- Use different transmission protocols such as TCP, UDP and RTP.
- Run trace-routes.
- Determine traffic rate and packet loss (ixchariot.com, 2010)

More detailed information about q-Check can be found in Appendix II

4.4.3 RUDE & CRUDE.

RUDE & CRUDE are a Real-time UDP Data Emitter (RUDE) and a Collector for RUDE (CRUDE). These two Linux programs provide the ability to create UDP streams from one host and receive and measure them at another. "*RUDE is a small and flexible program that generates traffic to the network, which can be received and logged on the other side of the network with the CRUDE.*" (RUDE, 2002)

4.4.4 ping.

ping is the standard networking tool for measuring network latency. It is an acronym that stands for "Packet INternet Groper". It works by using the Internet Control Message Protocol (ICMP) to transmit an Echo Request packet to a host, the host responds with an

Echo Response. From the echo response packet the RTT of the packet can be calculated. ping has the ability to alter the size of the packet to gain information as to how actual data packets may behave.

White (2006) expresses concerns that ping may not be the best tool for measuring RTT and latency across the internet due to the points raised in Karl Auerbach's 2004 white paper "Limitations of ICMP Echo for network measurement" (Auerbach, 2004). This may be true for a packet traversing the internet which will pass through multiple nodes. However in this report all testing will be done across a lab condition LAN so these issues should not come into play and no alternative to ping will be sought.

4.4.5 top.

top is a command on Unix/Linux systems that displays the processor activity and memory usage in real time. Using this tool regular readings of the CPU usage of the tc program can be made. Running top in "batch" mode by appending -b to the top command like so:

```
Router$ top -b
```

This makes top run consecutively taking a full readout every few seconds. Combining this command with a grep command like so:

```
Router$ top -b | grep -i "tc"
```

Will output only the lines that contain the words contained in the quotes. The ability to filter only the program we wish to investigate will be very useful.

4.4.6 show proc cpu / show proc memory.

These two Cisco IOS commands will allow the CPU and memory usage to be measured. The show proc cpu command delivers a host of information. A sample output of show proc cpu is shown below:

```
router#show processes cpu
CPU utilization for five seconds: 8%/4%; one minute: 6%; five
minutes: 5%
PID Runtime (uS) Invoked uSecs 5Sec 1Min 5Min TTY Process
1 384 32789 11 0.00% 0.00% 0.00% 0 Load Meter
2 2752 1179 2334 0.73% 1.06% 0.29% 0 Exec
3 318592 5273 60419 0.00% 0.15% 0.17% 0 Check heaps
4 4 1 4000 0.00% 0.00% 0.00% 0 Pool Manager
5 6472 6568 985 0.00% 0.00% 0.00% 0 ARP Input
< output omitted >
```

As can be seen the top line gives the cpu utilisation in the past five seconds, one minute and five minutes. Depending on how the command is used any one of the three results may be most useful.

show proc memory provides the following output:

```

router>show processes memory
Total: 106206400, Used: 7479116, Free: 98727284
PID TTY Allocated Freed Holding Getbufs Retbufs Process
0 0 81648 1808 6577644 0 0 *Init*
0 0 572 123196 572 0 0 *Sched*
0 0 10750692 3442000 5812 2813524 0 *Dead*
1 0 276 276 3804 0 0 Load Meter
2 0 228 0 7032 0 0 CEF Scanner
3 0 0 0 6804 0 0 Check heaps
4 0 18444 0 25248 0 0 Chunk Manager
5 0 96 0 6900 0 0 Pool Manager
< output omitted >

```

This command does not give the memory usage over a set time period so the command needs to be run in the middle of the test. The total memory in use will be the measurement that will be recorded.

4.4.7 Chosen Measurement Tools.

The OS system monitor tools `top` and `show proc` are the logical choices to measure CPU and memory usage as they are baked into the operating systems and have immediate access to the data that is required to be recorded.

For measuring latency and RTT `ping` shall be used as it is the standard for latency measurements. This experiment is taking place on a laboratory test-bed LAN hence the problems highlighted by Auerbach (2004) are not relevant.

The final decision is which traffic simulator to use? RUDE & CRUDE can only send UDP traffic, whilst this is an important part of the experiment it would be better to use a simulator that can transmit both TCP and UDP traffic. Iperf and Q-check can send and receive UDP and TCP streams. However Iperf is the most configurable as almost every attribute of the streams can be modified to suit the testing pattern required. It is because of this flexibility in configuration that Iperf has been chosen over Q-Check to simulate TCP and UDP traffic on the test network.

5.0 Experimental Investigation Methodology.

5.1 Objectives Of Investigation.

The main objective of the investigation is to compare the traffic shaping implementations of Linux and Cisco Systems. This will be achieved by monitoring the effect of network traffic and recording the attributes of this traffic specified in section 4.3. The investigation will aim to provide proof that traffic can indeed be shaped to a pre-defined limit and that the traffic is not affected adversely by the shaping mechanisms. The definable aims of the experimental investigation are as follows:

- Establish a baseline for each network topology on each platform with no shaping configuration implemented.
- Configure both platforms for shaping each of the scenarios outlined in section 5.3.

- Test each scenario on each platform using the testing methodology described in section 5.4.
- Gather results of the tests.
- Analyse results (Section 6)

The results of the shaping testing will then be analysed. It is hoped that this analysis will confirm that these methods of shaping would be suitable for use by an ISP. Meaning that the shaping is effective and traffic is limited to the pre-determined value that was configured. In addition it is possible to shape at different rates in different directions and different hosts can be shaped at different limits.

5.2 Implemented Network Topologies.

Two network topologies are used in this investigation. The first is a simple client and server topology as shown in Fig.5.1. The client connects to a switch which in turn connects to the building router. This router then connects directly to the server, which will act as "the internet" in this investigation.

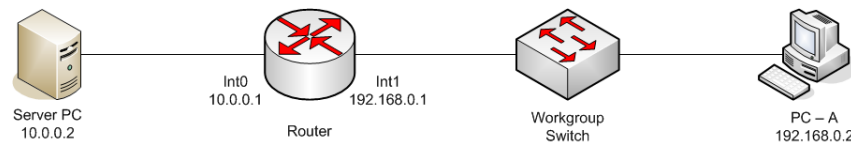


Fig.5.1

Topology one. With a single client.

The second topology adds a second client to the "building" side of the router this is demonstrated in Fig.5.2

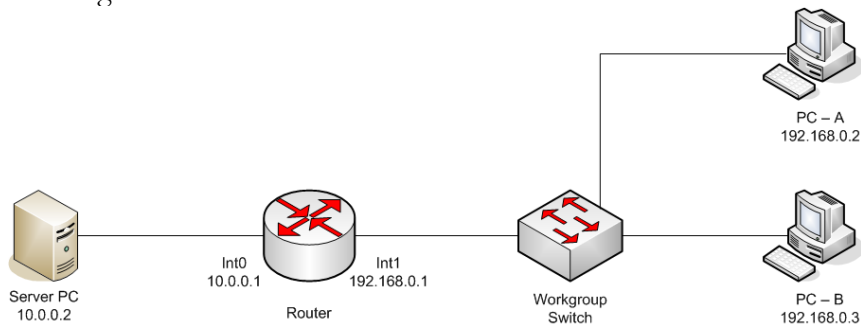


Fig.5.2

Topology two. With two clients.

The router will either be the Cisco router or the Linux machine. The 192.168.0.0/24 network is considered to be the "building" network in each topology and the 10.0.0.0/24 network is considered to be the "internet". Therefore any traffic flowing from the Server PC to the host PC is considered downstream (or downloaded) traffic and any traffic flowing from a host PC to the server PC is considered upstream (or uploaded) traffic. All the tests on topology two were conducted simultaneously. i.e. any ping tests or iperf streams were performed from (or to) both hosts at the same time. This was done in order to gain some insight into scalability and how the routers handle multiple streams.

5.3 Traffic Shaping Scenarios.

In order to fully test the shaping mechanisms of the two implementations five shaping scenarios are constructed and tested on each platform. The first three scenarios were deployed on Topology one. They are:

Scenario A:

The single host PC is traffic shaped so that its upload and download traffic speed can not exceed 1Mbps. This would be an acceptable speed for an ISP's "entry level" broadband speed package.

Scenario B:

The single host PC is traffic shaped so that its upload and download traffic speed can not exceed 10Mbps. This would be an acceptable speed for an ISP's "premium" or "mid-range" broadband speed package.

Scenario C:

The single host PC is traffic shaped so that it has a download speed of 10Mbps and an upload speed of 1Mbps. This asymmetric shaping policy is similar to that of ADSL and a similar service may wish to be sold by an MTB ISP.

The final two scenarios were implemented on topology two and hope to prove that separate hosts can be traffic shaped to similar or different traffic speeds.

Scenario D:

Both PC-A and PC-B are shaped at a symmetric 10Mbps traffic rate. This should show that both hosts can connect at the full speed of their connection and no borrowing or sharing of bandwidth occurs.

Scenario E:

PC-A is shaped to 1Mbps in both traffic directions and PC-B is shaped to 10Mbps in both directions. This scenario should show that individual hosts can be shaped at different speeds. Thus allowing an ISP to offer a tiered connection rate service at the MTB.

The router configuration scripts for each of these scenarios can be found in appendices IV & V.

5.4 Testing Methodology.

Section 4.3 defined the network traffic attributes that are to be measured. To recap, these are:

- TCP & UDP Throughput
- Network Latency
- Jitter
- Packet loss
- Out of order packets

The percentage CPU and memory utilisation on the router also need to be measured.

5.4.1 TCP Throughput.

To measure TCP throughput Iperf is used to generate and document a TCP stream. The test is run in both directions to determine values for upload and download TCP throughput. Iperf is configured to run for 120 seconds (2 minutes). Iperf is configured to take throughput measurements every five seconds. Iperf also provides a final report that provides the average throughput for the 120 second period. The results of this test are exported to a text file for later analysis.

When testing topology two, streams in the upload direction require two separate instances of iperf to be run on the server PC. This allows for simultaneous transmission of streams. In order for the traffic to flow to the correct instance iperf has to be run on two different ports. This is achieved using the iperf command:

```
iperf -s -p 5002
```

This command causes iperf to run as a server on port 5002 not the default port of 5001 which is in use by the other iperf instance.

5.4.2 UDP Throughput, Jitter & Packet Loss.

UDP throughput is also measured using Iperf. However, as UDP is a connectionless protocol and has no means to self regulate its transmission speed it is important to make sure that several streams of varying bitrates were used. This was done so that the effect of shaping on different traffic bitrates can be observed. In all five streams of the following bitrates are used:

- 512kbps
- 1Mbps
- 2Mbps
- 10Mbps
- 20Mbps

These streams are again set to run for 120 seconds and Iperf is instructed to take throughput readings every 5 seconds. Again the tests are performed in both directions, in order to gather data on both the upstream and downstream traffic. For each of these streams Iperf was able to record:

- The average total UDP throughput.
- The recorded jitter at the receiving server.
- Percentage packet loss.

As with the TCP tests when the two host topology was in use iperf was run on two separate ports on the server PC. Using the command:

```
iperf -s -u -p 5002
```

This caused the iperf instance to run on port 5002 not the default port of 5001. All streams on topology two are run simultaneously.

Again the outputs of all these tests are saved to a text file for further analysis. The commands used to configure Iperf for both TCP and UDP testing can be found in Appendix III.

5.4.3 Network Latency.

Network latency is recorded using a simple `ping` command. In order to refine accuracy 100 ICMP echo packets are sent from the host to the server, the size of these packets is set at 1024 bytes in order to more closely simulate network packets. The command used is as follows:

```
ping -n 100 -l 1024
```

This command is repeated on the server PC so as to have measured network latency in both directions of traffic flow.

5.4.4 Router CPU & Memory Utilisation.

The method for recording CPU and Memory utilisation was different for each platform.

5.4.4.1 Measuring CPU & Memory Utilisation On Linux.

As each of the Iperf streams are running, the `top` command is run on the router in batch mode. In order to filter out only the CPU and memory usage the following `grep` command is used:

```
top -b | grep -i "Cpu(s)" -A2
```

This command returns the line that contains "Cpu(s)" which is the CPU utilisation, and the following two lines which give memory and swap file utilisation. Again the data obtained for each traffic stream is saved to a text file for later analysis.

5.4.4.2 Measuring CPU & Memory Utilisation On Cisco.

The `show proc` commands discussed in section 4.4.6 is used to record the memory and CPU utilisation on the Cisco router. Unfortunately there was no way of running both commands simultaneously so they had to be run successively.

The `show proc mem` command is entered at precisely 55 seconds into the 120 second streams. The output is recorded and saved for later analysis. At 115 seconds into the stream the `show proc cpu` command is entered and again the output is recorded and saved for later analysis. The CPU usage over the last 1min is displayed on the first line of the commands output and this is the data that is required.

It is also worth mentioning that the act of using these measuring tools may also affect the amount of CPU and memory usage. However, due to the nature of this problem it would be hard to verify.

5.5 The Importance Of Obtaining Baseline.

Before any shaping configuration was configured on any of the topologies a baseline was obtained. This was done by performing all the tests outlined in section 5.4 and recording

the relevant data. This ensures that any data recorded from the shaping scenarios detailed in section 5.3 can be compared to traffic operating on a network upon which no shaping has been enforced. .

6.0 Experimental Results.

6.1 What Is Expected?

Before the experiment was carried out it was expected that both methods of traffic shaping would perform to a satisfactory level i.e. the simulated traffic would conform to the two levels of shaping configured. It was hoped that the shaping methods would successfully conform the TCP and UDP throughput to less than the configured limits but by no more than a margin of 10%.

It was also possible to predict the amount of UDP packets that would be dropped for each of the five bitrates transmitted across the topology. Table 6.1 shows the expected UDP packet loss for all five UDP bitrates at both shaping levels.

UDP Bitrate (Mbps)	Expected Packet Loss When Shaped To 1Mbps (%)	Expected Packet Loss When Shaped To 10Mbps (%)
0.512	0	0
1.000	0	0
2.000	50	0
10.000	90	0
20.000	95	50

Table.6.1

The Expected UDP Packet Loss At Both Configured Shaping Levels.

What is not known is how each shaping method will affect the other attributes of the UDP traffic i.e. jitter, network latency and out of order packets. It is also unknown how the percentage CPU and memory use of each of the tested platforms are affected by the implementation of the shaping mechanisms.

6.2 Analysis Of Recorded Data.

The raw data obtained for tests run on the Cisco & Linux platforms can be found in Appendix VII and VIII respectively. The scenarios all implemented correctly and traffic was shaped to the desired levels. However as mentioned in section 6.1 there are still unknowns. The next few sections aim to investigate these unknown factors.

6.2.1 Analysis Of The Effect Of Shaping On TCP Throughput.

Table 6.2 contains the average values of baseline TCP throughput and shaped TCP throughput on each platform and topology.

Platform/Topology	Average Baseline Throughput (Mbps)	Average 1Mbps Shaped Throughput (Mbps)	Average 10Mbps Shaped Throughput (Mbps)
Cisco/Single Host	49.1	0.962	9.62
Cisco/Dual Host	25.2	0.962	9.62
Linux/Single Host	93.8	0.958	9.58
Linux/Dual Host	47.3	0.958	9.58

Table.6.2
The Effects Of Shaping On TCP Throughput

Table 6.2. demonstrates that Shaping works. Cisco's generic traffic shaping appears to shape closer to the configured amount than Linux's HTP implementation. Cisco shaped to within 96.2% on the desired limit and Linux shaped to 95.8% of the limit at both shaping levels. This was well within the 10% margin stipulated for success earlier.

What is surprising is the baseline throughput for the Cisco platform because even in control conditions the Cisco 2621 router was not able to attain 50Mbps. This is probably due to the specifications of the router but it is disappointing that the device could not attain close to the protocol maximum of 100Mbps.

6.2.2 Analysis Of The Effect of Shaping On UDP Throughput.

The following two tables, Tables 6.3 and 6.4, document the effect of shaping on the UDP throughput on the Cisco and Linux platforms respectively.

Topology	Average UDP Bitrate (Mbps)	Average Baseline Throughput (Mbps)	Average 1Mbps Shaped Throughput (Mbps)	Average 10Mbps Shaped Throughput (Mbps)
Single Host	0.512	0.512	0.512	0.512
Single Host	1.000	1.000	0.972	1.000
Single Host	2.000	2.000	0.972	2.000
Single Host	10.000	10.000	0.972	9.073
Single Host	20.000	20.000	0.972	9.720
Dual Host	0.512	0.455	0.452	0.447
Dual Host	1.000	0.939	0.936	0.984
Dual Host	2.000	1.860	0.967	1.845
Dual Host	10.000	8.345	0.972	8.175
Dual Host	20.000	16.100	0.972	9.695

Table.6.3
The Effects Of Traffic Shaping On UDP Throughput On The Cisco Platform.

Topology	UDP Bitrate (Mbps)	Average Baseline Throughput (Mbps)	Average 1Mbps Shaped Throughput (Mbps)	Average 10Mbps Shaped Throughput (Mbps)
Single Host	0.512	0.512	0.512	0.512
Single Host	1.000	1.000	0.972	1.000
Single Host	2.000	2.000	0.972	2.000
Single Host	10.000	10.000	0.972	9.720
Single Host	20.000	20.000	0.972	9.720
Dual Host	0.512	0.453	0.512	0.476
Dual Host	1.000	0.978	0.970	0.967
Dual Host	2.000	1.860	0.972	1.832
Dual Host	10.000	8.523	0.972	8.253
Dual Host	20.000	16.325	0.972	9.582

Table.6.4

The Effects Of Traffic Shaping On UDP Throughput On The Linux Platform.

These tables show that UDP traffic of a bit-rate higher than that of the shaping value is shaped within expectations on both platforms. In this situation neither Cisco or Linux performs better. UDP traffic equal to the shaped level is also shaped although this is within expected margins. UDP traffic at a bit rate less than the shaped limit is not shaped.

There seems to be a problem with the results for the dual host topologies on both platforms. The baseline throughputs are less than expected. As this is happening on both platforms it is unlikely that it is a problem with either the routing mechanisms or shaping mechanisms. It is possible that the server PC is not of a powerful enough specification to handle two instances of iperf generating traffic streams. However, comparison of the baseline to the shaped throughput values shows that the shaping mechanisms are not effected by this.

6.2.3 Analysis Of The Effect of Shaping On UDP Jitter.

Tables 6.5 & 6.9 document the effects that the implementation of traffic shaping has on UDP jitter over the two tested platforms.

Topology	UDP Bitrate (Mbps)	Average Baseline Jitter (ms)	Average 1Mbps Shaped Jitter (ms)	Average 10Mbps Shaped Jitter (ms)
Single Host	0.512	0.000	1.046	0.000
Single Host	1.000	0.000	12.720	0.000
Single Host	2.000	0.000	4.509	0.000
Single Host	10.000	1.503	4.827	2.214
Single Host	20.000	0.746	6.634	3.124
Dual Host	0.512	0.006	28.036	0.198
Dual Host	1.000	1.550	9.134	0.027
Dual Host	2.000	0.461	5.453	2.295
Dual Host	10.000	0.279	8.337	1.948
Dual Host	20.000	0.393	5.993	6.095

Table.6.5

The Effects Of Traffic Shaping On UDP Jitter On The Cisco Platform.

Topology	UDP Bitrate (Mbps)	Average Baseline Jitter (ms)	Average 1Mbps Shaped Jitter (ms)	Average 10Mbps Shaped Jitter (ms)
Single Host	0.512	0.000	0.000	0.000
Single Host	1.000	0.000	140.145	0.000
Single Host	2.000	0.000	15.372	0.015
Single Host	10.000	0.004	5.791	2.841
Single Host	20.000	0.000	6.774	3.104
Dual Host	0.512	0.028	0.000	4.061
Dual Host	1.000	1.2110	3.925	1.529
Dual Host	2.000	0.353	7.443	4.495
Dual Host	10.000	0.443	7.126	2.264
Dual Host	20.000	0.026	7.675	2.182

Table.6.6

The Effects Of Traffic Shaping On UDP Jitter On The Linux Platform.

Both tables show that jitter increases when shaping is applied on both platforms. However, jitter only occurs when the stream is actually being shaped i.e. the UDP bit-rate is equal to or higher than the configured shaping level. Jitter seems to be higher when the stream is equal to the shaping level, the amount of jitter decreases as the stream bit-rate increases.

Both the Cisco and Linux implementations perform similarly in this respect. Neither platforms causes considerably higher jitter values than the other.

6.2.4 Analysis Of The Effect of Shaping On UDP Packet Loss.

As previously mentioned in section 6.2.1 it is possible to predict the values of packet loss at each shaped level. Tables 6.7 & 6.8 display the actual levels of UDP packet loss recorded. There is however one unexpected result. Streams of an equal value to the shaped limit lose more packets than is expected. Packet loss of between 2-3% is recorded, this again shows that the shaping is occurring at a level slightly lower than the specific values configured.

Topology	UDP Bitrate (Mbps)	Average Baseline Packet Loss (%)	Average 1Mbps Shaped Packet Loss (%)	Average 10Mbps Shaped Packet Loss (ms)
Single Host	0.512	0.000	0.000	0.000
Single Host	1.000	0.000	2.100	0.000
Single Host	2.000	0.003	51.000	0.000
Single Host	10.000	0.000	90.000	2.867
Single Host	20.000	0.001	95.000	51.000
Dual Host	0.512	0.000	1.250	0.003
Dual Host	1.000	0.000	1.600	0.000
Dual Host	2.000	0.025	49.500	0.030
Dual Host	10.000	0.024	88.000	1.388
Dual Host	20.000	0.001	94.000	34.1667

Table.6.7

The Effects Of Traffic Shaping On UDP Packet Loss On The Cisco Platform.

Topology	UDP Bitrate (Mbps)	Average Baseline Packet Loss (%)	Average 1Mbps Shaped Packet Loss (%)	Average 10Mbps Shaped Packet Loss (ms)
Single Host	0.512	0.000	0.000	0.000
Single Host	1.000	0.000	1.533	0.000
Single Host	2.000	0.000	51.000	0.015
Single Host	10.000	0.000	90.000	2.700
Single Host	20.000	0.230	95.000	51.000
Dual Host	0.512	0.000	0.000	0.005
Dual Host	1.000	0.000	0.880	0.000
Dual Host	2.000	0.000	48.500	0.000
Dual Host	10.000	0.011	88.500	1.952
Dual Host	20.000	0.000	94.000	48.25

Table.6.8

The Effects Of Traffic Shaping On UDP Packet Loss On The Linux Platform.

6.2.5 Analysis Of The Effect of Shaping On UDP Packet Delivery.

Tables 6.9 & 6.10 show the data retrieved about out of order packet delivery.

Topology	UDP Bitrate (Mbps)	Average Baseline Out Of Order Packets.	Average 1Mbps Shaped Out Of Order Packets	Average 10Mbps Shaped Out Of Order Packets
Single Host	0.512	0	0	0
Single Host	1.000	0	0	0
Single Host	2.000	0	0	0
Single Host	10.000	0	0	0
Single Host	20.000	0	0	0
Dual Host	0.512	0	0	0
Dual Host	1.000	0	0	0
Dual Host	2.000	0	0	0
Dual Host	10.000	0	0	0
Dual Host	20.000	0	0	0

Table.6.9

The Effects Of Traffic Shaping On UDP Packet Delivery On The Cisco Platform.

Topology	UDP Bitrate (Mbps)	Average Baseline Out Of Order Packets.	Average 1Mbps Shaped Out Of Order Packets	Average 10Mbps Shaped Out Of Order Packets
Single Host	0.512	0	0	0
Single Host	1.000	0	796	0
Single Host	2.000	0	659	0
Single Host	10.000	0	523	858
Single Host	20.000	0	513	621
Dual Host	0.512	0	0	0
Dual Host	1.000	0	560	0
Dual Host	2.000	0	668	0
Dual Host	10.000	0	532	438
Dual Host	20.000	0	515	529

Table.6.10

The Effects Of Traffic Shaping On UDP Packet Delivery On The Linux Platform.

Surprisingly the Cisco platform did not deliver a single UDP packet out of order. Whether this is because the Cisco device is a dedicated routing machine is unknown. It is possible that the Cisco router delivers as many packets that it can in order then simply drops the rest. The Linux machine may just make a best effort delivery, and therefore some packets will be delivered out of sequence. It is safe to say that the Linux device did in fact deliver

many packets out of order, and as UDP has no facility to reorder packets these would cause problems. On the Linux platform however, out of order delivery only occurred at bit-rates that were equal to or greater than the level of shaping implemented. Meaning that streams lower than the shaped limit were unaffected.

6.2.6 Analysis Of The Effect of Shaping On CPU & Memory Use.

6.2.6.1 When Shaping TCP Traffic.

Table 6.11 shows the average CPU usage when the TCP stream is shaped.

Platform/Topology	Average Baseline CPU Usage (%)	Average 1Mbps Shaped CPU Usage (%)	Average 10Mbps Shaped CPU Usage (%)
Cisco/Single Host	39.000	1.000	29.000
Cisco/Dual Host	21.000	28.000	49.000
Linux/Single Host	2.375	0.127	0.113
Linux/Dual Host	0.130	0.105	0.112

Table.6.11

The effect of shaping on CPU Usage.

The two platforms seem to behave differently when shaping is implemented. Cisco seems to have relatively high CPU usage at baseline and then, in the case of the single host, CPU usage seems to decrease when the shaping is applied. It is possible that the shaping calculations do not take place on the main processor and this is an indicator of such an architecture. However, this is pure conjecture and besides on the dual host topology CPU uses rises when shaping is applied and rises again when a second host is added to the topology. It is mentioned earlier that the Cisco router used in this example seems underpowered and this could be a result of that.

The Linux system behaves slightly differently. Again on the single host topology the CPU usage seems to decrease as shaping as is applied and reduces again when a second host is added. This is counter intuitive and this decrease may be indicative of other background processes that are running. On the dual host topology the CPU usage does not seem to trend in any direction. It may be that the CPU usage due to shaping is not measurable by this method.

Table 6.12 documents the average memory in use during he shaping experiments.

Platform/Topology	Average Baseline Memory Usage (%)	Average 1Mbps Shaped Memory Usage (%)	Average 10Mbps Shaped Memory Usage (%)
Cisco/Single Host	19.90	20.50	20.70
Cisco/Dual Host	20.10	20.9	21.14
Linux/Single Host	11.10	11.63	11.64
Linux/Dual Host	17.97	18.12	16.38

Table.6.12

The Effect of Shaping On Memory Usage.

There does not seem to be a discernable trend between implementing shaping and then increasing the shaping level. However, both platforms exhibited a rise in memory usage when a second host was added to the topology. This is probably due to the additional buffering that the second host requires.

Overall the tests on CPU and memory usage seem largely inconclusive. No real trends of either increasing or decreasing usage have been observed.

6.2.6.2 When Shaping UDP Traffic.

The following four tables, Tables 6.13-16 document the CPU and memory uses when UDP traffic is shaped on the two Cisco and Linux Platforms.

Topology	UDP Bitrate (Mbps)	Average Baseline CPU Usage (%)	Average 1Mbps Shaped CPU Usage (%)	Average 10Mbps Shaped CPU Usage (%)
Single Host	0.512	1.50	1.00	1.33
Single Host	1.000	1.00	2.00	2.00
Single Host	2.000	2.50	5.00	4.00
Single Host	10.000	13.00	22.33	29.00
Single Host	20.000	27.00	45.00	54.00
Dual Host	0.512	1.50	2.50	2.16
Dual Host	1.000	2.00	4.00	3.67
Dual Host	2.000	4.50	9.00	8.33
Dual Host	10.000	22.00	43.50	44.50
Dual Host	20.000	42.5	47.00	44.33

Table.6.13

The Effect Of Shaping on CPU Usage on the Cisco Platform.

Topology	UDP Bitrate (Mbps)	Average Baseline CPU Usage (%)	Average 1Mbps Shaped CPU Usage (%)	Average 10Mbps Shaped CPU Usage (%)
Single Host	0.512	0.145	0.140	0.127
Single Host	1.000	0.145	0.147	0.150
Single Host	2.000	0.135	0.143	0.143
Single Host	10.000	0.145	0.173	0.110
Single Host	20.000	0.135	0.133	0.100
Dual Host	0.512	0.115	0.120	0.143
Dual Host	1.000	0.140	0.125	0.132
Dual Host	2.000	0.130	0.110	0.123
Dual Host	10.000	0.140	0.105	0.108
Dual Host	20.000	0.125	0.105	0.122

Table.6.14

The Effect Of Shaping on CPU Usage on the Linux Platform.

The CPU only seems to have a discernable increase when shaping is implemented on the Cisco platform. CPU usage does increase as streams of a bit-rate higher than the shaping limit are used. The 20Mbps stream causes the Cisco platform to utilise almost 50% of the CPU cycles on the machine. However, on the Linux platform there seem to be no real increase of CPU usage across the board. This is in some way understandable as the Linux machine has a CPU that is far more powerful then the Cisco router.

Topology	UDP Bitrate (Mbps)	Average Baseline Memory Usage (%)	Average 1Mbps Shaped Memory Usage (%)	Average 10Mbps Shaped Memory Usage (%)
Single Host	0.512	19.9	20.4	20.7
Single Host	1.000	19.9	20.4	20.7
Single Host	2.000	19.9	20.4	20.7
Single Host	10.000	19.9	20.4	20.7
Single Host	20.000	19.9	20.4	20.7
Dual Host	0.512	20.1	20.9	21.1
Dual Host	1.000	20.1	20.9	21.1
Dual Host	2.000	20.1	20.9	21.1
Dual Host	10.000	20.1	20.9	21.1
Dual Host	20.000	20.1	20.9	21.1

Table.6.15
The Effect Of Shaping On Memory Usage On The Cisco Platform.

Topology	UDP Bitrate (Mbps)	Average Baseline Memory Usage (%)	Average 1Mbps Shaped Memory Usage (%)	Average 10Mbps Shaped Memory Usage (%)
Single Host	0.512	11.49	11.64	11.64
Single Host	1.000	11.55	11.64	11.64
Single Host	2.000	11.53	11.64	11.64
Single Host	10.000	11.53	11.64	11.64
Single Host	20.000	11.53	11.64	11.64
Dual Host	0.512	13.46	18.16	15.78
Dual Host	1.000	13.46	18.12	15.77
Dual Host	2.000	13.48	18.13	15.86
Dual Host	10.000	13.62	18.12	15.86
Dual Host	20.000	13.61	18.14	15.89

Table.6.13
The Effect Of Shaping On Memory Usage On The Linux Platform.

Memory usage on the Cisco platform does seem to increase slightly when shaping is implemented, but it seems to not change when streams of a larger bit rate are shaped. Memory usage on both platforms does increase when a second host is added which was expected.

6.2.7 Analysis Of The Effect of Shaping On Network Latency.

Based on the gathered data it appears that neither of the shaping implementations have a measurable effect on network latency.

7.0 Conclusions & Recommendations.

7.1 Conclusions Drawn.

The results obtained from the tests provide a proof of concept that the implemented shaping methods have the ability to limit traffic to a pre-defined bandwidth. Thus both the Cisco and Linux shaping systems can provide the end user bandwidth control required by an ISP. The scenarios implemented prove that both platforms are flexible enough in their configuration to manage bandwidth in several different ways. It is possible to configure:

- Multiple end user bandwidth levels.

- Different shaping levels in different directions to achieve asynchronicity.
- Multiple hosts can be shaped at multiple bandwidth levels.

These three abilities mean that an ISP should be able to perform all the end user bandwidth control that they require. Both the platforms can perform these tasks and perform them well. Not only is all traffic shaped to an acceptable level, but the attributes of the traffic are not affected negatively. So an ISP should be able to use these techniques to limit the maximum bandwidth the end user can achieve and thus solve the problem that this report set out to investigate.

Unfortunately the tests did not provide an insight into how much resources these shaping mechanisms consume on the device. This is partially due to the tests implemented not being granular enough and also due to the small nature of the implemented network technologies. In a real life application it is possible that hundreds of end users will be being shaped simultaneously. So it is important that the hardware in use is able to handle the load. Whilst the CPU and memory tests did not provide specific information as was hoped. They did indicate that any hardware that is to be used for this task needs to be of an adequate specification.

The Cisco hardware used for testing in this report was found to be underpowered for the task at hand. The unfortunate fact that it was not able to provide a bandwidth measurement close to the 100Mbps protocol maximum was disappointing. However this was to be expected given the nature of the device. It was legacy hardware that was only designed for use in branch operations. Any real world Cisco implementation of end user bandwidth management would need a far more powerful device than the one implemented here.

Conversely the Linux machine, even though it was of a similar vintage (approx 5 years old), handled the implementation of traffic shaping very well. In fact it was difficult to determine how many CPU cycles were actually being consumed by the `tc` program. Due to the problems in sufficiently measuring CPU and memory usage this report is unable to provide a prediction on how well the implementations will scale to larger groups of end users. It is expected that with sufficiently powerful machines shaping should scale well. However more research may need to be carried out in future to prove this.

In conclusion both Cisco and Linux traffic shaping methods can be considered for use by an ISP or network manager. Each implementation has their own positive and negative attributes. The next section will aim to investigate these attributes and provide some recommendations for any ISP or network manager wishing to use these shaping techniques.

7.2 Recommendations To ISPs and Network Managers.

There are several questions to be answered before choosing a traffic shaping implementation these include, but are not limited to:

- Is one method clearly better than the other?
- How easy are these methods to configure?
- What is the cost of implementation?
- What is the cost up upgrade?

- How well is the platform supported?

This section will attempt to provide answers to these questions but will not attempt to state which platform to choose.

7.2.1 Is there A Clear Winner?

The short answer to this question is no. There is not one method of traffic shaping that is clearly better than the other. The tests performed in this report have shown that both the Cisco and Linux platforms perform similarly. Cisco maybe has an edge on Linux when shaping UDP streams but only with regards to delivering packets in order. This is such a small and insignificant difference between the two that it should not affect the choice to be made.

If there is a winner in this report it is probably class based shaping. Both platforms use a class based approach. The ability to classify certain types of traffic and make a policy decision about it enables a great degree of flexibility when defining end user-maximums and upload/download speed configurations. One other aspect of class based shaping that has not been investigated in this report but is theoretically feasible is the possibility of using it to provide a contention ratio to the provided service. Consider a class defined by an end users IP or MAC address this class can then be shaped to say 10Mbps synchronously. The end user will then not be able to receive or transmit data at more than the configured maximum. However this bandwidth is not contended. By adding other end users to the same class they can be forced into contending for this 10Mbps. If for example 20 users were grouped into a 10Mbps shaped class then the service to each user would be a 10Mbps service with a 20:1 contention ratio. Admittedly this has not been tested, but any further research may wish to investigate this.

7.2.2 Ease Of Configuration.

Both platforms require specialist knowledge to configure for traffic shaping. The information on how to configure these platforms is relatively easy to find. Cisco has its knowledge base and Linux has the open source community which has put together several How-To's and FAQs on the subject. The actual configurations can be quite tricky to construct. As long as the desired outcomes are known then a configuration can be written. Do however be aware that there are often several ways to perform the same task on each platform. To get an idea of what the configurations look like the configuration scripts for each of the implemented testing scenarios have been included in appendices IV & V.

One advantage that Linux has over Cisco with respect to configuration is the fact that Linux is an open source technology that uses a command shell. This means that it is possible to implement a configuration solution that can be scripted and can perhaps be integrated into a control and configuration database or program. With Cisco's IOS being a proprietary software it may be more difficult to integrate into this kind of control architecture. This is something however that the ISP or network manager may need to consider.

7.2.3 Cost Of Implementation.

The initial cost of the networking hardware required in order to implement these platforms is a very important consideration. An this is an area where the Linux platform has a strong advantage over Cisco. The hardware required to run the Linux shaping implementation is a lot cheaper than Cisco. Consider the two pieces of hardware implemented in this report. The Cisco 2621 router when new would have cost over £1500 (Google, 2010), whereas the Linux hardware was a desktop machine that probably cost no more than £400. It is expected that a more powerful Cisco router would be required to implement the platform in a real world scenario. This would be even more expensive. The Linux platform could probably be rolled out in the real world using the same hardware that was used in this test. The only real problem with that would be that the Linux hardware may not be as reliable.

Reliability is another consideration. If the network needs high availability then the Cisco devices would be able to deliver this. At a price. However if the network can stand some downtime then implementing a Linux system on cheap hardware may be a consideration. It is also possible however to implement a Linux system on very high quality server grade hardware to attain the high availability of Cisco. This flexibility is another advantage of Linux.

7.2.4 Cost To Upgrade.

As mentioned in the previous section Linux is ahead of Cisco when it comes to affordability. It also beats Cisco in upgradability too. Cisco devices are sold as appliances and so their only upgradeable part is their operating system whereas Linux can be implemented on off the shelf server hardware. This means that if an implementations hardware needs to be upgraded this can be done much easier and cheaper on the Linux platform. So Linux beats Cisco on pure monetary cost. But there is one area that Cisco beats Linux hands down. Support.

7.2.5 Is It Supported?

Cisco wins at support. If it is necessary Cisco will sell you the equipment, install the equipment, configure the equipment and support the equipment for its lifetime. As long as you can afford it. This service is quite attractive to companies who do not have in-house support staff. It is possible that an ISP may wish to go down this route. The other side of this is the fact that Linux is an open source software that runs on off the shelf hardware. There is no Linux software and hardware support. To obtain that a 3rd party company would need to be contracted. Either that or have an in house support team.

Another way of looking at it is this. The Cisco hardware and software is well documented as it is a product offered for sale by Cisco Systems. The Linux documentation is as good but it is written by users and developers. So if your traffic shaping implementation develops a fault, with Cisco you can call your dedicated support agent whereas on Linux you turn to Google.

7.2.6 Total Cost Of Ownership (TCO).

The TCO is a combination of the cost of implementation, the cost of configuration, any upgrade costs, warranty cost and support costs. Plus sundry charges such as electricity

charges. Table 7.1 gives an idea of some “ballpark figures” for a 10 unit Cisco and Linux implementation.

Approximate Cost	Cisco (£)	Linux (£)
Equipment	15,000	7500
Configuration & Maintenance	200,000	210,000
Support / Warranty	15,000	12,500
Total Approximate 5 year TCO	230,000	220,000

Table.7.1

A comparison of 5 year TCO for Cisco and Linux.

The equipment cost has been calculated for the off the shelf price of a Cisco integrated services router2911 (Google, 2010b) and a server grade Linux machine of similar specification (Dell, 2010). The configuration and maintenance costs have been calculated at five times the annual wage of a Cisco (itjobswatch.com, 2010) and Linux (itjobswatch.com, 2010b) specialist respectively. The support/warranty costs have been calculated at the cost quoted for a cisco SMARTnet support contract (ciscohardwaremaintenance.com, 2010) and a contract from the Linux machine manufacturer (Dell, 2010).

In this approximate calculation of 5 year TCO Linux comes out as the cheaper of the two options. It is also worth noting that if high availability is not an issue then even cheaper Linux hardware could be chosen thus reducing the TCO even further.

7.2.7 To Conclude.

So, Cisco or Linux? This report does not aim to make that decision. However these are the main recommendations. Use Cisco if:

- You require a fully comprehensive support structure.
- You have Cisco qualified staff at your disposal.
- High network availability is a determining factor.
- Cost is not a factor.

Or use Linux if:

- Cost is a factor.
- You have staff well versed in Linux.
- You can outsource support to a 3rd party.
- You foresee hardware upgrades will be vital.

Again it is stated that both implementations perform the task well, it will be external considerations that inform your choice.

8.0 Critical Evaluation.

8.1 Were The Objectives Of The Report Satisfied?

This report’s main aims were to research, investigate and evaluate the various bandwidth control methodologies and to implement a simple network. So that the two most feasible

methods could be tested and compared. Both these objectives have been satisfied to a suitable extent.

The investigation section of the report identified Linux and Cisco traffic shaping as the two bandwidth control methods that would be tested. These two methods were then implemented on a test network and a series of scenarios were implemented and the resulting traffic was analysed. The results of these tests have proved that these two methods of traffic shaping would meet an ISP's requirements. The detailed traffic analysis has provided quantitative results about how the traffic has been affected by the shaping methods implemented.

It is hoped that anyone reading the report will be left with an understanding of bandwidth control methodologies and would then be able to make an informed decision about which one to choose, should the need arise.

8.2 Evaluation Of Experimental Results.

The results obtained from the experimental section of this report have been quantitative in nature and have provided proof that traffic shaping can be configured as desired. The analysis of the results showed that whilst the shaping methods on both platforms affect throughput as expected they also increase jitter but to a detrimental extent.

It was hoped that by measuring the CPU and memory usage on the shaping hardware, as shaping was taking place, some insight could be gained into how the topologies may scale to larger sizes. However this part of the experiment did not provide the expected results and no such insight was gained. This was largely due to the decision to monitor overall CPU and memory usage rather than finding the specific processes that were performing the shaping on each platform. If the experiment was to be run again it may be prudent to devise a more accurate way of measuring these values.

The original plan for the experimental topology was to scale the number of end users up to 10 hosts. It soon became apparent that the resources to perform such a test would not be available to the author. Instead it was decided to scale the testing topology to a single host and then a dual host topology. In the hope that this, along with the CPU and memory usage data, would indicate how such a shaped network would scale. However, as mentioned earlier the CPU and memory usage measurement methodology was not as accurate as wished and so no such information could be gained. The two host topology did however provide good quantitative data about how the network traffic is affected when multiple traffic streams are shaped simultaneously.

Due to the failure to find adequate resources to build a test network the author was forced to use his own equipment to perform the tests. This equipment worked well but the choice to use a single server PC in the dual host topology needs to be examined. This PC was a rather low powered machine with little RAM and seemed to struggle to generate the dual iperf streams. Unfortunately no analysis of the host and server systems was performed during the experimentation so it was not possible to prove that this was actually occurring. In any further testing a symmetrical network as shown in fig.7.1 should be set up so that

traffic streams can be generated at separate sources and received at separate destinations. This will ensure that no single server or host can cause such a problem.

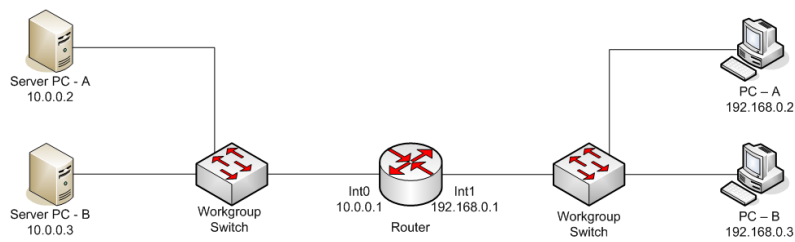


Fig.8.1

Proposed future testing topology.

8.3 Do Any Areas Require Further Research?

In the process of completing this report several areas of further study have presented themselves. Should anyone wish to refine the methodology of this report then it is suggested that the following be studied further.

8.3.1 System Monitoring Methods.

The main point of failure in this report as the inability to accurately measure the CPU and memory usage of the routers. Any further experimentation should ensure that it is able to successfully pinpoint the exact processes that are performing traffic shaping and monitor them specifically.

8.3.2 Testing For Scalability.

Knowing whether these shaping methods scale up to hundreds and possibly thousands of end users was an original objective for this report. Unfortunately it was not possible to determine this on dual user topology. Any further research should aim to implement a larger scale testing network, perhaps through the use of either virtual machines or simulation software.

8.3.3 Other Bandwidth Control Methods.

Section 3 of this report identified a number of traffic control methods that have not been investigated. Further research may wish to investigate them.

The Cisco platform has four traffic shaping methods and only one has been tested here. It also is able to perform traffic policing. It would be interesting to see how traffic policing affects network traffic when compared to shaping.

HTB shaping was chosen in Linux however CBQ is the default shaping mechanism in `tc`. Therefore it would make sense for someone to test CBQ and ensure that it controls the traffic in a comparable way to HTB.

It became apparent during the research phase of the report that it would be possible to write a whole report just on the many queuing disciplines that could have been used. A cursory attempt has been made to describe the numerous queuing disciplines encountered during research but there would definitely be some good in further investigation into how using different qdiscs in a HTB implementation affected performance.

8.3.4 Further Investigation For ISPs.

Section 1.2 identified several areas that would not be covered by this report. It is hoped that any further investigation in this field would move towards a more real world implementation and would begin to investigate how shaping would perform therein. ISPs also need to provide network security to their customers and themselves. The topologies configured in this report class traffic on IP address. This would not be practical on a public LAN as there is nothing to stop someone changing their IP. This mean that a non paying customer could feasibly spoof the IP address of someone with an account and obtain connection fraudulently. Further research into this kind of IP accountability would certainly be required before a system could be rolled out. It could be as simple as classifying traffic on a more “hard” attribute for example the MAC address rather than the IP address.

8.4 Evaluation Of The Project.

The original idea of the report was to determine if Cisco systems hardware could be used to control an end users obtainable bandwidth. The idea for this came from the authors' work with ask4™, a local ISP who deliver connectivity by means of building LANs. The author had knowlege that ask4™ achieved end user bandwidth limitation using a Liniux based solution and wished to investigate wheteher the Cisco platform could perform the same kind of traffic control. It soon became clear that in order to investigate the Cisco platform the Linux platform would also need to be investigated. It was decided that an in depth investigation into various traffic control methodologies would be carried out to gauge the suitability of each method. This investigation and research phase discovered that Cisco could deliver the same kind of bandwith contol options as Linux. And so it was decided that both of the platforms should be tested to investigate their effects on the network traffic.

This project has proved that Cisco and Linux traffic shaping mechanisms can be configured to shape end user traffic in the manner expected by an ISP. It has also provided quantitatve data on how these traffic shaping methods affect network traffic.

8.5 Evaluation Of The Project Management.

Appendix IX contains the original project specification document produced in November 2009. This document was the first step on a long road towards completeing the project and writing this report. The original tack plan contained in the project specification separated the project up into four separate phases:

- An initial investigation and research phase which ran from late October to the end of December 2009.
- The testing phase in which the findings from the investigation phase were put to the test. This phase was to run from begining of January 2010 till the end of February 2010.
- After this was the evaluation phase, in which the data collected from the testing phase was evaluated and analysed. The evaluation phase was due to end at the end of March 2010.
- The final phase was the concluding phase in which the report would be completed and handed in. The deadline for the report hand in being the 23rd of April.

These dates were never set in stone it was expected that some of the phases may over run their allotted time frames and the timescale was developed with this in mind. The investigation phase lasted longer than expected, research into the various methodologies was still continuing as the end of January approached. It was then vital to begin the testing phase. The decision was made to prioritise the testing phase and to wrap up the investigative research.

As of early February the testing methodology was beginning to take shape. The original decision to test using the university's equipment was quickly discarded due to worries about access to equipment meant that the author decided to build the test networks with his own equipment in his own home. It was around this time that the first major setback occurred. The machine allocated for use as the Linux router suffered a serious logic board failure. This put testing back another two weeks whilst a replacement was sourced.

As the end of February was nearing the author made the decision to overlap the testing and evaluation phases. Work began on documenting the investigation phase as the testing phase was still in progress. By early March all equipment had been sourced and the topologies were configured for testing. By the third week of March all data had been acquired and the testing phase was considered complete.

Moving into April the evaluation phase neared completion. And the project moved into the concluding phase. Work on this report was completed during the second week of April. Overall the project management has been adequate the report has been produced ahead of schedule even though the investigation and testing phases over ran their allotted time frames.

If the project was to be started again from scratch the only changes that would be made would be to find more accurate methods of CPU and memory comparison along with expanding the topologies to use multiple servers.

References & Bibliography.

ARBOR NETWORKS, (2009) *Arbor e30 Data Sheet* [Online].
http://www.arbornetworks.com/index.php?option=com_docman&task=doc_download&gid=354 - Last Accessed 18th March 2010

AUERBACH, Karl (2004) *Limitations of ICMP Echo for network measurement* [Online]
<http://www.iwl.com/white-papers/limitations-of-icmp-echo-for-network-measurement.html> - Last Accessed 23rd March 2010.

BALLIACHE, Leonardo (2003) *Differentiated Service on Linux HOWTO* [Online].
<http://www.opalsoft.net/qos/DS.htm> - Last Accessed 24th March 2010.

BLUE COAT, (2009) *Blue Coat Packetshaper Data Sheet* [Online].
<http://www.bluecoat.com/doc/direct/7941> - Last Accessed 18th March 2010

BLUE COAT, (2010). *Blue Coat Packetshaper* [Online].
<http://www.bluecoat.com/products/packetshaper> - Last Accessed 18th March 2010

BRIDLE, Chris (2009). *An investigation into quality of service and its effect on internet application traffic.* BSc (Hons). Computer Networking. Faculty of ACES. Sheffield Hallam University.

BROWN, Martin A. (2006). *Traffic Control HOWTO* [Online]. <http://linux-ip.net/articles/Traffic-Control-HOWTO/index.html> - Last Accessed 24th March 2010

CELLAN-JONES, Rory (2009). *Britons say broadband 'essential'*. [Online].
<http://news.bbc.co.uk/1/hi/technology/8079637.stm> - Last Accessed 17th April 2010.

CISCO, (2005). *Comparing Traffic Policing and Traffic Shaping for Bandwidth Limiting* [Online]
http://www.cisco.com/en/US/tech/tk543/tk545/technologies_tech_note09186a00800a3a25.shtml - Last Accessed 14th March 2010

CISCO, (2010). *Integrated Services - Introduction* [Online].
http://www.cisco.com/en/US/products/ps6611/products_ios_protocol_group_home.html - Last Accessed 11th March 2010

CISCO, (2010b). *Internetworking Design Basics* [Online].
<http://www.cisco.com/en/US/docs/internetworking/design/guide/nd2002.html#wp3380> - Last Accessed 14th March 2010

CISCO, (2010c). *Cisco IOS Quality of Service Solutions Configuration Guide, Release 12.2 - Policing and Shaping Overview*. [Online].
http://www.cisco.com/en/US/docs/ios/12_2/qos/configuration/guide/qcfcplsh.html - Last Accessed 21st March 2010

CISCO, (2010d). *Modular Quality of Service Command-Line Interface Overview*. [Online]
http://www9.cisco.com/en/US/docs/ios/12_2/qos/configuration/guide/qcfmdcli.html - Last Accessed 21st March 2010

CISCO, (2010e). *Configuring the Modular Quality of Service Command-Line Interface*. [Online]
http://www9.cisco.com/en/US/docs/ios/12_2/qos/configuration/guide/qcfmcli2.html -
Last Accessed 21st March 2010

CISCO, (2010f). *Configuring IP Access Lists* [Online].
http://www.cisco.com/en/US/products/sw/secursw/ps1018/products_tech_note09186a00800a5b9a.shtml - Last Accessed 22nd March 2010

CISCOHARDWAREMAINTENANCE.COM (2010) *SMARTnet Pricing for CISCO2911/K9* [Online].
<http://www.cisohardwaremaintenance.com/catalog/ciscosmartnetquote.php?smartnetfor=CISCO2911%2FK9&smb=0&smba=1&engineer=0&cover=8&response=0&length=1> -
Last Accessed 19th April 2010

CNET, (2010). *Cisco 2621 Ethernet, Fast Ethernet Router Specifications*. [Online].
http://reviews.cnet.com/routers/cisco-2621-ethernet-fast/4507-3319_7-112030.html?tag=rnav - Last Accessed 22nd March 2010

Dell (2010). *The Dell Online Store: Build Your System* [Online].
<http://configure.euro.dell.com/dellstore/config.aspx?c=uk&cs=ukbsdt1&kc=305&l=en&oc=SV1R510&s=bsd&sbc=poweredge-r510> - Last Accessed 19th April 2010

DEVERA, Martin (2002). *HTB Linux queuing discipline manual - user guide* [Online].
<http://luxik.cdi.cz/~devik/qos/htb/manual/userg.htm> - Last Accessed 24th March 2010

FERGUSON, Geoff & HUSTON, Paul (1998). *Quality of Service - Delivering QoS on the Internet and in Corporate Networks*. New York, John Wiley & Sons Inc

GHEORGHE, Lucian (2006). *Designing and Implementing Linux Firewalls and QoS using netfilter, iproute2, NAT, and L7-filter*. Birmingham, Packt Publishing.

GOOGLE (2010). *Cisco Systems Cisco 2621XM Router* [Online].
<http://www.google.co.uk/products/catalog?q=Cisco+2621+router&hl=en&cid=11218652734135080394&ei=GxvHS9aoBom5-QaGyuCrAg&sa=button&ved=0CAkQggwADgA#p> - Last accessed 15th April 2010

GOOGLE (2010b). *Cisco 2900 series router* [online].
<http://www.google.co.uk/products?q=cisco+2900+series+router&aq=f> - last accessed 19th April 2009

HARVARD, (Not Dated). *Glossary of terms* [Online].
<http://cyber.law.harvard.edu/readinessguide/glossary.html> - Last Accessed 6th March 2010.

HUBERT, Bert (2004). *Linux Advanced Routing & Traffic Control HOWTO* [Online]
<http://lartc.org/lartc.html> - Last Accessed 16th March 2010

IETF, (1980). *RFC 768 - User Datagram Protocol* [Online] <http://www.ietf.org/rfc/rfc768.txt>
- Last Accessed 6th March 2010

IETF, (1981). *RFC: 793 - Transmission Control Protocol*. [Online]
<http://tools.ietf.org/html/rfc793> - Last Accessed 6th March 2010

IETF, (1994). *RFC: 1633 - Integrated Services in the Internet Architecture: an Overview*. [Online] <http://tools.ietf.org/html/rfc1633> - Last Accessed 11th March 2010

IETF, (1997). *RFC: 2205 - Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification*. [Online] <http://datatracker.ietf.org/doc/rfc2205/> - Last Accessed 11th March 2010

IETF, (1998). *RFC: 2475 - An Architecture for Differentiated Services*. [Online] <http://www.ietf.org/rfc/rfc2475.txt> - Last accessed 14th March 2010

IPERF, (2008). *Iperf Home Page* [Online]. <http://iperf.sourceforge.net> - Last Accessed 6th March 2010.

ITJOBWATCH.COM (2010). *Cisco Jobs* [Online]. <http://www.itjobswatch.co.uk/jobs/uk/cisco.do> - Last Accessed 19th April 2010

ITJOBWATCH.COM (2010b). *Linux Jobs* [Online]. <http://www.itjobswatch.co.uk/jobs/uk/linux.do> - Last Accessed 19th April 2010

IXCHARIOT.COM, (2010). *Q-Check Data Sheet* [Online]. <http://www.ixchariot.com/products/datasheets/qcheck.html> - Last Accessed - 23rd March 2010

LINUX.DIE.NET, (not dated). *ping(8) - Linux man page* [Online] <http://linux.die.net/man/8/ping> - Last Accessed 6th March 2010

LINUX FOUNDATION, (2009). *IProute2* [Online] <http://www.linuxfoundation.org/collaborate/workgroups/networking/iproute2> - Last Accessed 24th March 2010

LINUX.ORG, (2008). *Linux Timeline*. [Online] http://www.linux.org/info/linux_timeline.html - Last Accessed 23rd March 2010

LYON, Gordon (2009). *The Official Nmap Project Guide to Network Discovery and Security Scanning - TCP/IP reference* [Online]. <http://nmap.org/book/tcpip-ref.html> - Last Accessed 25th March 2010

MIKROTIK, (not dated). *What is RouterOS?* [Online] http://www.mikrotik.com/pdf/what_is_routeros.pdf - Last Accessed 18th March 2010

RUDE, (2002) *RUDE & CRUDE* [Online]. <http://rude.sourceforge.net/> - Last Accessed 23rd March 2010

SZIGETI, Tim & HATTINGH, Christina (2005). *End-to-end QoS Network Design: Quality of service in LANs, WANs, and VPNs*: Indianapolis, Cisco Press.

Tanenbaum, Andrew. S. (2005). *Computer Networks*. 4th ed. New Jersey, Pearson Education International.

UBIK, Sven (2001). *Traffic shaping precision test*. [Online]
<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.72.5898&rep=rep1&type=pdf>
- Last Accessed March 7th 2010

WHITE, Steve P. (2006). *A Linux based LAN to internet traffic shaping and IP accounting solution*.
BSc (Hons) Faculty of ACES. Sheffield Hallam University.

List Of Figures & Tables.

Figures.

Fig.1.1	The Investigated Situation.
Fig.1.2	Controlling traffic at the building.
Fig.2.1	The UDP Header.
Fig.2.2	RSVP
Fig.2.3	The Leaky Bucket Algorithm.
Fig.2.4	The token bucket algorithm.
Fig.2.5	Comparing shaping and policing.
Fig.2.6	A Generic Queuing Example.
Fig.3.1	The Hierarchical Internetwork Design Model.
Fig.4.1	How GTS works.
Fig.4.2	A simple Network.
Fig.4.3	The hierarchical qdisc/class relationship on a Linux Interface.
Fig.4.4	Jitter
Fig.5.1	Topology one. With a single client.
Fig.5.2	Topology two. With two clients.
Fig.8.1	Proposed future testing topology.

Tables.

Table.6.1	The Expected UDP Packet Loss At Both Configured Shaping Levels.
Table.6.2	Effects Of Shaping On TCP Throughput.
Table.6.3	The Effects Of Traffic Shaping On UDP Throughput On The Cisco Platform.
Table.6.4	The Effects Of Traffic Shaping On UDP Throughput On The Linux Platform.
Table.6.5	The Effects Of Traffic Shaping On UDP Jitter On The Cisco Platform.
Table.6.6	The Effects Of Traffic Shaping On UDP Jitter On The Linux Platform.
Table.6.7	The Effects Of Traffic Shaping On UDP Packet Loss On The Cisco Platform.
Table.6.8	The Effects Of Traffic Shaping On UDP Packet Loss On The Linux Platform.
Table.6.9	The Effects Of Traffic Shaping On UDP Packet Delivery On The Cisco Platform.
Table.6.10	The Effects Of Traffic Shaping On UDP Packet Delivery On The Linux Platform.
Table.6.11:	The effect of shaping on CPU Usage.
Table.6.12:	The Effect of Shaping On Memory Usage.
Table.6.13:	The Effect Of Shaping on CPU Usage on the Cisco Platform.
Table.6.14:	The Effect Of Shaping on CPU Usage on the Linux Platform.
Table.6.15:	The Effect Of Shaping On Memory Usage On The Cisco Platform.
Table.6.13:	The Effect Of Shaping On Memory Usage On The Linux Platform.
Table.7.1:	A comparison of 5 year TCO for Cisco and Linux.

Appendices.

Appendix I – iPerf.

The information in this appendix is taken verbatim from: <http://openmaniak.com/iperf.php>

Iperf is a tool to measure the bandwidth and the quality of a network link. Jperf can be associated with Iperf to provide a graphical frontend written in Java.

The network link is delimited by two hosts running Iperf.

The quality of a link can be tested as follows:

- Latency (response time or RTT): can be measured with the ping command.
- Jitter (latency variation): can be measured with an Iperf UDP test.
- Datagram loss: can be measured with an Iperf UDP test.
-

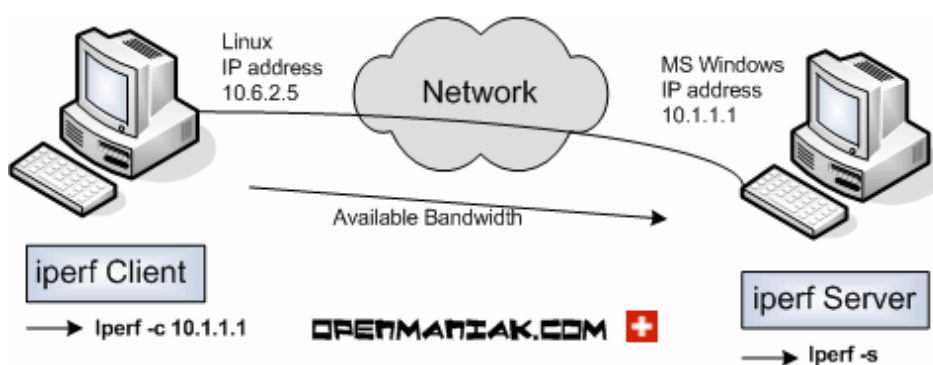
The bandwidth is measured through TCP tests.

To be clear, the difference between TCP (Transmission Control Protocol) and UDP (User Datagram Protocol) is that TCP use processes to check that the packets are correctly sent to the receiver whereas with UDP the packets are sent without any checks but with the advantage of being quicker than TCP.

Iperf uses the different capacities of TCP and UDP to provide statistics about network links.

Finally, Iperf can be installed very easily on any UNIX/Linux or Microsoft Windows system. One host must be set as client, the other one as server.

Here is a diagram where Iperf is installed on a Linux and Microsoft Windows machine. Linux is used as the Iperf client and Windows as the Iperf server. Of course, it is also possible to use two Linux boxes.



Iperf tests:

no arg.	Default	settings	-p,	-t,	-i	Port, timing and interval
-b	Data	format	-u,		-b	UDP tests, bandwidth settings
-r	Bi-directional	bandwidth	-m			Maximum Segment Size display
-d	Simultaneous	bi-directional	-M			Maximum Segment Size settings
	bandwidth		-P			Parallel tests
-w	TCP Window size		-h			help

Appendix II – Q-Check.

*Information in this appendix is taken verbatim from:
<http://www.ixchariot.com/products/datasheets/qcheck.html>*

What's good-looking, powerful, improves your quality of life and is FREE? Qcheck, the network troubleshooting utility from Ixia, quickly checks network response time, throughput, and streaming performance. It even runs anywhere-to-anywhere traceroute!

Install Qcheck on the computer you'll use to run the Qcheck console. Qcheck runs on any computer running Windows 2000, NT or XP. Install QCheck on the computer you'll use to run the Qcheck console. Qcheck runs on any computer running Windows 2000, NT or XP

Install an Ixia Performance Endpoint (or Qcheck itself, which has the endpoint built in) on any other computers. You can download Performance Endpoint software free from Ixial!

Qcheck is built using IxChariot technology. IxChariot is a powerful network assessment tool which is used to test networks and Wireless Devices. The following chart compares the benefits of IxChariot and Qcheck over Ping

Become a network testing expert using IxChariot. Attend a free webinar [here](#).

Qcheck vs. Ping and IxChariot

	Ping	Qcheck	IxChariot
Measures response time of IP networks	✓	✓	✓
Simulates real application flows across the network to test connectivity and performance.		✓	✓
Gauge network throughput with different traffic types (TCP, UDP, RTP)		✓	✓
Runs traceroute between any two workstations on your network, regardless of their locations.		✓	✓
Determines at what rate streaming traffic is received and how much packet loss occurs.		✓	✓
Troubleshoot wireless performance problems by comparing performance to signal strength (RSSI) and CPU utilization.		✓	✓
Tests a network using multiple application flows for Voice, Video and Data Simultaneously			✓
Scale applications tests across 10's, 100's or 1000's of hosts			✓
Tests the effect of Virtualization on network Services			✓
Measure one-way delay, MOS, and MDI to ensure successful rollouts of IP voice and video services			✓

That's it! Based on the parameters you select, the Qcheck console will instruct any two endpoints to run a test and return the results to you at the Qcheck console.

- For a **response time** test, Qcheck returns the minimum, maximum and average number of seconds it took to complete a transaction.
- For a **throughput** test, Qcheck returns the amount of data per second that was successfully sent between the two endpoints.
- For a **streaming** test, Qcheck returns the rate at which the streaming data was received by the second endpoint and the amount of packet loss that occurred.
- For a **traceroute** test, Qcheck returns the number of hops, average hop latency, and the address and names of the host at each hop.

Key Features

- For a response time test, Qcheck returns the minimum, maximum and average number of seconds it took to complete a transaction
- For a throughput test, Qcheck returns the amount of data per second that was successfully sent between the two endpoints
- For a streaming test, Qcheck returns the rate at which the streaming data was received by the second endpoint and the amount of packet loss that occurred
- For a traceroute test, Qcheck returns the number of hops, average hop latency, and the address and names of the host at each hop

Qcheck and Traceroute

Qcheck's anywhere-to-anywhere traceroute function sets you free! You can now originate a traceroute from any user's workstation on the network that is running a Performance Endpoint. Run anywhere-to-anywhere traceroute from our most popular Performance Endpoints to any other TCP/IP connection. The receiving end in a traceroute test doesn't even need to have endpoint software installed! Setting up a traceroute is easy:

- Press the Traceroute button and select a protocol (TCP OR UDP)
- Identify the IP addresses of the two endpoints you're targeting (Endpoint 1 must be running a Performance Endpoint)
- Press the Run button
- Qcheck runs the traceroute and reports the results, the sequence of hops, the hop latency in ms, and the name of each hop location.

Your Problem

Someone in accounting calls the Help Desk saying he can't access the database server.

I've got a lot of remote employees connected to my network by 56 Kbps dial-up modems. I wonder what kind of throughput they see.

The reception from the company's videoconferencing system is lousy.

You've detected a slow connection between New York & San Francisco but you're in Houston. How do you isolate the problem?

The Qcheck Solution

A Qcheck response time test determines if this is a network connectivity problem or not. Qcheck can also determine if this is a problem being experienced by one user, one department, or many employees.

A Qcheck throughput test indicates how quickly a computer can transmit data across any network. And, from your desk, you can drive Qcheck tests between any two computers on your network.

A Qcheck streaming test evaluates the network's ability to support multimedia traffic, letting you know the rate at which traffic is received and how many packets get lost along the way.

A Qcheck on-demand traceroute initiates a traceroute test between any two workstations on your network, regardless of their location.

Appendix III – iperf configurations.

On the iperf host:

For TCP throughput:

```
iperf -c server.ip.add.ress -t 120 -i 5
```

or UDP Throughput:

At 512kbps bitstream.

```
iperf -c server.ip.add.ress -u -b 512k -t 120 -i 5
```

At 1Mbps bitstream.

```
iperf -c server.ip.add.ress -u -b 1m -t 120 -i 5
```

At 2Mbps bitstream.

```
iperf -c server.ip.add.ress -u -b 2m -t 120 -i 5
```

At 10Mbps bitstream.

```
iperf -c server.ip.add.ress -u -b 10m -t 120 -i 5
```

At 20Mbps bitstream

```
iperf -c server.ip.add.ress -u -b 20m -t 120 -i 5
```

On the iperf server:

To set machine as a server:

```
Iperf -s
```

To set machine as a UDP server:

```
Iperf -s -u
```

To set machine to serve on a different port:

```
Iperf -s -p 5002
```

Appendix IV – Linux Traffic Shaping Scripts.

Scenario A.

Host's upload and download speed are shaped at 1Mbps.

To limit download speed.

```
sudo tc qdisc add dev eth1 root handle 1: htb
sudo tc class add dev eth1 parent 1:0 classid 1:1 htb rate 100Mbit
sudo tc class add dev eth1 parent 1:1 classid 1:10 htb rate 1Mbit
sudo tc qdisc add dev eth1 parent 1:10 sfq quantum 1514b perturb 15
sudo tc filter add dev eth1 protocol ip parent 1:0 prio 5 u32 match ip
dst 192.168.0.2 flowid 1:10
```

To limit upload speed.

```
sudo tc qdisc add dev eth0 root handle 2: htb
sudo tc class add dev eth0 parent 2:0 classid 2:1 htb rate 100Mbit
sudo tc class add dev eth0 parent 2:1 classid 2:10 htb rate 1Mbit
sudo tc qdisc add dev eth0 parent 2:10 sfq quantum 1514b perturb 15
sudo tc filter add dev eth0 protocol ip parent 2:0 prio 5 u32 match ip
src 192.168.0.2 flowid 2:10
```

Scenario B.

Host's upload and download speed are shaped at 10Mbps.

To limit download speed.

```
sudo tc qdisc add dev eth1 root handle 1: htb
sudo tc class add dev eth1 parent 1:0 classid 1:1 htb rate 100Mbit
sudo tc class add dev eth1 parent 1:1 classid 1:10 htb rate 10Mbit
sudo tc qdisc add dev eth1 parent 1:10 sfq quantum 1514b perturb 15
sudo tc filter add dev eth1 protocol ip parent 1:0 prio 5 u32 match ip
dst 192.168.0.2 flowid 1:10
```

To limit upload speed.

```
sudo tc qdisc add dev eth0 root handle 2: htb
sudo tc class add dev eth0 parent 2:0 classid 2:1 htb rate 100Mbit
sudo tc class add dev eth0 parent 2:1 classid 2:10 htb rate 10Mbit
sudo tc qdisc add dev eth0 parent 2:10 sfq quantum 1514b perturb 15
sudo tc filter add dev eth0 protocol ip parent 2:0 prio 5 u32 match ip
src 192.168.0.2 flowid 2:10
```

Scenario C.

Host has an asynchronous connection. Download speed is shaped at 10Mbps upload speed is shaped at 1Mbps.

To limit download speed.

```
sudo tc qdisc add dev eth1 root handle 1: htb
sudo tc class add dev eth1 parent 1:0 classid 1:1 htb rate 100Mbit
sudo tc class add dev eth1 parent 1:1 classid 1:10 htb rate 10Mbit
sudo tc qdisc add dev eth1 parent 1:10 sfq quantum 1514b perturb 15
sudo tc filter add dev eth1 protocol ip parent 1:0 prio 5 u32 match ip
dst 192.168.0.2 flowid 1:10
```

To limit upload speed.

```
sudo tc qdisc add dev eth0 root handle 2: htb
sudo tc class add dev eth0 parent 2:0 classid 2:1 htb rate 100Mbit
sudo tc class add dev eth0 parent 2:1 classid 2:10 htb rate 1Mbit
sudo tc qdisc add dev eth0 parent 2:10 sfq quantum 1514b perturb 15
```

```
sudo tc filter add dev eth0 protocol ip parent 2:0 prio 5 u32 match ip
src 192.168.0.2 flowid 2:10
```

Scenario D.

A second host is added to the topology. Both hosts are shaped at 10Mbps synchronous.

To limit download speed.

```
sudo tc qdisc add dev eth1 root handle 1: htb
sudo tc class add dev eth1 parent 1:0 classid 1:1 htb rate 100Mbit
sudo tc class add dev eth1 parent 1:1 classid 1:10 htb rate 10Mbit
sudo tc qdisc add dev eth1 parent 1:10 sfq quantum 1514b perturb 15
sudo tc filter add dev eth1 protocol ip parent 1:0 prio 5 u32 match ip
dst 192.168.0.2 flowid 1:10
sudo tc class add dev eth1 parent 1:1 classid 1:20 htb rate 10Mbit
sudo tc qdisc add dev eth1 parent 1:20 sfq quantum 1514b perturb 15
sudo tc filter add dev eth1 protocol ip parent 1:0 prio 5 u32 match ip
dst 192.168.0.3 flowid 1:20
```

To limit upload speed.

```
sudo tc qdisc add dev eth0 root handle 2: htb
sudo tc class add dev eth0 parent 2:0 classid 2:1 htb rate 100Mbit
sudo tc class add dev eth0 parent 2:1 classid 2:10 htb rate 10Mbit
sudo tc qdisc add dev eth0 parent 2:10 sfq quantum 1514b perturb 15
sudo tc filter add dev eth0 protocol ip parent 2:0 prio 5 u32 match ip
src 192.168.0.2 flowid 2:10
sudo tc class add dev eth0 parent 2:1 classid 2:20 htb rate 10Mbit
sudo tc qdisc add dev eth0 parent 2:20 sfq quantum 1514b perturb 15
sudo tc filter add dev eth0 protocol ip parent 2:0 prio 5 u32 match
ip src 192.168.0.3 flowid 2:20
```

Scenario E.

Both host's are shaped differently. Host 1 on 1Mbps Synchronous. Host 2 on 10Mbps Synchronous.

To limit download speed

```
sudo tc qdisc add dev eth1 root handle 1: htb
sudo tc class add dev eth1 parent 1:0 classid 1:1 htb rate 100Mbit
sudo tc class add dev eth1 parent 1:1 classid 1:10 htb rate 1Mbit
sudo tc qdisc add dev eth1 parent 1:10 sfq quantum 1514b perturb 15
sudo tc filter add dev eth1 protocol ip parent 1:0 prio 5 u32 match ip
dst 192.168.0.2 flowid 1:10
sudo tc class add dev eth1 parent 1:1 classid 1:20 htb rate 10Mbit
sudo tc qdisc add dev eth1 parent 1:20 sfq quantum 1514b perturb 15
sudo tc filter add dev eth1 protocol ip parent 1:0 prio 5 u32 match ip
dst 192.168.0.3 flowid 1:20
```

To limit upload speed

```
sudo tc qdisc add dev eth0 root handle 2: htb
sudo tc class add dev eth0 parent 2:0 classid 2:1 htb rate 100Mbit
sudo tc class add dev eth0 parent 2:1 classid 2:10 htb rate 1Mbit
sudo tc qdisc add dev eth0 parent 2:10 sfq quantum 1514b perturb 15
sudo tc filter add dev eth0 protocol ip parent 2:0 prio 5 u32 match ip
src 192.168.0.2 flowid 2:10
sudo tc class add dev eth0 parent 2:1 classid 2:20 htb rate 10Mbit
sudo tc qdisc add dev eth0 parent 2:20 sfq quantum 1514b perturb 15
sudo tc filter add dev eth0 protocol ip parent 2:0 prio 5 u32 match ip
src 192.168.0.3 flowid 2:20
```

Appendix V – Cisco Traffic Shaping Scripts.

Scenario A.

Host's upload and download speed are shaped at 1Mbps.

```
en
!
conf t
!
interface fa0/0
ip address 10.0.0.1 255.255.255.0
no shutdown
!
interface fa0/1
ip address 192.168.0.1 255.255.255.0
no shutdown
!
access-list 101 permit ip any 192.168.0.2 0.0.0.0
!
!
class-map match-all incoming_to_192.168.0.2
match access-group 101
!
policy-map shape_incoming_to_192.168.0.2
class incoming_to_192.168.0.2
shape average 1000000
!
interface fa0/1
service-policy output shape_incoming_to_192.168.0.2
!
access-list 102 permit ip 192.168.0.2 0.0.0.0 any
!
class-map match-all outgoing_from_192.168.0.2
match access-group 102
!
policy-map shape_outgoing_from_192.168.0.2
class outgoing_from_192.168.0.2
shape average 1000000
!
interface fa0/0
service-policy output shape_outgoing_from_192.168.0.2
!
```

Scenario B.

Host's upload and download speed are shaped at 10Mbps.

```
en
!
conf t
!
interface fa0/0
ip address 10.0.0.1 255.255.255.0
no shut
!
interface fa0/1
ip address 192.168.0.1 255.255.255.0
no shut
```

```

!
access-list 101 permit ip any 192.168.0.2 0.0.0.0
!
!
class-map match-all incoming_to_192.168.0.2
match access-group 101
!
policy-map shape_incoming_to_192.168.0.2
class incoming_to_192.168.0.2
shape average 10000000
!
interface fa0/1
service-policy output shape_incoming_to_192.168.0.2
!
access-list 102 permit ip 192.168.0.2 0.0.0.0 any
!
class-map match-all outgoing_from_192.168.0.2
match access-group 102
!
policy-map shape_outgoing_from_192.168.0.2
class outgoing_from_192.168.0.2
shape average 10000000
!
interface fa0/0
service-policy output shape_outgoing_from_192.168.0.2
!

```

Scenario C.

Host has an asynchronous connection. Download speed is shaped at 10Mbps upload speed is shaped at 1Mbps.

```

en
!
conf t
!
interface fa0/0
ip address 10.0.0.1 255.255.255.0
no shut
!
interface fa0/1
ip address 192.168.0.1 255.255.255.0
no shut
!
access-list 101 permit ip any 192.168.0.2 0.0.0.0
!
!
class-map match-all incoming_to_192.168.0.2
match access-group 101
!
policy-map shape_incoming_to_192.168.0.2
class incoming_to_192.168.0.2
shape average 10000000
!
interface fa0/1
service-policy output shape_incoming_to_192.168.0.2
!
access-list 102 permit ip 192.168.0.2 0.0.0.0 any
!
class-map match-all outgoing_from_192.168.0.2
match access-group 102

```

```
!  
policy-map shape_outgoing_from_192.168.0.2  
class outgoing_from_192.168.0.2  
shape average 1000000  
!  
interface fa0/0  
service-policy output shape_outgoing_from_192.168.0.2  
!
```

Scenario D.

A second host is added to the topology. Both hosts are shaped at 10Mbps synchronous.

```
en  
!  
conf t  
!  
interface fa0/0  
ip address 10.0.0.1 255.255.255.0  
no shut  
!  
interface fa0/1  
ip address 192.168.0.1 255.255.255.0  
no shut  
!  
access-list 101 permit ip any 192.168.0.2 0.0.0.0  
!  
class-map match-all incoming_to_192.168.0.2  
match access-group 101  
!  
access-list 103 permit ip any 192.168.0.3 0.0.0.0  
!  
class-map match-all incoming_to_192.168.0.3  
match access-group 103  
!  
policy-map shape_incoming_traffic  
class incoming_to_192.168.0.2  
shape average 10000000  
class incoming_to_192.168.0.3  
shape average 10000000  
!  
interface fa0/1  
service-policy output shape_incoming_traffic  
!  
access-list 102 permit ip 192.168.0.2 0.0.0.0 any  
!  
class-map match-all outgoing_from_192.168.0.2  
match access-group 102  
!  
access-list 104 permit ip 192.168.0.3 0.0.0.0 any  
!  
class-map match-all outgoing_from_192.168.0.3  
match access-group 104  
!  
policy-map shape_outgoing_traffic  
class outgoing_from_192.168.0.2  
shape average 10000000  
class outgoing_from_192.168.0.3  
shape average 10000000  
!  
interface fa0/0
```

```
service-policy output shape_outgoing_traffic
```

Scenario E.

Both hosts' are shaped differently. Host 1 on 1Mbps Synchronous. Host 2 on 10Mbps Synchronous.

```
en
!
conf t
!
interface fa0/0
ip address 10.0.0.1 255.255.255.0
no shut
!
interface fa0/1
ip address 192.168.0.1 255.255.255.0
no shut
!
access-list 101 permit ip any 192.168.0.2 0.0.0.0
!
class-map match-all incoming_to_192.168.0.2
match access-group 101
!
access-list 103 permit ip any 192.168.0.3 0.0.0.0
!
class-map match-all incoming_to_192.168.0.3
match access-group 103
!
policy-map shape_incoming_traffic
class incoming_to_192.168.0.2
shape average 1000000
class incoming_to_192.168.0.3
shape average 10000000
!
interface fa0/1
service-policy output shape_incoming_traffic
!
access-list 102 permit ip 192.168.0.2 0.0.0.0 any
!
class-map match-all outgoing_from_192.168.0.2
match access-group 102
!
access-list 104 permit ip 192.168.0.3 0.0.0.0 any
!
class-map match-all outgoing_from_192.168.0.3
match access-group 104
!
policy-map shape_outgoing_traffic
class outgoing_from_192.168.0.2
shape average 1000000
class outgoing_from_192.168.0.3
shape average 10000000
!
interface fa0/0
service-policy output shape_outgoing_traffic
```

Appendix VI – Linux tc shaping methods

Method one - Shaping takes place at 1st child class enforcing all traffic to same limit if filtered. Using CBQ.

```
# Inbound eth1 shaping to 1Mbps
# Set the root qdisc
tc qdisc add dev eth1 root handle 1: cbq avpkt 1000 bandwidth 100mbit
```

```
# Set the 1st child class
tc class add dev eth1 parent 1: classid 1:1 cbq rate 1mbit allot 1500
prio 5 bounded isolated
```

```
# Set the filter
tc filter add dev eth1 parent 1: protocol ip prio 16 u32 match ip dst
192.168.0.2 flowid 1:1
```

```
# Outbound eth0 shaping to 1Mbps
# Set the root qdisc
tc qdisc add dev eth0 root handle 1: cbq avpkt 1000 bandwidth 100mbit
```

```
# Set the 1st child class
tc class add dev eth0 parent 1: classid 1:1 cbq rate 1mbit allot 1500
prio 5 bounded isolated
```

```
# Set the filter
tc filter add dev eth0 parent 1: protocol ip prio 16 u32 match ip dst
10.0.0.2 flowid 1:1
```

Method Two - Shaping takes place at 2nd child class level so multiple levels of shaping can be applied. Uses CBQ

```
# Inbound eth1 shaping to 1Mbps
# Set the root qdisc
tc qdisc add dev eth1 root handle 10: cbq bandwidth 100Mbit avpkt 1000
```

```
# Set the 1st child class
tc class add dev eth1 parent 10:0 classid 10:10 cbq bandwidth 100Mbit
rate 100Mbit allot 1514 weight 10Mbit prio 5 maxburst 20 avpkt 1000
bounded
```

```
#Set the 2nd child class - to shape to 1Mbps
tc class add dev eth1 parent 10:10 classid 10:100 cbq bandwidth 100Mbit
rate 1Mbit allot 1514 weight 128Kbit prio 5 maxburst 20 avpkt 1000
bounded
```

```
#set the child qdisc for this class
tc qdisc add dev eth1 parent 10:100 sfq quantum 1514b perturb 15
```

```
# Set the child filter for the qdisc
tc filter add dev eth1 parent 10:0 protocol ip prio 5 u32 match ip dst
192.168.0.2 flowid 10:100
```

Method 3 using HTB.

```
#Inbound on eth1 - shaping to 1Mbps
```

```
# Set root qdisc
```

```
tc qdisc add dev eth1 root handle 1: htb default 30
```

```
# Set 1st child class
```

```
tc class add dev eth1 parent 1: classid 1:1 htb rate 100mbit burst 15k
```

```
# Set 2nd Child Class
```

```
tc class add dev eth1 parent 1:1 classid 1:10 htb rate 1mbit ceil  
100mbit burst 15k
```

```
# Set Child qdisc
```

```
tc qdisc add dev eth1 parent 1:10 handle 10: sfq perturb 10
```

```
# Set filter for qdisc
```

```
tc filter add dev eth1 parent 1: protocol ip prio 16 u32 match ip dst  
192.168.0.2 flowid 1:1
```

OR

```
# To limit download speed.
```

```
tc qdisc add dev eth1 root handle 1: htb
```

```
tc class add dev eth1 parent 1:0 classid 1:1 htb rate 100Mbit
```

```
tc class add dev eth1 parent 1:1 classid 1:10 htb rate 1Mbit
```

```
tc qdisc add dev eth1 parent 1:10 sfq quantum 1514b perturb 15
```

```
tc filter add dev eth1 protocol ip parent 1:0 prio 5 u32 match ip dst  
192.168.0.2 flowid 1:10
```

```
# To limit upload speed.
```

```
tc qdisc add dev eth0 root handle 2: htb
```

```
tc class add dev eth0 parent 2:0 classid 2:1 htb rate 100Mbit
```

```
tc class add dev eth0 parent 2:1 classid 2:10 htb rate 1Mbit
```

```
tc qdisc add dev eth0 parent 2:10 sfq quantum 1514b perturb 15
```

```
tc filter add dev eth0 protocol ip parent 2:0 prio 5 u32 match ip dst  
10.0.0.2 flowid 2:10
```

```
#to confirm config on interface
```

```
sudo tc class show dev eth1
```

```
# to delete config from interface
```

```
sudo tc qdisc del root dev eth1
```

Appendix VII – Raw Data From The Cisco Platform.

Results For Topology One Baseline.

:
PC-A Download:

	TCP Throughput (Mbps)	UDP Throughput (Mbps)	Jitter (ms)	Packets Lost (%)	Packets Arrived Out of Order	CPU Usage (%)	Memory Utilised (%)
TCP Stream	48.8	n/a	n/a	n/a	n/a	25	19.9
UDP @ 512kbps	n/a	0.512	0	0.000	0	1	19.9
UDP @ 1Mbps	n/a	1.000	0	0.000	0	1	19.9
UDP @ 2Mbps	n/a	2.000	0	0.005	0	2	19.9
UDP @ 10Mbps	n/a	10.000	0	0.000	0	13	19.9
UDP @ 20 Mbps	n/a	20.000	0	0.002	0	27	19.9

Latency = 1ms

PC-A Upload:

	TCP Throughput (Mbps)	UDP Throughput (Mbps)	Jitter	Packets Lost	Packets Arrived Out of Order	CPU Usage (%)	Memory Utilised (%)
TCP Stream	49.4	n/a	n/a	n/a	n/a	53	19.9
UDP @ 512kbps	n/a	0.512	0.000	0	0	2	19.9
UDP @ 1Mbps	n/a	1.000	0.000	0	0	1	19.9
UDP @ 2Mbps	n/a	2.000	0.000	0	0	3	19.9
UDP @ 10Mbps	n/a	10.000	3.006	0	0	13	19.9
UDP @ 20 Mbps	n/a	20.000	1.493	0	0	27	19.9

Latency = 1 ms

Results For Scenario A.

PC-A Download – Shaped at 1Mbps:

	TCP Throughput (Mbps)	UDP Throughput (Mbps)	Jitter (ms)	Packets Lost (%)	Packets Arrived Out of Order	CPU Usage (%)	Memory Utilised (%)
TCP Stream	0.962	n/a	n/a	n/a	n/a	0	20.4
UDP @ 512kbps	n/a	0.512	0.000	0.0	0	1	20.4
UDP @ 1Mbps	n/a	0.972	13.050	2.1	0	2	20.4
UDP @ 2Mbps	n/a	0.972	3.362	51.0	0	5	20.4
UDP @ 10Mbps	n/a	0.972	4.506	90.0	0	23	20.4
UDP @ 20 Mbps	n/a	0.972	6.952	95.0	0	45	20.4

Latency = 1 ms

PC-A Upload -shaped at 1Mbps:

	TCP Throughput (Mbps)	UDP Throughput (Mbps)	Jitter (ms)	Packets Lost (%)	Packets Arrived Out of Order	CPU Usage (%)	Memory Utilised (%)
TCP Stream	0.962	n/a	n/a	n/a	n/a	0	20.4
UDP @ 512kbps	n/a	0.512	3.139	0.0	0	1	20.4
UDP @ 1Mbps	n/a	0.972	12.830	2.1	0	2	20.4
UDP @ 2Mbps	n/a	0.972	5.485	51.0	0	5	20.4
UDP @ 10Mbps	n/a	0.972	4.532	90.0	0	22	20.4
UDP @ 20 Mbps	n/a	0.972	5.735	95.0	0	45	20.4

Latency = 1 ms

Results For Scenario B.

PC-A Download – Shaped at 10Mbps:

	TCP Throughput (Mbps)	UDP Throughput (Mbps)	Jitter (ms)	Packets Lost (%)	Packets Arrived Out of Order	CPU Usage (%)	Memory Utilised (%)
TCP Stream	9.62	n/a	n/a	n/a	n/a	30	20.7
UDP @ 512kbps	n/a	0.512	0.000	0.0	0	1	20.7
UDP @ 1Mbps	n/a	1.000	0.000	0.0	0	2	20.7
UDP @ 2Mbps	n/a	2.000	0.000	0.0	0	4	20.7
UDP @ 10Mbps	n/a	9.700	2.244	2.9	0	29	20.7
UDP @ 20 Mbps	n/a	9.720	2.312	51.0	0	54	20.7

Latency = 1 ms

PC-A Upload – Shaped at 10Mbps:

	TCP Throughput (Mbps)	UDP Throughput (Mbps)	Jitter	Packets Lost	Packets Arrived Out of Order	CPU Usage (%)	Memory Utilised (%)
TCP Stream	9.62	n/a	n/a	n/a	n/a	27	20.7
UDP @ 512kbps	n/a	0.512	0.000	0.0	0	1	20.7
UDP @ 1Mbps	n/a	1.000	0.000	0.0	0	2	20.7
UDP @ 2Mbps	n/a	2.000	0.000	0.0	0	4	20.7
UDP @ 10Mbps	n/a	9.710	0.994	2.8	0	29	20.7
UDP @ 20 Mbps	n/a	9.720	3.282	51.0	0	54	20.7

Latency = 1 ms

Results For Scenario C.

PC-A Download – Shaped at 10Mbps:

	TCP Throughput (Mbps)	UDP Throughput (Mbps)	Jitter (ms)	Packets Lost (%)	Packets Arrived Out of Order	CPU Usage (%)	Memory Utilised (%)
TCP Stream	9.62	n/a	n/a	n/a	n/a	29	20.7
UDP @ 512kbps	n/a	0.512	0.000	0.0	0	2	20.7
UDP @ 1Mbps	n/a	1.000	0.000	0.0	0	2	20.7
UDP @ 2Mbps	n/a	2.000	0.000	0.0	0	4	20.7
UDP @ 10Mbps	n/a	9.700	3.404	2.9	0	29	20.7
UDP @ 20 Mbps	n/a	9.720	3.779	51.0	0	54	20.7

Latency = 1 ms

PC-A Upload – Shaped at 1Mbps:

	TCP Throughput (Mbps)	UDP Throughput (Mbps)	Jitter (ms)	Packets Lost (%)	Packets Arrived Out of Order	CPU Usage (%)	Memory Utilised (%)
TCP Stream	0.962	n/a	n/a	n/a	n/a	3	20.7
UDP @ 512kbps	n/a	0.521	0.000	0.0	0	1	20.7
UDP @ 1Mbps	n/a	0.972	12.282	2.1	0	2	20.7
UDP @ 2Mbps	n/a	0.972	4.681	51.0	0	5	20.7
UDP @ 10Mbps	n/a	0.972	5.442	90.0	0	22	20.7
UDP @ 20 Mbps	n/a	0.972	7.215	95.0	0	45	20.7

Latency = 1 ms

Results For The Topology Two Baseline.

PC-A Download:

	TCP Throughput (Mbps)	UDP Throughput (Mbps)	Jitter (ms)	Packets Lost (%)	Packets Arrived Out of Order	CPU Usage (%)	Memory Utilised (%)
TCP Stream	27.7	n/a	n/a	n/a	n/a	28	20.1
UDP @ 512kbps	n/a	0.465	0.000	0.0	0	1	20.1
UDP @ 1Mbps	n/a	0.913	6.230	0.0	0	2	20.1
UDP @ 2Mbps	n/a	1.630	1.844	0.1	0	4	20.1
UDP @ 10Mbps	n/a	7.250	0.000	0.0	0	17	20.1
UDP @ 20 Mbps	n/a	13.400	0.000	0.0	0	31	20.1

Latency = 1 ms

PC-A Upload:

	TCP Throughput (Mbps)	UDP Throughput (Mbps)	Jitter (ms)	Packets Lost (%)	Packets Arrived Out of Order	CPU Usage (%)	Memory Utilised (%)
TCP Stream	29.1	n/a	n/a	n/a	n/a	14	20.1
UDP @ 512kbps	n/a	0.512	0	0.000	0	2	20.1
UDP @ 1Mbps	n/a	1.000	0	0.000	0	2	20.1
UDP @ 2Mbps	n/a	2.000	0	0.000	0	5	20.1
UDP @ 10Mbps	n/a	9.990	0	0.068	0	27	20.1
UDP @ 20 Mbps	n/a	20.000	0	0.001	0	54	20.1

Latency = 1 ms

PC-B Download:

	TCP Throughput (Mbps)	UDP Throughput (Mbps)	Jitter (ms)	Packets Lost (%)	Packets Arrived Out of Order	CPU Usage (%)	Memory Utilised (%)
TCP Stream	23.6	n/a	n/a	n/a	n/a	28	20.1
UDP @ 512kbps	n/a	0.332	0.023	0	0	1	20.1
UDP @ 1Mbps	n/a	0.843	0.000	0	0	2	20.1
UDP @ 2Mbps	n/a	1.800	0.000	0	0	4	20.1
UDP @ 10Mbps	n/a	6.150	0.037	0	0	17	20.1
UDP @ 20 Mbps	n/a	11.100	1.573	0	0	31	20.1

Latency = 1 ms

PC-B Upload:

	TCP Throughput (Mbps)	UDP Throughput (Mbps)	Jitter (ms)	Packets Lost (%)	Packets Arrived Out of Order	CPU Usage (%)	Memory Utilised (%)
TCP Stream	24.2	n/a	n/a	n/a	n/a	14	20.1
UDP @ 512kbps	n/a	0.512	0.000	0.000	0	2	20.1
UDP @ 1Mbps	n/a	1.000	0.000	0.000	0	2	20.1
UDP @ 2Mbps	n/a	2.000	0.000	0.000	0	5	20.1
UDP @ 10Mbps	n/a	9.990	1.077	0.026	0	27	20.1
UDP @ 20 Mbps	n/a	20.000	0.000	0.002	0	54	20.1

Latency = 1 ms

Results For Scenario D.

PC-A Download – Shaped at 10Mbps:

	TCP Throughput (Mbps)	UDP Throughput (Mbps)	Jitter (ms)	Packets Lost (%)	Packets Arrived Out of Order	CPU Usage (%)	Memory Utilised (%)
TCP Stream	9.62	n/a	n/a	n/a	n/a	60	21.3
UDP @ 512kbps	n/a	0.509	0.000	0.000	0	1	21.3
UDP @ 1Mbps	n/a	0.945	5.296	0.000	0	3	21.3
UDP @ 2Mbps	n/a	1.820	0.000	0.180	0	6	21.3
UDP @ 10Mbps	n/a	7.360	3.224	0.056	0	27	21.3
UDP @ 20 Mbps	n/a	9.720	2.688	29.000	0	72	21.3

Latency = 1 ms

PC-A Upload – Shaped at 10Mbps:

	TCP Throughput (Mbps)	UDP Throughput (Mbps)	Jitter (ms)	Packets Lost (%)	Packets Arrived Out of Order	CPU Usage (%)	Memory Utilised (%)
TCP Stream	9.56	n/a	n/a	n/a	n/a	59	21.3
UDP @ 512kbps	n/a	0.512	0.000	0.0	0	3	21.3
UDP @ 1Mbps	n/a	1.000	0.000	0.0	0	4	21.3
UDP @ 2Mbps	n/a	2.000	0.000	0.0	0	10	21.3
UDP @ 10Mbps	n/a	9.720	3.416	2.7	0	63	21.3
UDP @ 20 Mbps	n/a	9.720	2.424	51.0	0	14	21.3

Latency = 1 ms

PC-B Download – Shaped at 10Mbps:

	TCP Throughput (Mbps)	UDP Throughput (Mbps)	Jitter (ms)	Packets Lost (%)	Packets Arrived Out of Order	CPU Usage (%)	Memory Utilised (%)
TCP Stream	9.62	n/a	n/a	n/a	n/a	60	21.3
UDP @ 512kbps	n/a	0.255	0.992	0.000	0	1	21.3
UDP @ 1Mbps	n/a	0.989	0.000	0.000	0	3	21.3
UDP @ 2Mbps	n/a	1.560	4.304	0.000	0	6	21.3
UDP @ 10Mbps	n/a	5.900	0.003	0.060	0	27	21.3
UDP @ 20 Mbps	n/a	9.620	3.076	11.000	0	72	21.3

Latency = 1 ms

PC-B Upload – Shaped at 10Mbps:

	TCP Throughput (Mbps)	UDP Throughput (Mbps)	Jitter (ms)	Packets Lost (%)	Packets Arrived Out of Order	CPU Usage (%)	Memory Utilised (%)
TCP Stream	9.62	n/a	n/a	n/a	n/a	59	21.3
UDP @ 512kbps	n/a	0.512	0.000	0.0	0	3	21.3
UDP @ 1Mbps	n/a	1.000	0.134	0.0	0	4	21.3
UDP @ 2Mbps	n/a	2.000	0.000	0.0	0	10	21.3
UDP @ 10Mbps	n/a	9.720	3.051	2.7	0	63	21.3
UDP @ 20 Mbps	n/a	9.710	11.422	51.0	0	14	21.3

Latency = 1 ms

Results For Scenario E.

PC-A Download – Shaped at 1Mbps:

	TCP Throughput (Mbps)	UDP Throughput (Mbps)	Jitter (ms)	Packets Lost (%)	Packets Arrived Out of Order	CPU Usage (%)	Memory Utilised (%)
TCP Stream	0.962	n/a	n/a	n/a	n/a	28	20.9
UDP @ 512kbps	n/a	0.392	56.117	2.5	0	2	20.9
UDP @ 1Mbps	n/a	0.900	5.751	1.1	0	3	20.9
UDP @ 2Mbps	n/a	0.957	5.339	48.0	0	7	20.9
UDP @ 10Mbps	n/a	0.972	6.742	86.0	0	30	20.9
UDP @ 20 Mbps	n/a	0.972	6.451	93.0	0	63	20.9

Latency = 1 ms

PC-A Upload – Shaped at 1Mbps:

	TCP Throughput (Mbps)	UDP Throughput (Mbps)	Jitter (ms)	Packets Lost (%)	Packets Arrived Out of Order	CPU Usage (%)	Memory Utilised (%)
TCP Stream	0.962	n/a	n/a	n/a	n/a	28	20.9
UDP @ 512kbps	n/a	0.512	0.000	0.0	0	3	20.9
UDP @ 1Mbps	n/a	0.972	12.556	2.1	0	5	20.9
UDP @ 2Mbps	n/a	0.972	5.531	51.0	0	11	20.9
UDP @ 10Mbps	n/a	0.972	9.392	90.0	0	57	20.9
UDP @ 20 Mbps	n/a	0.972	5.535	95.0	0	31	20.9

Latency = 1 ms

PC-B Download – Shaped at 10Mbps:

	TCP Throughput (Mbps)	UDP Throughput (Mbps)	Jitter (ms)	Packets Lost (%)	Packets Arrived Out of Order	CPU Usage (%)	Memory Utilised (%)
TCP Stream	9.62	n/a	n/a	n/a	n/a	28	20.9
UDP @ 512kbps	n/a	0.505	0.000	0.019	0	2	20.9
UDP @ 1Mbps	n/a	0.982	0.000	0.000	0	3	20.9
UDP @ 2Mbps	n/a	1.610	7.007	0.000	0	7	20.9
UDP @ 10Mbps	n/a	6.240	0.000	0.110	0	30	20.9
UDP @ 20 Mbps	n/a	9.660	4.289	12.000	0	63	20.9

Latency = 1 ms

PC B Upload – Shaped at 10Mbps:

	TCP Throughput (Mbps)	UDP Throughput (Mbps)	Jitter (ms)	Packets Lost (%)	Packets Arrived Out of Order	CPU Usage (%)	Memory Utilised (%)
TCP Stream	9.62	n/a	n/a	n/a	n/a	28	20.9
UDP @ 512kbps	n/a	0.512	0.000	0.0	0	3	20.9
UDP @ 1Mbps	n/a	1.000	0.000	0.0	0	5	20.9
UDP @ 2Mbps	n/a	2.000	0.162	0.0	0	11	20.9
UDP @ 10Mbps	n/a	9.720	3.269	2.7	4	57	20.9
UDP @ 20 Mbps	n/a	9.710	9.266	51.0	0	31	20.9

Latency = 1 ms

Appendix VIII – Raw Data From The Linux Platform.

Results For The Topology One Baseline.

PC-A Download:

	TCP Throughput (Mbps)	UDP Throughput (Mbps)	Jitter (ms)	Packets Lost (%)	Packets Arrived Out of Order	CPU Usage (%)	Memory Utilised (%)
TCP Stream	93.1	n/a	n/a	n/a	n/a	4.61	10.70
UDP @ 512kbps	n/a	0.512	0.000	0	0	0.18	11.45
UDP @ 1Mbps	n/a	1.000	0.000	0	0	0.16	11.54
UDP @ 2Mbps	n/a	2.000	0.000	0	0	0.14	11.50
UDP @ 10Mbps	n/a	10.000	0.006	0	0	0.15	11.50
UDP @ 20 Mbps	n/a	20.000	0.000	0	0	0.13	11.50

Latency = 0 ms

PC-A Upload:

	TCP Throughput (Mbps)	UDP Throughput (Mbps)	Jitter (ms)	Packets Lost (%)	Packets Arrived Out of Order	CPU Usage (%)	Memory Utilised (%)
TCP Stream	94.5	n/a	n/a	n/a	n/a	0.14	11.50
UDP @ 512kbps	n/a	0.512	0.00	0	0	0.11	11.53
UDP @ 1Mbps	n/a	1.000	0.000	0.00	0	0.13	11.56
UDP @ 2Mbps	n/a	2.000	0.000	0.00	0	0.13	11.56
UDP @ 10Mbps	n/a	10.000	0.001	0.00	0	0.14	11.56
UDP @ 20 Mbps	n/a	20.000	0.000	0.23	0	0.14	11.56

Latency = 0 ms

Results For Scenario A.

PC-A Download – Shaped at 1Mbps:

	TCP Throughput (Mbps)	UDP Throughput (Mbps)	Jitter (ms)	Packets Lost (%)	Packets Arrived Out of Order	CPU Usage (%)	Memory Utilised (%)
TCP Stream	0.958	n/a	n/a	n/a	n/a	0.11	11.62
UDP @ 512kbps	n/a	0.512	0.000	0.0	0	0.12	11.63
UDP @ 1Mbps	n/a	0.972	7.542	1.5	768	0.17	11.63
UDP @ 2Mbps	n/a	0.972	6.047	51.0	660	0.16	11.64
UDP @ 10Mbps	n/a	0.972	6.397	90.0	524	0.15	11.64
UDP @ 20 Mbps	n/a	0.972	5.072	95.0	512	0.11	11.64

Latency = 0 ms

PC-A Upload – Shaped at 1Mbps:

	TCP Throughput (Mbps)	UDP Throughput (Mbps)	Jitter (ms)	Packets Lost (%)	Packets Arrived Out of Order	CPU Usage (%)	Memory Utilised (%)
TCP Stream	0.958	n/a	n/a	n/a	n/a	0.13	11.64
UDP @ 512kbps	n/a	0.512	0.000	0.0	0	0.15	11.64
UDP @ 1Mbps	n/a	0.971	5.324	1.6	819	0.16	11.64
UDP @ 2Mbps	n/a	0.972	32.029	51.0	656	0.14	11.64
UDP @ 10Mbps	n/a	0.972	4.824	90.0	522	0.17	11.64
UDP @ 20 Mbps	n/a	0.972	7.534	95.0	514	0.12	11.64

Latency = 0 ms

Results For Scenario B.

PC-A Download – Shaped at 10Mbps:

	TCP Throughput (Mbps)	UDP Throughput (Mbps)	Jitter (ms)	Packets Lost (%)	Packets Arrived Out of Order	CPU Usage (%)	Memory Utilised (%)
TCP Stream	9.58	n/a	n/a	n/a	n/a	0.12	11.64
UDP @ 512kbps	n/a	0.512	0.000	0.0	0	0.14	11.64
UDP @ 1Mbps	n/a	1.000	0.000	0.0	0	0.13	11.64
UDP @ 2Mbps	n/a	2.000	0.000	0.0	0	0.14	11.64
UDP @ 10Mbps	n/a	9.710	3.029	2.8	780	0.12	11.64
UDP @ 20 Mbps	n/a	9.730	2.943	51.0	617	0.10	11.64

Latency = 0 ms

PC-A Upload Shaped at 10Mbps:

	TCP Throughput (Mbps)	UDP Throughput (Mbps)	Jitter (ms)	Packets Lost (%)	Packets Arrived Out of Order	CPU Usage (%)	Memory Utilised (%)
TCP Stream	9.58	n/a	n/a	n/a	n/a	0.10	11.64
UDP @ 512kbps	n/a	0.512	0.000	0.000	0	0.10	11.64
UDP @ 1Mbps	n/a	1.000	0.000	0.000	0	0.15	11.64
UDP @ 2Mbps	n/a	2.000	0.000	0.044	0	0.15	11.64
UDP @ 10Mbps	n/a	9.730	2.458	2.600	902	0.10	11.64
UDP @ 20 Mbps	n/a	9.730	3.331	51.000	628	0.10	11.64

Latency = 0 ms

Results For Scenario C.

PC-A Download – Shaped at 10Mbps:

	TCP Throughput (Mbps)	UDP Throughput (Mbps)	Jitter (ms)	Packets Lost (%)	Packets Arrived Out of Order	CPU Usage (%)	Memory Utilised (%)
TCP Stream	9.58	n/a	n/a	n/a	n/a	0.12	11.64
UDP @ 512kbps	n/a	0.512	0	0	0	0.14	11.64
UDP @ 1Mbps	n/a	1	0	0	0	0.17	11.64
UDP @ 2Mbps	n/a	2	0	0	0	0.14	11.64
UDP @ 10Mbps	n/a	9.71	2.871	2.7	891	0.11	11.64
UDP @ 20 Mbps	n/a	9.73	3.037	51	617	0.1	11.64

Latency = 0 ms

PC-A Upload – Shaped at 1Mbps:

	TCP Throughput (Mbps)	UDP Throughput (Mbps)	Jitter (ms)	Packets Lost (%)	Packets Arrived Out of Order	CPU Usage (%)	Memory Utilised (%)
TCP Stream	0.958	n/a	n/a	n/a	n/a	0.14	11.64
UDP @ 512kbps	n/a	0.512	0.000	0.0	0	0.15	11.64
UDP @ 1Mbps	n/a	0.972	407.569	1.5	800	0.11	11.64
UDP @ 2Mbps	n/a	0.972	8.041	51.0	660	0.13	11.64
UDP @ 10Mbps	n/a	0.973	6.151	90.0	523	0.20	11.64
UDP @ 20 Mbps	n/a	0.972	7.716	95.0	514	0.17	11.64

Latency = 0 ms

Results For The Topology Two Baseline.

PC-A Download:

	TCP Throughput (Mbps)	UDP Throughput (Mbps)	Jitter (ms)	Packets Lost (%)	Packets Arrived Out of Order	CPU Usage (%)	Memory Utilised (%)
TCP Stream	54.6	n/a	n/a	n/a	n/a	0.13	17.81
UDP @ 512kbps	n/a	0.402	0.110	0	0	0.12	17.81
UDP @ 1Mbps	n/a	0.980	0.000	0	0	0.12	17.81
UDP @ 2Mbps	n/a	1.780	0.000	0	0	0.13	17.81
UDP @ 10Mbps	n/a	5.570	1.773	0	0	0.13	18.11
UDP @ 20 Mbps	n/a	14.9	0.000	0	0	0.14	18.11

Latency = 0 ms

PC-A Upload:

	TCP Throughput (Mbps)	UDP Throughput (Mbps)	Jitter (ms)	Packets Lost (%)	Packets Arrived Out of Order	CPU Usage (%)	Memory Utilised (%)
TCP Stream	50.3	n/a	n/a	n/a	n/a	0.13	18.12
UDP @ 512kbps	n/a	0.512	0	0.000	0	0.11	18.12
UDP @ 1Mbps	n/a	1.000	0	0.000	0	0.16	18.12
UDP @ 2Mbps	n/a	2.000	0	0.000	0	0.13	18.12
UDP @ 10Mbps	n/a	10.000	0	0.018	0	0.15	18.12
UDP @ 20 Mbps	n/a	20.000	0	0.000	0	0.11	18.12

Latency = 0 ms

PC-B Download:

	TCP Throughput (Mbps)	UDP Throughput (Mbps)	Jitter (ms)	Packets Lost (%)	Packets Arrived Out of Order	CPU Usage (%)	Memory Utilised (%)
TCP Stream	40.7	n/a	n/a	n/a	n/a	0.13	17.81
UDP @ 512kbps	n/a	0.384	0.000	0.000	0	0.12	17.81
UDP @ 1Mbps	n/a	0.930	4.842	0.000	0	0.12	17.81
UDP @ 2Mbps	n/a	1.660	1.302	0.000	0	0.13	17.81
UDP @ 10Mbps	n/a	8.620	0.000	0.006	0	0.13	18.11
UDP @ 20 Mbps	n/a	10.400	0.106	0.000	0	0.14	18.11

Latency = 0 ms

PC-B Upload:

	TCP Throughput (Mbps)	UDP Throughput (Mbps)	Jitter (ms)	Packets Lost (%)	Packets Arrived Out of Order	CPU Usage (%)	Memory Utilised (%)
TCP Stream	43.5	n/a	n/a	n/a	n/a	0.13	18.12
UDP @ 512kbps	n/a	0.512	0.000	0.000	0	0.11	18.12
UDP @ 1Mbps	n/a	1.000	0.000	0.000	0	0.16	18.12
UDP @ 2Mbps	n/a	2.000	0.108	0.000	0	0.13	18.12
UDP @ 10Mbps	n/a	9.900	0.000	0.021	0	0.15	18.12
UDP @ 20 Mbps	n/a	20.000	0.000	0.000	0	0.11	18.12

Latency = 0 ms

Results For Scenario D.

PC-A Download – Shaped at 10Mbps:

	TCP Throughput (Mbps)	UDP Throughput (Mbps)	Jitter (ms)	Packets Lost (%)	Packets Arrived Out of Order	CPU Usage (%)	Memory Utilised (%)
TCP Stream	9.56	n/a	n/a	n/a	n/a	0.12	17.5
UDP @ 512kbps	n/a	0.508	0.000	0.019	0	0.14	17.51
UDP @ 1Mbps	n/a	0.940	0.000	0.000	0	0.13	17.51
UDP @ 2Mbps	n/a	1.890	7.575	0.000	0	0.12	17.51
UDP @ 10Mbps	n/a	9.000	2.541	0.007	4	0.12	17.51
UDP @ 20 Mbps	n/a	9.730	0.036	40.000	669	0.13	17.55

Latency = 0 ms

PC-A Upload – Shaped at 10Mbps:

	TCP Throughput (Mbps)	UDP Throughput (Mbps)	Jitter (ms)	Packets Lost (%)	Packets Arrived Out of Order	CPU Usage (%)	Memory Utilised (%)
TCP Stream	9.58	n/a	n/a	n/a	n/a	0.11	11.82
UDP @ 512kbps	n/a	0.512	0.000	0.0	0	0.17	11.82
UDP @ 1Mbps	n/a	1.000	0.000	0.0	0	0.14	11.82
UDP @ 2Mbps	n/a	2.000	7.559	0.0	0	0.14	11.82
UDP @ 10Mbps	n/a	9.730	3.320	2.6	903	0.10	11.82
UDP @ 20 Mbps	n/a	9.730	3.123	51.0	633	0.13	11.85

Latency = 0 ms

PC-B Download – Shaped at 10Mbps:

	TCP Throughput (Mbps)	UDP Throughput (Mbps)	Jitter (ms)	Packets Lost (%)	Packets Arrived Out of Order	CPU Usage (%)	Memory Utilised (%)
TCP Stream	9.58	n/a	n/a	n/a	n/a	0.12	17.50
UDP @ 512kbps	n/a	0.364	9.294	0.16	0	0.14	17.51
UDP @ 1Mbps	n/a	0.992	2.532	0.00	0	0.13	17.51
UDP @ 2Mbps	n/a	1.520	4.562	0.00	0	0.12	17.51
UDP @ 10Mbps	n/a	5.220	1.027	0.00	0	0.12	17.51
UDP @ 20 Mbps	n/a	9.060	2.394	3.70	0	0.13	17.55

Latency = 0 ms

PC-B Upload – Shaped at 10Mbps:

	TCP Throughput (Mbps)	UDP Throughput (Mbps)	Jitter (ms)	Packets Lost (%)	Packets Arrived Out of Order	CPU Usage (%)	Memory Utilised (%)
TCP Stream	9.58	n/a	n/a	n/a	n/a	0.11	11.82
UDP @ 512kbps	n/a	0.512	0.000	0.0	0	0.17	11.82
UDP @ 1Mbps	n/a	1.000	0.000	0.0	0	0.14	11.82
UDP @ 2Mbps	n/a	2.000	0.000	0.0	0	0.14	11.82
UDP @ 10Mbps	n/a	9.730	2.522	2.6	892	0.10	11.82
UDP @ 20 Mbps	n/a	9.73	2.873	51.0	652	0.13	11.85

Latency = 0 ms

Results For Scenario E.

PC-A Download – Shaped at 1Mbps:

	TCP Throughput (Mbps)	UDP Throughput (Mbps)	Jitter (ms)	Packets Lost (%)	Packets Arrived Out of Order	CPU Usage (%)	Memory Utilised (%)
TCP Stream	0.958	n/a	n/a	n/a	n/a	0.11	18.40
UDP @512kbps	n/a	0.511	0.000	0.00	0	0.10	18.16
UDP @ 1Mbps	n/a	0.968	4.953	0.26	380	0.13	18.12
UDP @ 2Mbps	n/a	0.971	7.611	46.00	680	0.12	18.13
UDP @ 10Mbps	n/a	0.972	7.991	87.00	536	0.10	18.12
UDP @ 20 Mbps	n/a	0.972	7.915	93.00	516	0.10	18.14

Latency = 0 ms

PC-A Upload – Shaped at 1Mbps:

	TCP Throughput (Mbps)	UDP Throughput (Mbps)	Jitter (ms)	Packets Lost (%)	Packets Arrived Out of Order	CPU Usage (%)	Memory Utilised (%)
TCP Stream	0.957	n/a	n/a	n/a	n/a	0.10	17.83
UDP @ 512kbps	n/a	0.512	0.000	0.0	0	0.14	17.83
UDP @ 1Mbps	n/a	0.972	2.897	1.5	739	0.12	17.82
UDP @ 2Mbps	n/a	0.972	7.274	51.0	656	0.10	18.38
UDP @ 10Mbps	n/a	0.972	6.261	90.0	528	0.11	18.41
UDP @ 20 Mbps	n/a	0.972	7.435	95.0	513	0.11	18.41

Latency = 0 ms

PC-B Download – Shaped at 10Mbps:

	TCP Throughput (Mbps)	UDP Throughput (Mbps)	Jitter (ms)	Packets Lost (%)	Packets Arrived Out of Order	CPU Usage (%)	Memory Utilised (%)
TCP Stream	9.58	n/a	n/a	n/a	n/a	0.11	18.40
UDP @ 512kbps	n/a	0.448	15.069	0.000	0	0.10	18.16
UDP @ 1Mbps	n/a	0.870	6.643	0.000	0	0.13	18.12
UDP @ 2Mbps	n/a	1.580	6.213	0.000	0	0.12	18.13
UDP @ 10Mbps	n/a	6.110	1.230	0.078	9	0.10	18.12
UDP @ 20 Mbps	n/a	9.510	0.254	8.600	589	0.10	18.14

Latency = 0 ms

PC-B Upload – Shaped at 10Mbps:

	TCP Throughput (Mbps)	UDP Throughput (Mbps)	Jitter (ms)	Packets Lost (%)	Packets Arrived Out of Order	CPU Usage (%)	Memory Utilised (%)
TCP Stream	9.58	n/a	n/a	n/a	n/a	0.10	17.83
UDP @512kbps	n/a	0.512	0.000	0.0	0	0.14	17.83
UDP @ 1Mbps	n/a	1.000	0.000	0.0	0	0.12	17.82
UDP @ 2Mbps	n/a	2.000	1.058	0.0	0	0.10	18.38
UDP @ 10Mbps	n/a	9.730	2.944	2.6	822	0.11	18.41
UDP @ 20 Mbps	n/a	9.730	4.410	51.0	632	0.11	18.41

Latency = 0 ms

Appendix IX – Original Project Specification.

Sheffield Hallam University.
Faculty of Arts, Engineering and Sciences
BSc (Hons) Network Management (Top – Up year)

Project Definition:

Student Name: Gregory James McMullin
Date: November 2009
Supervisor: Iain Hughes.
Level Of Project: BSc (Hons) Network Management.
Title Of Project: Connection speed throttling techniques on a multi-user LAN.
Type Of Project: Investigation Based.

Elaboration:

With high-speed broadband Internet connections quickly becoming an essential service many student halls and private blocks of flats are being built pre wired with a switched Ethernet Local Area Network (LAN). An Internet Service Provider (ISP) can then connect a high bandwidth leased line to the LAN and provide each networked flat or room with a direct high-speed connection to the Internet.

This kind of network architecture has the potential to provide the customer with connection speeds far in excess of current ADSL and cable connections, however any connection speeds will depend entirely on the ISP's network backbone and the restrictions put in place by the ISP.

With no restrictions in place a customer could feasibly connect a device to their Ethernet point and connect to the Internet at 100Mbps. This would not be acceptable, as ISPs need to be able to limit the connection speed that customers can achieve; this combined with bandwidth contention makes providing the service profitable. Also ISPs may wish to sell different levels of service so it is in their interest to be able to specify what speed a specific customer should receive. For example they may wish to sell an entry-level package of 1Mbps a top range package of 10 Mbps. An effective method of setting the connection speed limit on an Ethernet LAN is a priority for the ISP.

As it stands there seems to be no industry standard way of achieving these goals. This project aims to explore, test and evaluate some of the techniques that could be used to deliver effective connection speed limiting. And therefore provide a set of recommendations for any ISP wishing to deliver broadband in this fashion.

Project Ethics:

All ethical issues relating to the project have been considered and appropriate courses of action have been decided upon. This includes but is not limited to the use of human subjects during data collection and/or testing.

Project Objectives and Deliverable:

1. Research data rate throttling and bandwidth capping methods.
2. Investigate current practices in the industry.

3. Define a scenario that will be investigated.
4. Identify and research possible techniques for scenario.
5. Identify techniques that are worth testing.
6. Define a testing methodology.
7. Test techniques.
8. Gather results from tests.
9. Investigate and evaluate results.
10. Produce report and recommendations.
11. Define other areas of investigation that could further the findings of the report.

The deliverable for this project will be a report and a set of recommendations on the techniques available for data rate throttling and bandwidth capping.

Task Plan:

Task

Milestone Date

Investigation Phase:

- | | |
|---------------------------------------------|----------------------|
| 1. Research throttling and capping methods. | 27 th Nov |
| 2. Investigate current industry methods. | 4 th Dec |
| 3. Investigate current published research. | 18 th Dec |
| 4. Identify methods to test | 31 st Dec |

Testing Phase:

- | | |
|--------------------------------------------|----------------------|
| 5. Test methods. | 31 st Jan |
| 6. Document testing. | 10 th Feb |
| 7. Document and collate test results. | 17 th Feb |
| 8. Compare results from different methods. | 28 th Feb |

Evaluation Phase:

- | | |
|--------------------------------------------------------------------|----------------------|
| 9. Produce report on testing Phase. | 7 th Mar |
| 10. Make final recommendations. | 14 th Mar |
| 11. Produce a critical reflection on the project as a whole. | 21 st Mar |
| 12. Single out areas of investigation that could be expanded upon. | 28 th Mar |

Concluding Phase:

- | | |
|---------------------------------------------------|------------------------|
| 13. Conclude report. | 10 th April |
| 14. Write up, print and bind project for hand in. | 20 th April |
| 15. Hand in project. | 23 rd April |

Appendix X – The Contents Of The Supplied CD.

The Supplied CD contains the following files:

- Experimental Results.
 - Raw Data.zip – Contains the raw data collected from the experimentation.
- Iperf
 - Iperf.exe – the iperf program
- Referenced Reports
 - Arbor e30 Data Sheet.pdf
 - Cisco IOS Quality of Service Solutions Command Reference Release 12.2.pdf
 - Comparing Traffic Policing and Traffic Shaping for Bandwidth Limiting.pdf
 - Designing And Implementing Linux Firewalls And QoS (2006).pdf
 - Mikrotik_what_is_routeros.pdf
 - PacketShaper_Datasheet.0.pdf
- This Report
 - Dissertation.docx – an electronic copy of this report.

A final note from the author:

Should you have any questions or want to contact me about this report I can be reached at the following email address:

greg@mcmull.in