



PROJECT REPORT – MINT 709

ON

TRAFFIC SHAPING

Submitted By:
Gurvir Singh Gill
gurvir@ualberta.ca



**PROJECT REPORT SUBMITTED IN PARTIAL
FULFILLMENT OF THE REQUIREMENTS FOR
THE MASTER'S DEGREE**

**In
MINT
MASTERS IN INTERNETWORKING**

**UNIVERSITY OF ALBERTA
AUGUST 2007**

© Gurvir Singh Gill, 2007



Submitted To: Dr. Mike MacGregor

Submitted By: Er. Gurvir Singh Gill

Dated: August 7, 2007

Abstract

Traffic Shaping is a mechanism that alters the traffic characteristics of a stream of cells on a connection to achieve better network efficiency, while meeting the QoS objectives, or to ensure conformance at a subsequent interface. Traffic shaping provides a mechanism to control the amount and volume of traffic being sent into a network (bandwidth throttling), and the rate at which the traffic is being sent (rate limiting).

Every service provider in his contract has guaranteed certain Quality of Service to there customers .To maintain that QoS , they have to implement certain policies and differentiate between data. They differentiate between data
Simple traffic shaping schemes shape all traffic uniformly by rate. More sophisticated shapers first classify traffic.

Traffic shaping is used to control access to available bandwidth by preventing packet loss. Congestion can be avoided, by traffic shaping, that can occur when the sent traffic exceeds the access speed of its remote.

There are different ways with which bandwidth can be managed through traffic shaping .Traffic shaping can be achieved by applying rules on the applications. A particular amount of bandwidth can be allocated to certain applications , after that particular bandwidth is used then queue starts building up. Traffic Shaping can also be implemented on per-user by making rules for users. In that case bandwidth is allocated to user not to the particular application. In addition to setting traffic shaping on per-application and per-user, traffic shaping can also be applied to delay sensitive data to make it important. That can be done by applying priority to some particular data. So when there is congestion, priority is given to that data .

The main motive in this project is to make the network really close to the real scenario. There are two ISP's in this project, one is primary and other is backup ISP. There are one link for each branch office to primary ISP for primary voice + data and for video + data. The other link is for other branch to backup ISP for backup voice + data and for video + data.

In this project, GTS ,CBTS ,FRTS are applied to the network one by one and there effect has been noticed by sending just the voice and voice plus video. 2600, 2800 series routers and 3500, 3600 series switches are being used in the project. Windows Server 2003 is used as VLC video server and Dell Windows XP Professional notebooks are being used as VLC client. Cisco IP Phones 7960 are being used for sending voice traffic in the network. After applying GTS in the network, status of traffic shaping in Non-Active , but

when video is send over that link , then status of traffic shaping comes to Active. But when video was stopped the status of traffic shaping again comes to Non-Active.

In CBTS, CBWFQ was implemented as queuing mechanism. Two classes with different parameters were configured, to see the desired results. According to CBWFQ if one class goes out of it available bandwidth then it can use the remaining bandwidth of the other. To see that result practically, this part had been divided into two sub parts. In the first the class with fewer bandwidth goes out of it's available bandwidth so it used the remaining bandwidth of the other class.

In FRTS, one class was implemented on voice traffic and default class is for video and other data. Due to not enough bandwidth the video traffic starts dropping at a high rate, but voice traffic is going smoothly over the same interface but in different class. Video/data didn't use the remaining bandwidth of the class used for voice because traffic shaping is being implemented in the network.

Protocols selection also plays very important role in the network. In this project Border Gateway Protocol (BGP) and Open Short Path First (OSPF) are being used. BGP is the protocol that is being widely used by ISP's

Contents

Abstract	4
Contents	6
Acknowledgement	12

1 Introduction of Traffic Shaping

1.1 Background	13
1.2 Traffic Shaping	13
1.2.1 Why We Use Traffic Shaping?	14
1.2.2 Managing Bandwidth with TS.	14
1.2.3 Traffic Classification.	14
1.3 Major ISP's Using Traffic Shaping.	15
1.4 QoS Mechanism.	16
1.5 Difference between Traffic Shaping and Policing.	16
1.6 Token Bucket.	18
1.7 Differences Between Traffic Shaping Mechanisms.	22
1.8 Summary	22

2 Network Components and Protocols

2.1 Border Gateway Protocol.	23
2.1.1 Path Attributes.	23
2.1.2 BGP Path Selection.	24

2.1.3	Enabling BGP Routing.	24
2.1.4	Configuring BGP Neighbors.	25
2.1.5	Sending Default-Route.	26
2.1.6	Setting Local Preferences.	27
2.1.7	Specifying Route Maps.	28
2.2	Open Short Path First.	29
2.2.1	SPF Algorithm.	30
2.3	Real Time Protocol.	31
2.4	Network Time Protocol.	31
2.5	Skinny Client Protocol	32
2.6	Equipment Used	32
2.7	Monitoring Tools and Player Used.	33
2.8	Topology.	34
2.8.1	Logical Topology.	35
2.8.2	Physical Topology.	36
2.9	IP Addressing	36
2.10	Explanation of Topology Used	38
2.11	Summary	42

3 CISCO Call Manager Express

3.1	Setting Up DHCP Service.	43
3.2	Setting IP phones with CME.	44
3.3	Setting Up Initial Extensions.	45
3.3.1	Manual Setup Using the Router CLI.	46

3.4	Cisco IP Phones.47
3.4.1	Types of Cisco IP Phones.	47
3.5	IP Phone Startup Process.	48
3.6	Cisco IP Phone Codec Support.	49
3.7	Summary	50
4	Configurations	
4.1	Telus2's Running Configuration.51
4.2	Telus1's Running Configuration.	53
4.3	Shaw's Running Configuration	55
4.4	MINT-EDM1's Running Configuration.56
4.5	MINT-EDM2's Running Configuration.60
4.6	MINT-CAL's Running Configuration.	62
4.7	FR-SW's Running Configuration.66
4.8	Summary	68
5	Generic Traffic Shaping	
5.1	Generic Traffic Shaping.	69
5.2	Features of GTS.	70
5.3	Weighted Fair Queuing71
5.3.1	How WFQ Work?	71
5.3.2	Restrictions.	72
5.4	Configuring GTS.	73
5.5	Configure GTS for an Access List.74

5.6	Network Topology for GTS.	75
5.7	Shaw’s Configuration with GTS – Case1.	75
5.8	Statistics – Case1	76
5.8.1	Statistics 1.	76
5.8.2	Statistics 2 - Before Video was played.	76
5.8.3	Statistics 3 - After Video was played.	76
5.8.4	Statistics 4 - After Video was Stopped.	77
5.9	Summary of Statistics 1.	77
5.10	Statistics1 from Ethereal.	78
5.11	Telus1’s Configuration with GTS – Case2.	79
5.12	Statistics – Case 2.	79
5.12.1	Statistics 1.	77
5.12.2	Statistics 2 – Video was played and then stopped.	80
5.13	Summary of Statistics 2.	81
5.14	Statistics2 from Ethereal.	81
5.15	Conclusion.	82
6	Class Based Traffic Shaping	
6.1	Class Based Traffic Shaping.	84
6.2	Class-Based Weighted Fair Queuing.	84
6.2.1	Why CBWFQ?	84
6.2.2	How CBWFQ Works?	85
6.2.3	Advantages of CBTS.	85

6.3	CBTS with CBWFQ.	86
6.4	Configuring CBWFQ.	86
6.5	Configuring CBTS with CBWFQ.	85
6.5.1	Creating a Traffic Class.	87
6.5.2	Creating a Traffic Policy.	87
6.5.3	Attaching Policy to an Interface.	88
6.6	Network Topology for CBTS.	89
6.7	Telus1's Configuration with CBTS – Case 1.	89
6.8	Statistics1.	91
6.9	Summary of Statistics 1.	92
6.10	Telus1's Configuration with CBTS – Case 2.	92
6.11	Statistics2.	93
6.12	Statistics2's Summary.	94
6.13	Statistics from Ethereal.	95
6.14	Conclusion.	95
7	Frame Relay Traffic Shaping	
7.1	Frame Relay Traffic Shaping.	97
7.2	How FRTS Works?	99
7.3	Difference between VoFR and VoIP.	99
7.4	Reliability and QoS Guarantees.	100
7.5	Differences Between FRTS and GTS.	101
7.6	Priority Queuing.	101
7.6.1	How PQ Works?	102

7.6.2	Queue Sizes.	102
7.6.3	Configuring PQ.	103
7.7	BECN.	104
7.8	Configuring FRTS.	105
7.9	Network Topology for FRTS.	106
7.10	Telus1's Running Configuration with FRTS.	107
7.11	MINT-EDM1's Running Configuration with FRTS.	108
7.12	MINT-EDM2's Running Configuration with FRTS.	108
7.13	Statistics.	109
17.13.1	Statistics 1.	109
17.13.2	Statistics 2.	110
7.14	Statistics Summary.	112
7.15	Statistics from Ethereal.	113
7.16	Conclusion.	116
8	Theory v/s Experiment	
8.1	GTS with WFQ.	119
8.2	CBTS with CBWFQ.	119
8.3	FRTS with PQ.	120
8.4	Summary.	120
9	Conclusion and Future Directions.	121
10	Bibliography.	122

Acknowledgement

With the grace of God I have completed my project report. For this I am indebted to God Almighty for providing me with the power to believe in myself and hit the bull's eye.

I am grateful to Dr. Mike MacGregor without whose motivation and guidance this dissertation project could not have been completed. Dr. Mike MacGregor who is not only the Director of MINT but also guided me at each and every step. He provided all the equipment that was needed to make this scenario as real time scenario. He encouraged and motivated me throughout my academic program. I really want to thank Dr. Mike MacGregor for his continuous support in every aspect.

I would also like to thank my mentor Mr. Cesar Barrero and all other faculty members of MINT Program with whose support I am able to implement the concepts that I learnt during the courses of MINT and able to make this project a real time scenario project.

I owe my parents for giving me life in the first place, educating me with aspects from life and studies, unconditional support and encouragement to pursue my masters even when the interests went beyond boundaries of geography.

Last but not the least; I would like to thank Er. Maninder Pal Singh, my co partner in this project, for his continuous support throughout the project. He never accepted less than our best efforts.

I thank them all.

Chapter 1

Introduction of Traffic Shaping

1.1 Background

In the past few decades, advances in Internet traffic and technological improvements in network and communication infrastructure have revolutionized the internet and communication environment around us .The eighties and nineties of the past century have seen the rapid growth of world wide web that has a significant impact on our day-to-day lives.

Every service provider wants to provide better services to there client .So they have to implement some rules/policies so that they can differentiate between different traffic and applications. In some applications ,delay is non acceptable, they are time sensitive. For example VoIP , online gaming. On the other hand there are some applications that are not so sensitive to delay. For example non ISP's VoIP examples for that are Yahoo Messenger and Skype. Therefore, shaping increases buffer utilization on a router, but causes non-deterministic packet delays. Shaping can also interact with a Frame Relay network, adapting to indications of Layer-2 congestion in the WAN.

1.2 Traffic Shaping

Traffic shaping is an attempt to control computer network traffic in order to optimize or guarantee performance, low latency, and/or bandwidth by delaying packets. Traffic shaping deals with concepts of classification, queue disciplines ,enforcing policies, congestion management, quality of service (QoS), and fairness. Traffic shaping allows us to control the traffic going out an interface in order to match its flow to the speed of the remote, target interface and to ensure that the traffic conforms to policies contracted for it. Thus, traffic adhering to a particular profile can be shaped to meet downstream requirements, thereby eliminating bottlenecks in topologies with data-rate mismatches.

Traffic shaping provides a mechanism to control the volume of traffic being sent into a network, and the rate at which the traffic is being sent .This control can be accomplished in many ways and for many reasons but traffic shaping always simply consists in delaying packets. Its use is especially important in Frame Relay networks because the

switch cannot determine which packets take precedence, and therefore which packets should be dropped when congestion occurs.

1.2.1 Why We Use Traffic Shaping?

The primary reasons why we would use traffic shaping are:

- à to control access to available bandwidth,
- à to ensure that traffic conforms to the policies established for it, and
- à to regulate the flow of traffic
- à to avoid congestion that can occur when the sent traffic exceeds the access speed of its remote.
- à to prevent packet loss
- à to control the traffic going out an interface, matching the traffic flow to the speed of the interface.
- à to ensure that traffic conforms to the policies contracted for it.
- à to ensure that a packet adheres to a stipulated contract and determines the appropriate quality of service to apply to the packet.
- à to avoid bottlenecks and data-rate mismatches. For instance, central-to-remote site data speed mismatches.

1.2.2 Ways of Managing Bandwidth with Traffic Shaping

1. *Per-application rules.* With Traffic Shaping we can identify and categorize specific types of network traffic, constraining each particular category of traffic to use no more than a specified amount of bandwidth. For example we can give particular amount of bandwidth to data packets and when that amount is reached then the queue starts building up.
2. *Per-user rules.* With Traffic Shaping we can set per-user traffic limits to ensure that network traffic is shared fairly among all users. For example if there are 3 users X, Y and Z and we want that X gets 45% of the available bandwidth and 40% to Y and rest 15% to Z, then we can make certain rules per user and implement them.
3. *Priority management.* In addition to setting traffic limits on a per-application or per-user basis, traffic shaping can also be used to define the relative importance, or priority, of different types of traffic. For example if we want that voice is given more priority over all other traffic then we can implement that also. With that every time the voice will be given priority over other traffic.

1.2.3 Traffic Classification

Simple traffic shaping schemes shape all traffic uniformly by rate. More sophisticated shapers first classify traffic. Traffic classification categorises traffic (for example, based on port number or protocol number); each resulting traffic class can be treated differently

to differentiate service. For example, each traffic class could be subject to a different rate limit, shaped separately and/or prioritised relative to other traffic classes. This differentiation can be used by a network operator to treat different types of application traffic differently.

1.2.3 a - Types of Traffic

Every service provider in his contract has guaranteed certain Quality of Service to there customers (which includes companies, universities, domestic users). To maintain that QoS , they have to implement certain policies and differentiate between data. As some data are crucial and can't handle delay , loss (e.g voice , online gaming , video conferencing) , on the other hand there are some data , application that can afford certain delay(e.g peer-to-peer , non local VoIP).So with respect to that traffic has been classified into 3 types :

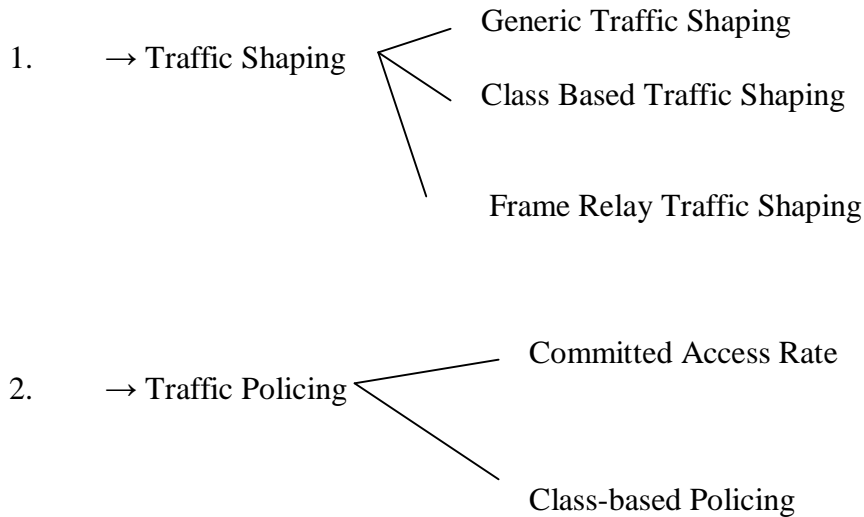
- à Sensitive – Sensitive traffic is traffic whose Quality of Service ISPs care about. This usually includes VoIP, online gaming, video streaming, and web surfing, but basically any application or protocol could fall under this umbrella. haping schemes are generally tailored in such a way that the Quality of Service of these selected uses is guaranteed, or at least prioritized over other classes of traffic.
- à Best-Effort – This is traffic that is either not sensitive to Quality of Service metrics (jitter, packet loss, latency) or traffic that is, and the ISP is not concerned about its Quality of Service. A typical example of the former would be peer-to-peer traffic . .
- à Undesired – This category is generally limited to the delivery of spam and traffic created by worms, botnets, and other malicious attacks.

1.3 Major Internet Service Providers Using Traffic Shaping

ISP	Country
Rogers Cable	Canada
Shaw Cable	Canada
Telus	Canada
Comcast Cable	USA
Cogeco Cable	USA
Road Runner	USA
Clearwire	USA
PlusNet	UK
BT Openworld	UK
Pipex	UK
Virgin Media	UK and USA

1.4 QoS Mechanism

There are two QoS mechanisms that are used to limit the available bandwidth to traffic classes.



1.5 Difference between Traffic Shaping and Policing

Traffic Shaping	Traffic Policing
1 .Shaping does not drops packets.	1. Policing does drop packets
2 . Shaping is less efficient in terms of memory utilization	2. Policing is more efficient in terms of memory utilization
3. Shaping requires an additional queuing system , thus increasing buffer usage.	3. Policing does not increase buffer usage
4. Shaping adds variable delay to traffic, possibly causing jitter	4. Policing drops packets , causing retransmissions .
5. Shaping can be applied only on outbound interface.	5. Policing can be applied on outbound and inbound interface.

6. Shaping controls bursts by smoothing the output rate.	6. Policing propagates bursts. Smoothing is not a part of policing.
--	---

Table 1.1

The *Figure 1.1* illustrates the key difference. Traffic policing propagates bursts. When the traffic rate reaches the configured maximum rate, excess traffic is dropped (or remarked). The result is an output rate that appears as a saw-tooth with crests and troughs. In contrast to policing, traffic shaping retains excess packets in a queue and then schedules the excess for later transmission over increments of time. The result of traffic shaping is a smoothed packet output rate.

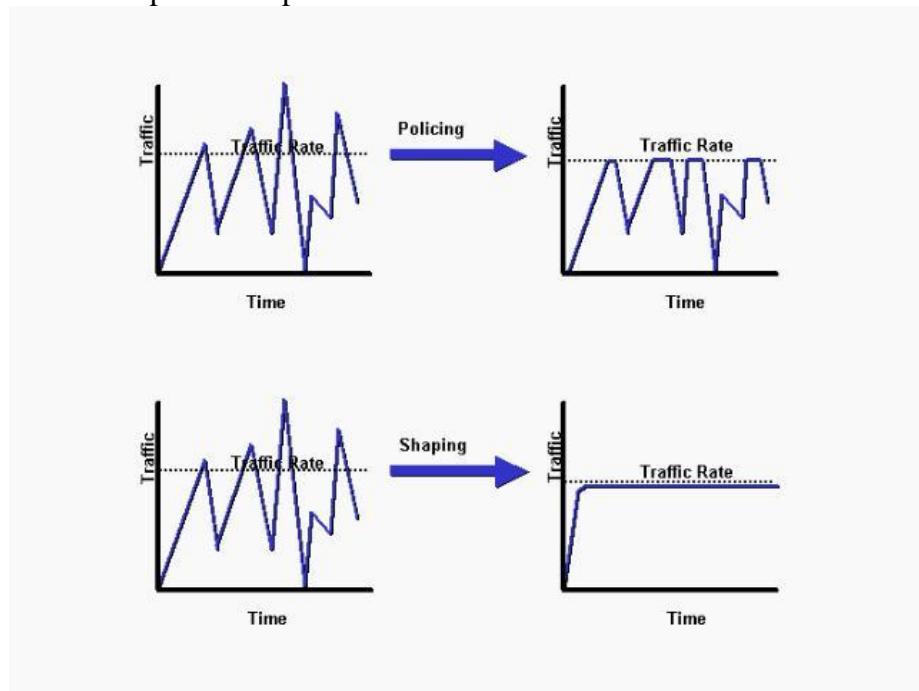


Figure 1.1

→ *Traffic Shaping* - Shaping meters the traffic rate and delays excessive traffic so that it stays within the desired rate limit. With shaping, traffic bursts are smoothed out producing a steadier flow of data. Reducing traffic bursts helps reduce congestion in the core of the network. Traffic shaping smoothes traffic by storing traffic above the configured rate in a queue.

→ *Traffic Policing* - Policing drops excess traffic in order to control traffic flow within specified limits. Policing does not introduce any delay to traffic that conforms to traffic policies. It can however, cause more TCP retransmissions, because traffic in excess of specified limits is dropped. Policing is more efficient in terms of memory utilization as no additional buffering of packets is needed.

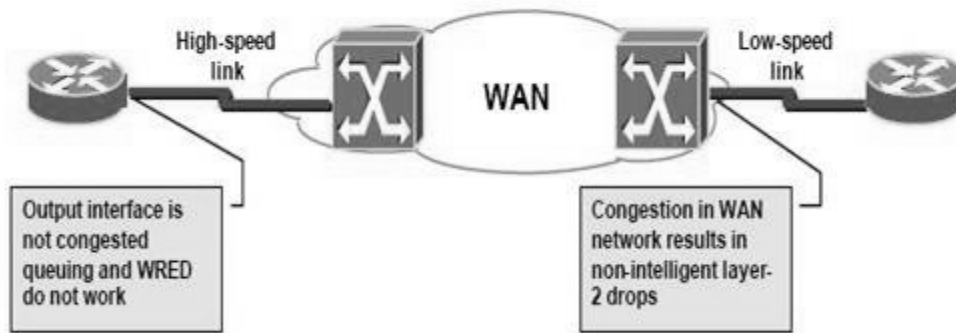


Figure 1.2

Both shaping and policing mechanisms are used in a network to control the rate at which traffic is admitted into the network. Both mechanisms use classification, so they can differentiate traffic. They also use metering to measure the rate of traffic and compare it to the configured shaping or policing policy.

1.6 Token Bucket

Token Bucket is the mechanism that is used to constrain traffic to a particular bit rate. The token bucket is a mathematical model used in a device that regulates the data flow. The token bucket model is used whenever a new packet is processed and the return value is either “conform” or “exceed” as shown in *figure 1.3* in the following page.

The mode has two basic components:

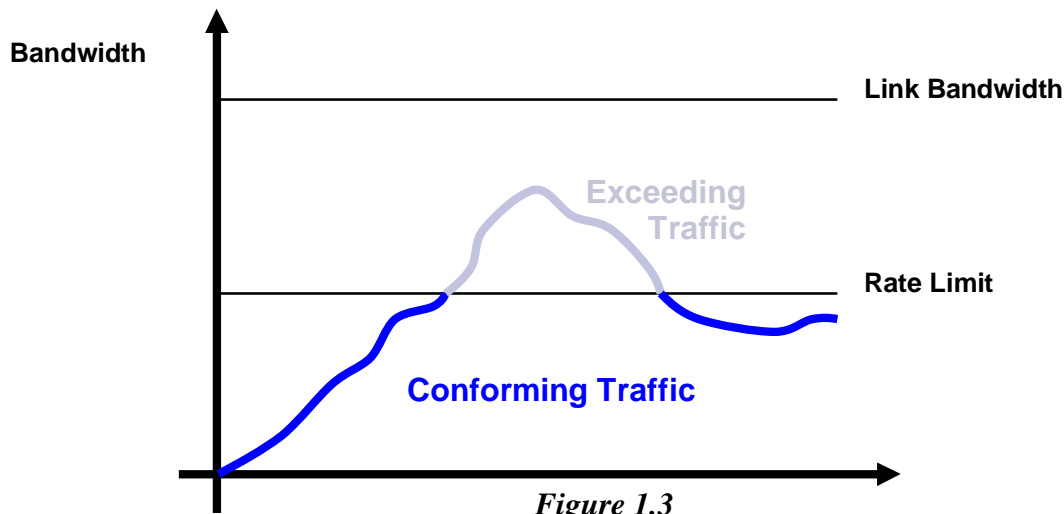
- *Tokens*: where each token represents the permission to send a fixed number of bits into the network
- *Bucket*: which has the capacity to hold a specified amount of tokens

Tokens are put into the bucket at a certain rate by the operating system. Each incoming packet, if forwarded, takes tokens from the bucket, representing the packet’s size. If the bucket fills to capacity, newly arriving tokens are discarded.

If there are not enough tokens in the bucket to send the packet, the regulator may:

- Wait for enough tokens to accumulate in the bucket (traffic shaping)
- Discard the packet (policing)

$$\text{Committed Information Rate (CIR)} = \text{burst size (Bc)} / \text{time interval}$$



Here are some terms explained that are used in traffic shaping:

- Mean rate (CIR) —Also called the committed information rate (CIR), it specifies how much data can be sent or forwarded per unit time on average.
- Burst size (Bc) —Also called the Committed Burst (Bc) size, it specifies in bits (or bytes) per burst how much traffic can be sent within a given unit of time to not create scheduling concerns.
- Excess Burst Size (Be) – It specifies length of burst .The Excess Burst size corresponds to the number of non-committed bits (those outside the committed information rate (CIR), that are still accepted by the Frame Relay switch but marked as discard eligible. The Be size allows more than the burst size to be sent during a time interval in certain situations. The switch will allow the packets belonging to the Excess Burst to go through but it will mark them by setting the discard eligible (DE) bit. When the Be size equals 0, the interface sends no more than the burst size every interval, achieving an average rate no higher than the mean rate. However, when the Be size is greater than 0, the interface can send as many as Bc+Be bits in a burst, if in a previous time period the maximum amount was not sent. Whenever less than the burst size is sent during an interval, the remaining number of bits, up to the Excess Burst size, can be used to send more than the burst size in a later interval.
- Time interval—Also called the measurement interval, it specifies the time quantum in seconds per burst.
- Discard Eligible Bit (DE) -We can specify which Frame Relay packets have low priority or low time sensitivity and will be the first to be dropped when a Frame Relay switch is congested. The mechanism that allows a Frame Relay switch to identify such packets is the DE bit.

Both GTS and FRTS use a construct called a token bucket to rate-limit traffic. A token bucket differs from a leaky bucket in that a token bucket should be thought of as being filled with tokens, not packets. We can think of tokens as permissions for a specific number of bits to be transmitted to the network. The token bucket is also commonly referred to as a *credit manager* that gives credits to traffic to be used for transmission.

Before a packet is sent out the interface, a certain number of tokens need to be removed from the bucket. Tokens fill the token bucket at a constant rate, and the bucket is a certain size

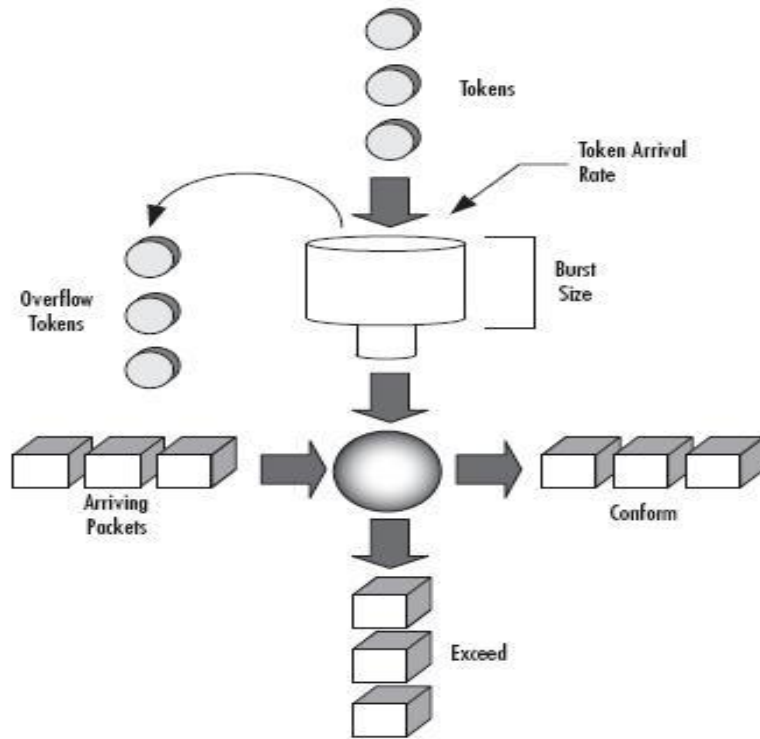


Figure 1.4

. After the bucket is full, newly arriving tokens are discarded. If the bucket is empty, an incoming packet has to wait for enough tokens to fill the bucket before it can be transferred. Thus, with the token bucket analogy, the burst size is roughly proportional to the size of the bucket. A depiction of a token bucket is shown in *Figure 1.4*.

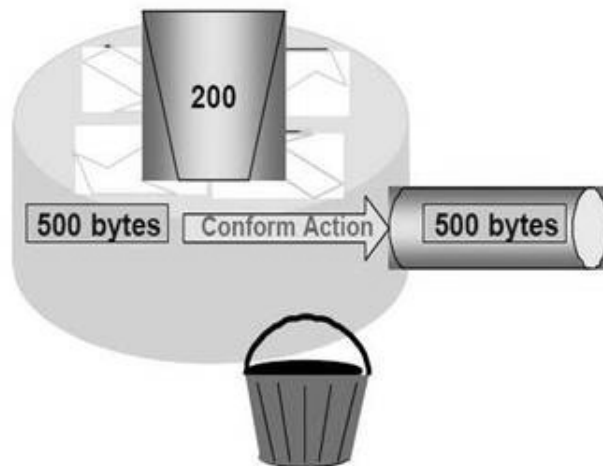


Figure 1.5

The *figure 1.5* shows a token bucket, with the current capacity of 700 bytes. When a 500-byte packet arrives at the interface, its size is compared to the bucket capacity (in bytes). The packet conforms to the rate limit (500 bytes < 700 bytes), and the packet is forwarded. 500 tokens are taken out of the token bucket leaving 200 tokens for the next packet.

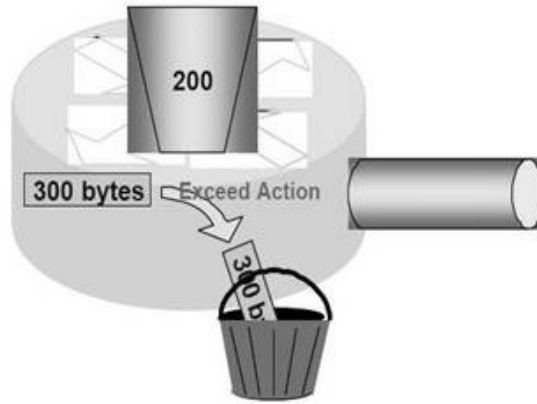
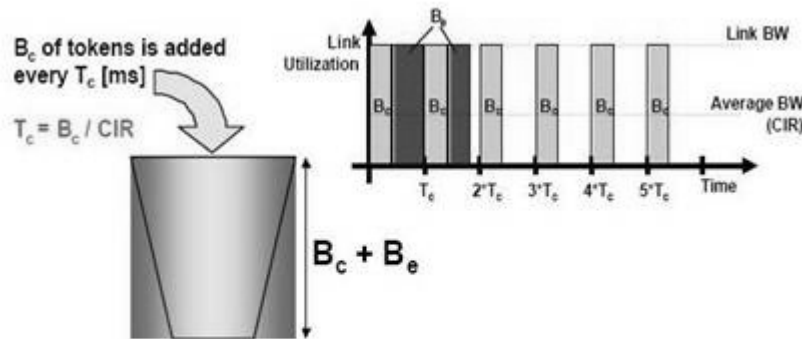


Figure 1.6

When the next packet arrives immediately after the first packet, and no new tokens have been added to the bucket (which is done periodically), the packet exceeds the rate limit. The packet size is greater than the current capacity of the bucket, and the exceed action is performed (drop in the case of pure policing, delay in the case of shaping).

In the token bucket metaphor, tokens are put into the bucket at a certain rate, which is B_c tokens every T_c seconds. The bucket itself has a specified capacity. If the bucket fills to capacity ($B_c + B_e$), it will overflow and therefore newly arriving tokens are discarded.



- B_c is normal burst size (specifies sustained rate)
- B_e is excess burst size (specifies length of burst)

Figure 1.7

In the token bucket metaphor, tokens are put into the bucket at a certain rate, which is B_c tokens every T_c seconds. The bucket itself has a specified capacity. If

the bucket fills to capacity ($B_c + B_e$), it will overflow and therefore newly arriving tokens are discarded.

1.7 Differences Between Traffic Shaping Mechanisms

	Generic Traffic Shaping	Class Based Traffic Shaping	Frame Relay Traffic Shaping
Command-Line Interface	Applies configuration on a per interface or subinterface basis	Applies configuration on a per-class basis	Applies configuration to all virtual (VCs) on an interface through inheritance mechanism
Traffic Group	“traffic group” command supported	Class-based WFQ (CBWFQ)	No “traffic group” group
Queues Supported	Weighted Fair Queueing (WFQ) per interface or subinterface	Class-based WFQ (CBWFQ)	WFQ, strict priority queue with WFQ, custom queue (CQ), priority queue (PQ), first-in first-out (FIFO) per VC

Table – 1.2

1.8 Summary

In this chapter we have gone through the brief introduction about traffic shaping. We also discussed the uses of traffic shaping and why traffic shaping is necessary now a days. In this chapter we had gone through a small tour about the ways of managing bandwidth and classification of traffic. Traffic Shaping and Traffic Policing are two QoS mechanisms that are used to limit the traffic are also discussed in this chapter. Different types of traffic shaping will be discussed in later chapters. Token Bucket, the mechanism used by traffic shaping is also discussed in this chapter.

Chapter-2

Network Components

Routing Protocols Overview

2.1 BGP – Border Gateway Protocol

Routing Protocol used to exchange routing information between networks. BGP forms a unique, unicast-based connection to each of its BGP-speaking peers. To increase the reliability of the peer connection, BGP uses TCP Port 179. BGP is a distance vector protocol. Each BGP node relies on downstream neighbors to pass along routes from their routing table; the node makes its route calculations based on those advertised routes and passes the results to upstream neighbors. BGP uses a list of AS numbers through which a packet must pass to reach the destination.

2.1.1 Path Attributes

A path attribute is a characteristic of an advertised BGP route. Some path attributes are familiar and some are non familiar.

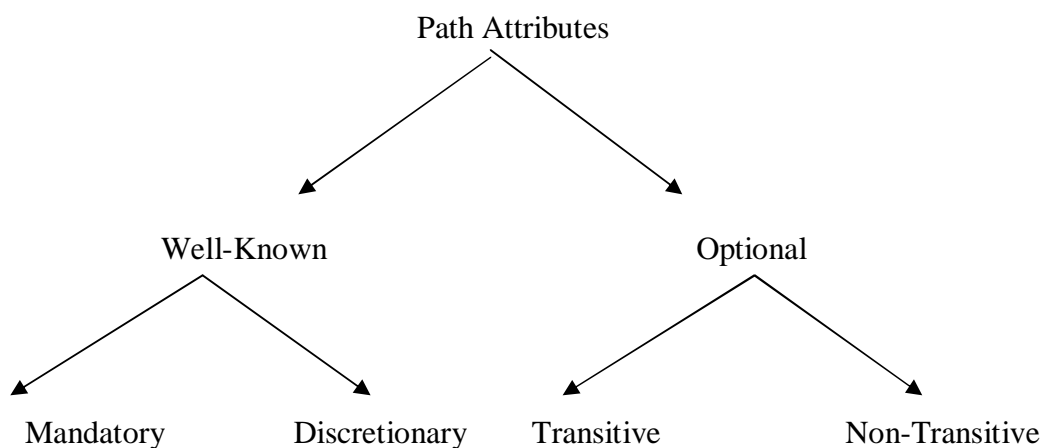


Figure – 2.1

Each path attribute falls into one of four categories:

- Well-known mandatory – These are recognized by all BGP implementations and that they must be included in all BGP Update messages.
- Well-known discretionary - These are recognized by all BGP implementations but they may or may not be sent in a specific Update message.
- Optional transitive – In this case BGP implementation is not required to support the attribute. BGP process should accept the path in which it is included, even if it doesn't support the attribute, and it should pass the path on to its peers.
- Optional nontransitive – In this case BGP implementation is not required to support the attribute. BGP process that does not recognize the attribute can quietly ignore the Update in which it is included and not advertise the path to its other peers.

BGP uses the following attributes in the route selection process:

- Weight
- Local preference
- Multi-exit discriminator
- Origin
- AS_path
- Next hop
- Community

2.1.2 BGP Path Selection

BGP could possibly receive multiple advertisements for the same route from multiple sources. BGP selects only one path as the best path. When the path is selected, BGP puts the selected path in the IP routing table and propagates the path to its neighbors. BGP uses the following criteria, in the order presented, to select a path for a destination:

- If the path specifies a next hop that is inaccessible, drop the update.
- Prefer the path with the largest weight.
- If the weights are the same, prefer the path with the largest local preference.
- If the local preferences are the same, prefer the path that was originated by BGP running on this router.
- If no route was originated, prefer the route that has the shortest AS path.

- If all paths have the same AS, path length, prefer the path with the lowest origin type (where IGP is lower than EGP, and EGP is lower than incomplete).
- If the origin codes are the same, prefer the path with the lowest MED attribute.
- If the paths have the same MED, prefer the external path over the internal path.
- If the paths are still the same, prefer the path through the closest IGP neighbor.
- Prefer the path with the lowest IP address, as specified by the BGP router ID.

2.1.3 Enabling BGP Routing

To enable BGP routing, establish a BGP routing process by using the following commands beginning in global configuration mode:

2.1.3a – Configuration Steps

Command	Purpose
1. Router(config)# router bgp <i>autonomous-system</i>	Enables a BGP routing process, which places we in router configuration mode
2. Router(config-router)# network <i>network-number</i> [mask <i>network-mask</i>]	Flags a network as local to this autonomous system and enter it to the BGP table

2.1.4 Configuring BGP Neighbors

There are two types of neighbors :

- a) Internal Neighbors - *Internal neighbors* are in the same autonomous system
- b) External Neighbors - *External neighbors* are in different autonomous systems

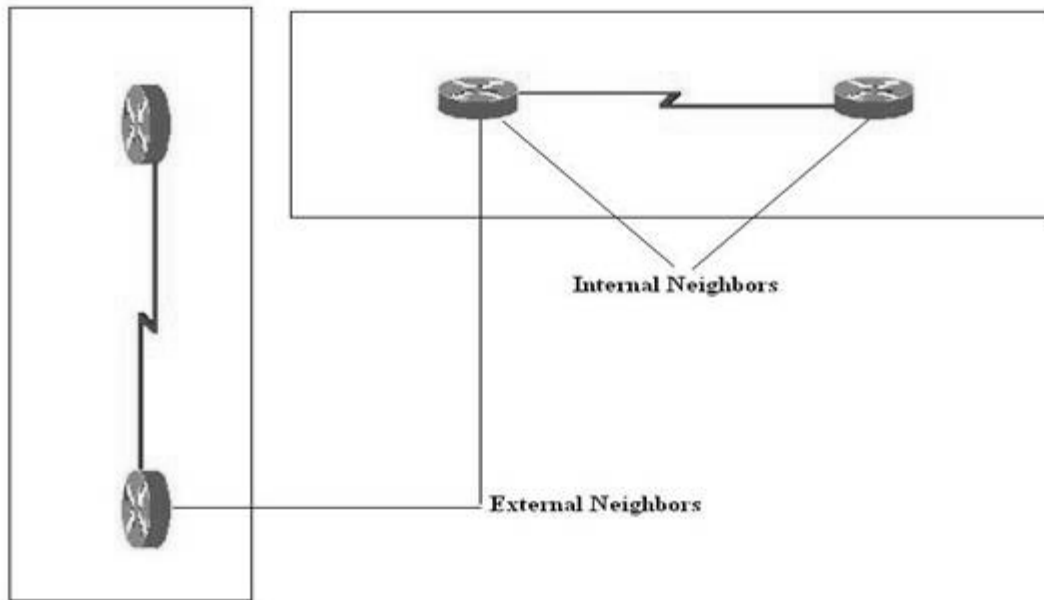


Figure – 2.2

2.1.4a – Configuration Steps

To configure BGP neighbors, use the following command, beginning in router configuration mode:

Command	Purpose
1.Router(config-router)# neighbor { <i>ip-address</i> <i>peer-group-name</i> } remote-as <i>number</i>	Specifies a BGP neighbor

2.1.5 Sending Default-Route

This command is used if we want to send default route to the neighbor. By default, we are not allowed to redistribute network 0.0.0.0. The neighbor default-originate command enables the local router to send the default route 0.0.0.0 to a neighbor. The neighbor can then use this route to reach the router if all other routes are not available. We do not need to explicitly configure network 0.0.0.0 on the router. By default, no default route forwarded to the neighbor. We gave default route in telus1, telus2 and shaw to the clients. Table 2.1 shows some part of configuration on telus2.

```
Telus2#sh runn
Building configuration...

Current configuration : 1601 bytes
!
router bgp 100
 no synchronization
```

```

bgp log-neighbor-changes
network 100.2.10.0 mask 255.255.255.252
!!
neighbor 100.2.10.2 default-originate
neighbor 100.2.10.2 route-map med4backup out
neighbor 100.2.20.2 default-originate
neighbor 100.2.20.2 route-map med4backup out
no auto-summary
!!
End

```

Table 2.1

2.1.6 Setting Local Preferences

In our scenario we had one link preferred over other. That is, one link is considered as primary and other as backup. We can achieve that setting local preference higher on one as compared to other. In the following table link with ip address 100.1.10.1 is given preference over the other. As mentioned earlier Telus ISP provides backup strategy , and this is how that strategy was implemented. Table 2.2 shows this :

```

MINT-EDM1#sh run
Building configuration...

Current configuration : 1601 bytes
!
neighbor 100.1.10.1 route-map lpref4primary out
neighbor 100.2.10.1 remote-as 100
neighbor 100.2.10.1 route-map lpref4backup out
neighbor 100.3.0.1 remote-as 1001
no auto-summary
!
ip http server
ip classless
!
!
route-map lpref4primary permit 10
set local-preference 200
!
route-map lpref4backup permit 10
set local-preference 90
!
!
End

```

Table 2.2

This things is shown in the following *Table 2.3*. Traceroute command had been issued at Calgary and packet travel through the preferred path.

```

MINT-CAL#traceroute 100.2.10.2

Type escape sequence to abort.
Tracing the route to 100.2.10.2

 1 100.5.0.1 8 msec 8 msec 8 msec
 2 100.1.10.2 [AS 100] 24 msec 24 msec *

MINT-CAL#

```

Table 2.3

2.1.7 Specifying Route Map

Route maps are used to control and modify routing information that is exchanged between routing domains. Use the route-map command in conjunction with the match and set commands to define the conditions for redistributing routes from one routing protocol to another and within the same routing protocol. Use the route-map command to define the name of the route-map and to specify permit or deny. Use the set and match commands to define the conditions for enabling policy routing or redistribution. When applied to incoming or outgoing routes, route maps enable us to control the redistribution of routes between two BGP peers. This part of the configuration is shown in *Table 2.4*

```

telus2#sh run
Building configuration...

Current configuration : 1601 bytes
!
 neighbor 100.2.10.2 default-originate
 neighbor 100.2.10.2 route-map med4backup out
 neighbor 100.2.20.2 remote-as 1001
 neighbor 100.2.20.2 description Head-Office Video Server
 neighbor 100.2.20.2 default-originate
 neighbor 100.2.20.2 route-map med4backup out
 no auto-summary
!
ip classless
!
!
ip http server
no ip http secure-server
!
!
 route-map med4backup permit 10
   set metric 200
!!
End

```

Table 2.4

2.2 OSPF – Open Short Path First

The Open Shortest Path First (OSPF) protocol is a hierarchical interior gateway protocol (IGP) for routing in Internet Protocol, although it is capable of receiving routes from and sending routes to other ASs. The largest entity within the hierarchy is the autonomous system (AS). An AS can be divided into a number of areas, which are groups of contiguous networks and attached hosts. Routers with multiple interfaces can participate in multiple areas. These routers, which are called Area Border Routers, maintain separate topological databases for each area. OSPF is perhaps the most widely-used IGP in large enterprise networks. OSPF uses a link-state in the individual areas that make up the hierarchy. A computation based on Dijkstra's algorithm is used to calculate the shortest path tree inside each area. By convention, area 0 represents the core or "backbone" region of an OSPF-enabled network. The backbone area has the identifier 0.0.0.0.

An OSPF network can be broken up into smaller networks. A special area called the backbone area forms the core of the network, and other areas are connected to it. Inter-area routing goes via the backbone. All areas must connect to the backbone; if no direct connection is possible, a virtual link may be established. A virtual link must have at least one endpoint in the backbone, and virtual links with both endpoints in the backbone can be defined to heal failures that make the backbone discontinuous .

An OSPF AS Consists of Multiple Areas Linked by Routers

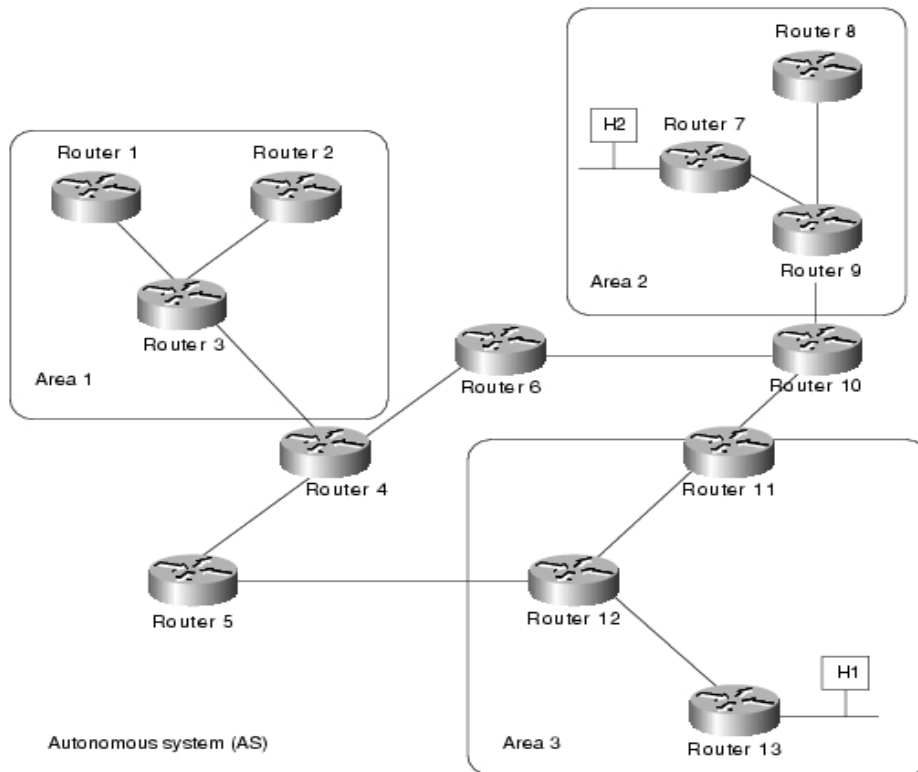


Figure – 2.3

In Figure 2.3 In the figure, routers 4, 5, 6, 10, 11, and 12 make up the backbone. If Host H1 in Area 3 wants to send a packet to Host H2 in Area 2, the packet is sent to Router 13, which forwards the packet to Router 12, which sends the packet to Router 11. Router 11 then forwards the packet along the backbone to Area Border Router 10, which sends the packet through two intra-area routers (Router 9 and Router 7) to be forwarded to Host H2. AS border routers running OSPF learn about exterior routes through exterior gateway protocols (EGPs), Border Gateway Protocol (BGP), or through configuration

2.2.1 SPF Algorithm

The Shortest Path First (SPF) routing algorithm is the basis for OSPF operations. When an SPF router is powered up, it initializes its routing-protocol data structures and then waits for indications from lower-layer protocols that its interfaces are functional.

After a router is assured that its interfaces are functioning, it uses the OSPF Hello protocol to acquire neighbors, which are routers with interfaces to a common network. The router sends hello packets to its neighbors and receives their hello packets. In addition to helping acquire neighbors, hello packets also act as keepalives to let routers know that other routers are still functional.

On multiaccess networks (networks supporting more than two routers), the Hello protocol elects a designated router and a backup designated router. Among other things, the designated router is responsible for generating LSAs for the entire multiaccess network. Designated routers allow a reduction in network traffic and in the size of the topological database.

When the link-state databases of two neighboring routers are synchronized, the routers are said to be adjacent. On multiaccess networks, the designated router determines which routers should become adjacent. Topological databases are synchronized between pairs of adjacent routers. Adjacencies control the distribution of routing-protocol packets, which are sent and received only on adjacencies.

Each router periodically sends an LSA to provide information on a router's adjacencies or to inform others when a router's state changes. By comparing established adjacencies to link states, failed routers can be detected quickly, and the network's topology can be altered appropriately. From the topological database generated from LSAs, each router calculates a shortest-path tree, with itself as root. The shortest-path tree, in turn, yields a routing table.

2.2.2 Enabling OSPF

Command	Purpose
1. Router(config)# router ospf <i>process-id</i>	Enables OSPF routing.
2. Router(config-router)# network <i>ip-address wildcard-mask</i> area <i>area-id</i>	Defines an interface on which OSPF runs and define the area ID for that interface.

Other Protocols Used:

2.3 RTP – Real Time Protocol

The Real-time Transport Protocol (or RTP) defines a standardized packet format for delivering audio and video over the Internet. Separate sessions are used for each media content (e.g. audio and video). The main advantage of this separation is to make it possible to receive only one part of the transmission, commonly audio data, which lowers the total bandwidth. RTP does not have a standard TCP or UDP port on which it communicates. The only standard that it obeys is that UDP communications are done via an even port and the next higher odd port is used for RTP Control Protocol (RTCP) communications. Although there are no standards assigned, RTP is generally configured to use ports 16384-32767. RTP can carry any data with real-time characteristics, such as interactive audio and video. RTP uses UDP. It uses a special set of messages (RTCP) to exchange periodic reports. In one RTP session, there is one media flow.

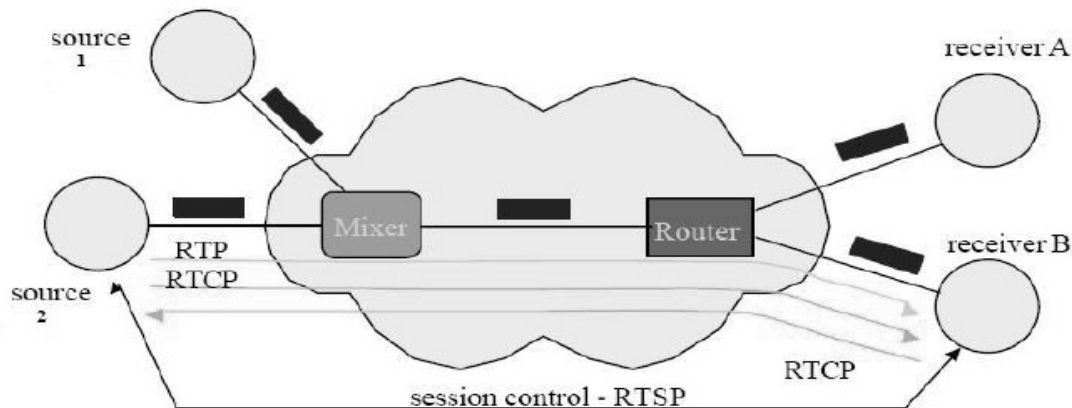


Figure – 2.4

Mixer is an intermediate system that combines RTP streams from different sources into a single stream. It can change the data format of the RTP packets.

2.4 NTP - Network Time Protocol

Network Time Protocol (NTP) allows us to synchronize our Cisco CME router to a single clock on the network, which is known as the clock master. NTP is disabled on all interfaces by default, but it is essential for Cisco CME.

The main characteristics of NTP are the following.

- fully automatic, keeps continuously the synchronization.

- suitable to synchronize one computer as well as a whole computer network.
- available on almost every type of computer.
- fault tolerant and dynamically auto-configuring.
- carrying UTC time, independent of time zones and day-light saving time.
- synchronization accuracy can reach 1 millisecond.

2.4.1 Configuration Steps :

Command	Purpose
1. Router(config)# clock timezone zone hours-offset [minutes-offset]	Sets the local time zone.
2. Router(config)# clock summer-time zone recurring [week day month hh:mm week day month hh:mm [offset]]	Specifies daylight savings time.
3. Router(config)# ntp server ip-address	Allows the clock on this router to be synchronized with the specified NTP server.



2.5 SCCP – Skinny Client Protocol

Skinny Client Control Protocol (SCCP) is a Cisco proprietary standard for terminal control for use with voice over IP (VoIP). SCCP is used between Cisco Call Manager Express and Cisco VOIP phones. It can be used as call control protocol. It has low memory and low CPU utilization. It is considered as low weight and less complex protocol. SCCP defines a simple and easy to use architecture.

With Skinny Client Control Protocol, the end stations in a network, which can be VoIP phone sets or personal computers with VoIP capability, run a program called the Skinny Client. The lightweight Skinny Client helps to minimize the cost and complexity of VoIP end stations. The audio communication between end stations makes use of the User Datagram Protocol (UDP) and the Internet Protocol (IP). Variants of SCCP are used by several companies other than Cisco.

2.6 Equipment Used

The topology discussed in this section was used as the basis for the all the experiments carried out in this project. Before going to the actual topology lets quickly go through the icons used in this report.

	Make and Series	Model Number	Picture
1.	Cisco 2600 Series Routers	2621	
2.	Cisco 2800 Series Router	2821	







3.	Cisco 3500 Series Router	3500XL	
4.	Cisco 3600 Series Router	3640	
5.	Cisco 3700 Series Router	3750G	
6.	Cisco IP Phones	7960	
7.	Windows Server	2003	
8.	Dell Notebooks	Inspiron 6000/6200	

Table 2.5

2.7 Monitoring Tools and Player Used

1. Ethereal (Version 0.10.14)
2. VLC Player (Version 3.4.5)
3. Debug and Show Commands on Routers.

2.8 Topology

2.8.1 Logical Topology

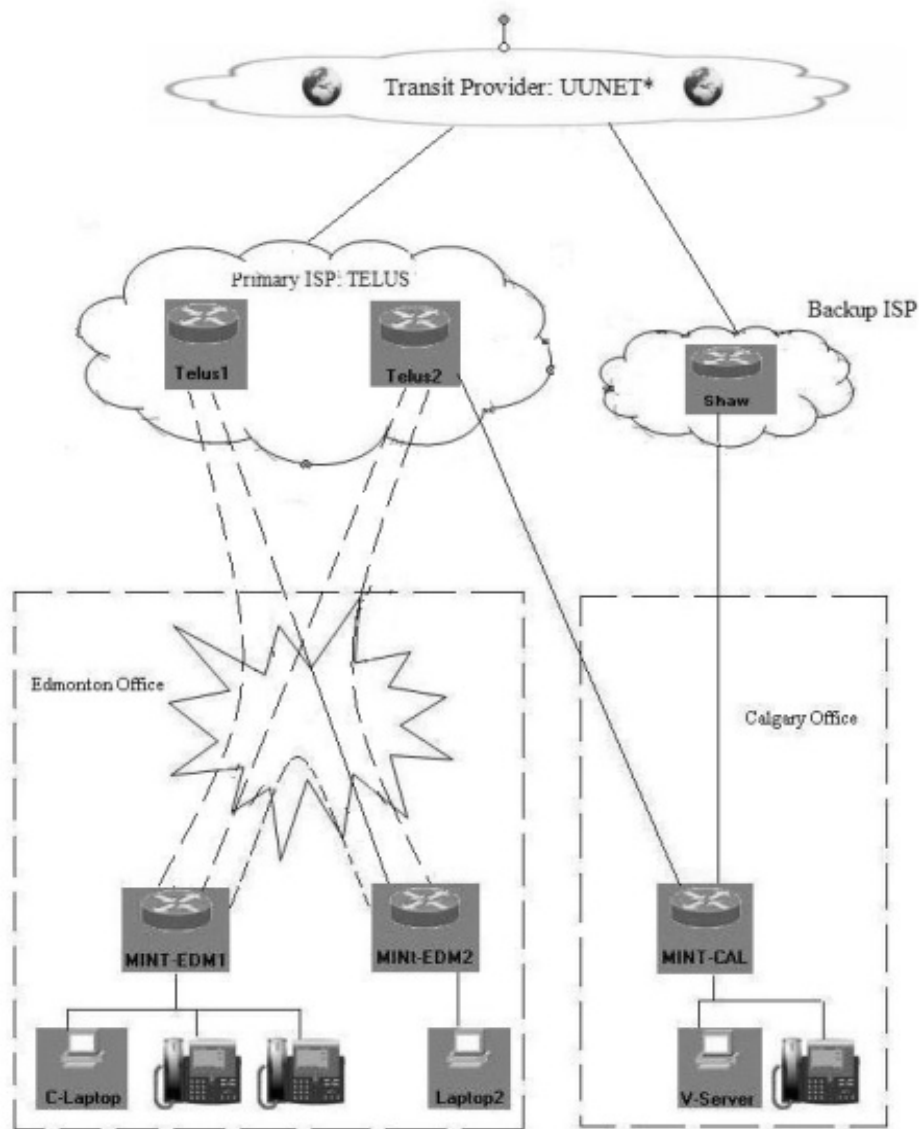


Figure – 2.5

In this network scenario we picked TELUS as primary ISP and Shaw as backup ISP. They are connected to Transit Network (UUNET). The Network Model we implemented is just like real-time scenario in which an organization named as MINT is a large enterprise , with two office in Canada . Head Office is situated in Edmonton and Branch Office is situated in Calgary. MINT has implemented video and voice strategies in the organization, so it needs enough bandwidth for both of them. Both the offices are used as receiver and transmitter for video/voice. In the contract TELUS has a SLA in which a guarantees reliable bandwidth for all kinds of data with the help of traffic shaping. Let us talk about the strategies devised by TELUS to realize these commitments.

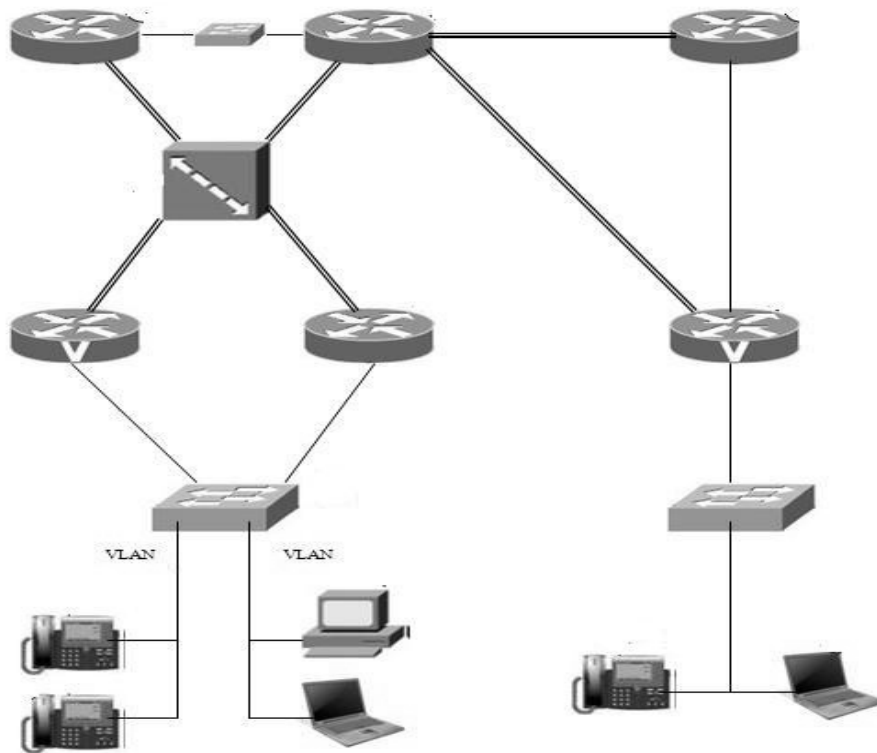
Telus has ties with another ISP named SHAW that provides backup for the Transit connection to UUNET (The Transit Provider). TELUS has provided a FRAME RELAY link to the Head Office location of MINT. MINT has two subnets, one for Data+Voice and other for Data+Video. As shown in diagram MINT-EDM1 is used as Data+Voice and MINT-EDM2 is used as Data+Video. As Frame Relay is used by TELUS, four Frame Relay PVC's connections are there, two to each subnet in MINT. So two Frame Relay PVC connects TELUS with MINT1 and two with MINT2. From which one act as primary and other for backup. That is, one of these PVCs will carry Voice+Data traffic from MINT-EDM1 and the other will carry the Video+Data traffic from the MINT-EDM2. The question that arises here is that why there are backup PVC's? The answer to that question is backup mechanism that TELUS provides to MINT, that is a part of agreement too.

Telus also implements an effective backup mechanism by providing a set of backup PVC connections to the Head office location from the Telus2 router. Thus, in case the serial link on Telus1 goes down, the data stream can failover to the backup link from Telus2. This mechanism is implemented by manipulating the BGP attributes. The Frame relay cloud has a bottleneck of 128Kbps at the FR-SW frame relay switch.

Calgary Branch Office has a primary link to the 'Telus1' router at the TELUS ISP and a backup link to the 'Shaw' router at the backup ISP, SHAW. Calgary Office has a video server and IP phone client too.

- Backing up MINT-CAL, if its serial link to Telus1 is down
- Backing up TELUS if its link to the Transit goes down.

2.8.2 The Physical Topology



2.9 IP Addressing

Following section will IP addressing that we used in the experiment. TELUS has provided the address range of 100.0.0.0/8 to the company's network. The subnets 100.3.0.0/30, 100.5.0.0/30, 100.1.10.0/30, 100.1.20.0/30, 100.2.10.0/30, 100.2.20.0/30 and 150.0.0.0/30 are used for the WAN links. SHAW has provide an IP range of 200.0.0.0/16, but is not subnetted due to lack of usage.

The following table summarizes the IP addresses scheme:

S. No.	Hostname of Switch / Router / Workstation / IP Phone	Interface / Subinterface name	IP address with CIDR	DHCP Pool Exclusion	DHCP Pool Network	Default Gateway
		Ethernet 1/0	100.0.0.1/30	NA	NA	Transit
		Serial 0/2	150.0.0.1/30	NA	NA	Transit

1	Telus1	Serial 0/1	100.5.0.1/30	NA	NA	Transit
		Serial0/0.1	100.1.20.1/30	NA	NA	Transit
		Serial0/0.2	100.1.10.1/30	NA	NA	Transit
2	Telus2	GbEthernet0/0	100.0.0.2/30	NA	NA	Transit
		Serial0/0/0.1	100.2.10.1/30	NA	NA	Transit
		Serial0/0/0.2	100.2.20.1/30	NA	NA	Transit
3	MINT-EDM1	Serial 0/1.1	100.2.10.2/30	NA	NA	100.2.10.1
		Serial 0/1.2	100.1.10.2/30	NA	NA	100.1.10.1
		Serial 0/1.3	100.3.0.2/30	NA	NA	100.1.10.1
		Ethernet 0/1	100.4.0.1/16	100.4.0.1 to 100.4.0.5	100.4.0.0 /16	100.1.10.1
4	MINT-EDM2	Serial 0/1.1	100.1.20.2/30	NA	NA	100.1.20.1
		Serial 0/1.2	100.2.20.2/30	NA	NA	100.2.10.1
		Serial 0/1.3	100.3.0.1/30	NA	NA	100.1.20.1
		Ethernet 0/1	100.6.0.1/16	NA	NA	100.1.20.1
5	Shaw	Serial 0/0/0	150.0.0.2/30	NA	NA	Transit
		GbEthernet0/1	200.0.0.1/30	NA	NA	Transit
6	MINT-CAL	Serial 0/1	100.5.0.2/30	NA	NA	100.5.0.1
		Ethernet 0/1	200.0.0.2/30	NA	NA	200.0.0.1
		Ethernet 0/0	100.8.0.1/16	100.8.0.1 to 100.8.0.5	100.8.0.0 /16	100.5.0.1
7	VLC Media Server	Ethernet	100.6.0.3	NA	NA	100.6.0.1

8	VLC Media Client	Ethernet	100.8.0.8	100.8.0.1 to 100.8.0.5	100.8.0.0 /16	100.8.0.1
9	IP Phone 1 at the Head Office	Ethernet	100.4.0.6	100.4.0.1 to 100.4.0.5	100.4.0.0 /16	100.4.0.1
10	IP Phone 1 at the Head Office	Ethernet	100.4.0.7	100.4.0.1 to 100.4.0.5	100.4.0.0 /16	100.4.0.1
11	IP Phone at the Branch Office	Ethernet	100.8.0.6	100.8.0.1 to 100.8.0.5	100.8.0.0 /16	100.8.0.1

Table 2.6- IP Addressing Scheme

2.10 Explanation of Topology Used

Frame relay is used in this project .Here are the name of the router/switches that are configured in our network.

2.10.1 Telus1 – Telus1 is an 3600 series router Telus1 and Telus2 are part of the Primary ISP of Telus. In that Telus1 is the primary having 5 links

- a) Link to MINT-EDM1 - This link is through serial0/0 . The sub interface through which it is connected to MINT-EDM1 is s0/0.2. The link to MINT-EDM1 is through frame relay cloud with DLCI 20 to the frame relay cloud and DLCI 21 from the frame relay cloud . That link is used as primary link for data + voice from MINT-EDM1.
- b) Link to MINT-EDM2 – This link is through serial0/0. The sub interface through which it is connected to MINT-EDM1 is s0/0.1. The link to MINT-EDM2 is through frame relay cloud with DLCI 22 to the frame relay cloud and DLCI 23 from the frame relay cloud . That link is used as primary link for data + voice from MINT-EDM2.
- c) Link to Telus2 – Telus1 and Telus2 are connected through switch .Telus\1 is using it’s ethernet1/0 port for that.

- d) Link to MINT-CAL – This link is through serial s0/1. This link serves as the primary link for Calgary branch to the Head-Office.
- e) Link to Shaw ISP - Shaw ISP and Telus1 are connected to each other. Telus1 uses its serial0/2 port for the same.

2.10.2 Telus2 - Telus2 is a Cisco 2800 series Integrated Services router. Telus1 and Telus2 are part of the Primary ISP of Telus. In that Telus1 have the primary links to the Telus ISP and Telus2 have the backup links for the same. Telus2 has 3 links :

- a) Link to MINT-EDM1 – This link is through serial0/0/0. The sub interface through which it is connected to MINT-EDM1 is s0/0/0.1. The link to MINT-EDM1 is through frame relay cloud with DLCI 16 to the frame relay cloud and DLCI 17 from the frame relay cloud to MINT-EDM1. That link is used as backup link for data + voice from MINT-EDM1.
- b) Link to MINT-EDM2 - This link is through serial0/0/0. The sub interface through which it is connected to MINT-EDM2 is s0/0/0.2. The link to MINT-EDM2 is through frame relay cloud with DLCI 18 to the frame relay cloud and DLCI 19 from the frame relay cloud. That link is used as backup link for data + voice from MINT-EDM2.
- c) Link to Telus1 – Telus1 and Telus2 are connected to each other through Switch. Telus1 is connected to the switch at port 24 with his Gigabit Ethernet 0/0.

2.10.3 Shaw – It is also an 2800 series router It is the ISP router for the Backup ISP named SHAW and plays the role of backing up the Primary ISP in case it goes down. It has two links :

- a) Link to Telus1 – It is the WAN link through which two ISP's are connected to each other .Shaw uses his serial 0/0/0 for this purpose.
- b) Link to MINT-CAL - -This link is through Gigabit Ethernet 0/1. This link is used only as backup link .The primary link of Calgary branch is connected to Telus1.

2.10.4 MINT-EDM1 – It is a 2600 series router .In our Lab model, it is used as a client side equipment, hosting Multi-service Voice and Data integration for the Head-office location. It connects Cisco 7960 IP phones across the WAN cloud by implementing Frame-relay based packet telephony solutions for the voice traffic between the two client equipments. It has the 3 interfaces configured. MINT-EDM1 and MINT-EDM2 are connected through frame relay switch, we can connect them directly but due to lack of ports we use frame relay to connect them. IP phones are installed on this site (EDM1) and MINT-EDM1 is also configured as VoIP gateway. CME (Call Manager Express) is used by us. We configured 3 interfaces, one interface (serial0/1) has 3 subinterfaces, loopback and one fast Ethernet :

- a) Link to Telus2 – Sub interface s0/1.1 is used to connect to Telus2. The link to Telus2 is through frame relay cloud with DLCI 17 to the frame relay cloud and DLCI 16 from the frame relay cloud. That link is used as backup link for data + voice from MINT-EDM1.

- b) Link to Telus1 - Sub interface s0/1.2 is used to connect to Telus1. The link to Telus1 is through frame relay cloud with DLCI 21 to the frame relay cloud and DLCI 20 from the frame relay cloud. That link is used as primary link for data + voice from MINT-EDM1.
- c) Link to MINT-EDM2 - Sub interface s0/1.3 is used to connect to MINT-EDM2. The link to MINT-EDM2 is through frame relay cloud with DLCI 24 to the frame relay cloud and DLCI 25 from the frame relay cloud. If MINT-EDM1 need to access MINT-EDM2 or vice versa then they don't need to go through ISP , they can directly access each other.
- d) Link to IP Phones - This interface(fast Ethernet 0/0) is connected to switch , as we use this interface to connect Cisco IP phones(7960) . In 3500 switch there are 2 Vlan's . One Vlan with the IP Address – 100.4.0.2 and other with IP Address 100.6.0.2.

Here is the capture from the MINT-EDM1 router. We excluded address 100.4.0.1 to 100.4.0.5.

The network is 100.4.0.0, from which the IP Phones can pick there IP Address. IP Phones got the following address: 100.4.0.6 and 100.4.0.10.

```

MINT-EDM1#sh ip dhcp binding
Bindings from all pools not associated with VRF:
IP address      Client-ID/      Lease expiration      Type
                Hardware address/
                User name
100.4.0.6       0100.0785.0563.1d    Mar 03 1993 05:00 AM    Automatic
100.4.0.10      0100.0750.d54f.a6    Mar 03 1993 05:04 AM    Automatic

```

Table – 2.7

```

MINT-EDM1#sh ip dhcp pool

Pool MINT :
Utilization mark (high/low)      : 100 / 0
Subnet size (first/next)         : 0 / 0
Total addresses                   : 65534
Leased addresses                  : 2
Pending event                     : none
1 subnet is currently in the pool :
Current index      IP address range      Leased addresses
100.4.0.1         100.4.0.1 - 100.4.255.254    2

```

Table – 2.8

- e) Loopback Address - We configured loopback here because of the following reasons:
 - i) Update Source - The interesting thing with BGP is that potential neighbors, or "peers", do not need to be directly connected and can use their loopback interfaces to form the peer relationships. It may well be to wer advantage to use loopbacks to form peer relationships rather than the actual interface facing the potential neighbor. This can be done because BGP uses static neighbor statements rather than any kind of dynamic neighbor discovery process.

ii) Destination Address – We have to specify the destination address in ip phone’s configuration. As in our case there are multiple paths for the voice, if we specify a serial or Ethernet interface address than it will take only that path and what will happen when that particular interface goes down? Even though the interface is up and working but the packets will not change there path. So that is the reason that we use loopback address as it never goes down.

Here is the small part of the configuration :

Configuration of MINT-EDM1, in which we mentioned the loopback address of MINT-CAL

```
dial-peer voice 1000 voip
  preference 1
  destination-pattern 2001
  session target ipv4:200.0.1.1
```

Table – 2.9

Configuration of MINT-CAL, in which we mentioned the loopback address of MINT-EDM1

```
dial-peer voice 2000 voip
  preference 1
  destination-pattern 1002
  session target ipv4:100.10.0.1
```

Table – 2.10

2.10.5 MINT-EDM2 - It is a 2600 series router . It is used as client-side equipment, hosting Multi-service Video and Data integration for the Head-Office location at Edmonton. It has the 3 interfaces configured(3 sub interfaces). MINT-EDM1 and MINT-EDM2 are connected through frame relay switch, we can connect them directly but due to lack of ports we use frame relay to connect them. We configured 3 interfaces, one interface (serial0/1) has 3 subinterfaces, loopback and one fast Ethernet :

- a) Link to Telus2 – Sub interface s0/1.2 is used to connect to Telus2. The link to Telus2 is through frame relay cloud with DLCI 19 to the frame relay cloud and DLCI 18 from the frame relay cloud. That link is used as backup link for data + voice from MINT-EDM2.
- b) Link to Telus1 - Sub interface s0/1.1 is used to connect to Telus1. The link to Telus1 is through frame relay cloud with DLCI 23 to the frame relay cloud and DLCI 22 from the frame relay cloud. That link is used as primary link for data + voice from MINT-EDM1.
- c) Link to MINT-EDM1 - Sub interface s0/1.3 is used to connect to MINT-EDM1. The link to MINT-EDM1 is through frame relay cloud with DLCI 25 to the frame relay cloud and DLCI 24 from the frame relay cloud. If MINT-EDM1 need to access MINT-EDM2 or vice versa then they don’t need to go through ISP , they can directly access each other.
- d) Link to Video Client – Fast Ethernet0/0 is used by VLC Video Client .
- e) Loopback Address – We have configured one loopback address here. The reason I already mentioned , is update source in BGP.

2.10.6 MINT-CAL - It is a 2600 series router. It is used as a client side equipment, hosting Multi-service Voice, Video and Data integration for the Branch-Office location at Calgary. It has the 3 interfaces configured(3 sub interfaces). MINT-EDM1 and MINT-EDM2 are connected through frame relay switch, we can connect them directly but due to lack of ports we use frame relay to connect them. We configured 4 interfaces, two fast Ethernet , one serial interface and one loopback .

- a) Link to Shaw – Interface fast Ethernet 0/1 is used to connect to Shaw. The link to Shaw is used as a backup only .It becomes active only when the primary link goes down.
- b) Link to Telus1 - Interface s0/1 is used to connect to Telus1. The link to Telus1 is used as primary for Calgary branch.

2.10.7 FR-SW - FR-SW is also a Cisco 3600 Series switch. . It is used as a Frame Relay Switch that establishes a Frame Relay cloud between the Primary ISP and the Head Office location. It has the following four serial connections. It has four serial links:

- a) Link to Telus1 – It has two PVC’s. DLCI’s number on that interface are 16 and 18 .
- b) Link to Telus2 – It has two PVC’s. DLCI’s number on that interface are 20 and 22 .
- c) Link to MINT-EDM1 – It has three PVC’s. DLCI’s number on that interface are 17,21 and 24.
- d) Link to MINT-EDM2 – It has three PVC’s DLCI’s number on that interface are 19,23 and 25.

2.11 Summary

In this chapter we became familiar with the components and protocol that we used in this project. BGP and OSPF are the routing protocols that we used in the project. We have given preference to one link over the other and implemented backup **statergy** on one ISP. Procedure for that is also explained with specific commands. Attributes of BGP and SPF algorithm for OSPF is also explained in this chapter. RTP, NTP and SCCP are the protocols used by voice in CISCO network. There purpose and there functionality is also well explained with aid of diagram in this chapter. In the second section of this chapter, we explained CICSO equipment that we used in our capstone project. In next section of this chapter, you will come to know the topologies, both logical and physical, that we used in the project. In the last section of this chapter IP Addressing scheme was explained.

Chapter 3

CISCO Call Manager Express

Cisco CME is an IOS based solution for call processing for small to medium sized deployments. It is used as VoIP integration solution.

Following are the steps that we have to go through before making IP Phones running.

- à Setting Up DHCP Service for Cisco CME
- à Setting IP phones with CME
- à Setting Up Initial Extensions and Phones

3.1 Setting Up DHCP Service for Cisco CME

When a Cisco IP phone is connected to the Cisco CME system, it automatically queries for a Dynamic Host Configuration Protocol (DHCP) server. The DHCP server responds by assigning an IP address to the Cisco IP phone and providing the IP address of the TFTP server through DHCP option 150. Then the phone registers with the Cisco CME server and attempts to get configuration and phone firmware files from the TFTP server.

3.1.1 – Configuration Steps :

Command	Purpose
1. Router(config)# ip dhcp pool <i>pool-name</i>	Creates a name for the DHCP server address pool
2. Router(config-dhcp)# network <i>ip-address</i> [<i>mask</i> <i>/prefix-length</i>]	Specifies the IP address of the DHCP address pool.
3. Router(config-dhcp)# option 150 ip <i>ip-address</i>	Specifies the TFTP server address from which the Cisco IP phone downloads the image configuration file.
4. Router(config-dhcp)# default-router <i>ip-address</i>	Specifies the router that the IP phones will use to send or receive IP traffic
5. Router(config-dhcp)# exit	Exits DHCP pool configuration mode.

3.2 Setting Up IP Phones with CME

In a Cisco CME system, the IP phones receive their initial configuration information and phone firmware from the TFTP server associated with the Cisco CME router. In most cases, the phones obtain the IP address of their TFTP server using the Dynamic Host Configuration Protocol (DHCP) option 150 command.

```
ip dhcp pool MINT
network 100.4.0.0 255.255.0.0
default-router 100.4.0.1
option 150 ip 100.4.0.1
domain-name Head-Office
```

Table 3.1

For Cisco CME operation, the TFTP server address obtained by the Cisco IP phones should point to the Cisco CME router IP address. The Cisco IP phones attempt to transfer a configuration file called XmlDefault.cnf.xml. This file is automatically generated by the Cisco CME router through the ip source-address command and placed in router memory. The XmlDefault.cnf.xml file contains the IP address that the phones use to register for service, using the Skinny Client Control Protocol (SCCP). This IP address should correspond to a valid Cisco CME router IP address (and may be the same as the router TFTP server address).

```
telephony-service
load 7960-7940 P00303020209
max-ephones 4
max-dn 4
ip source-address 100.4.0.1 port 2000
create cnf-files version-stamp Jan 01 2002 00:00:00

keepalive 45
```

Table 3.2

Access to the XML Default.cnf.xml file must be granted by using the tftp-server command on the router. The XMLDefault.cnf.xml file is automatically generated by the Cisco CME router with the ip source-address command and is placed in the router's flash memory.

3.2.1 - Configuration Steps

Command	Purpose
1. Router(config)# tftp-server flash:filename	Permits the Cisco CME router to provide TFTP access to the specified file by the IP phones served by the router
2. Router(config)# telephony-service	Enters telephony-service configuration mode
3. Router(config-telephony)# max-ephones max-phones	Sets the maximum number of Cisco IP phones to be supported by this router. Maximum – 48 In our case.
4. Router(config-telephony)# max-dn max-directory-numbers	Sets the maximum number of extensions (ephone-dns) to be supported by this router. Maximum – 192 in our case.
5. Router(config-telephony)# load phone-type firmware-file	Identifies the Cisco IP phone firmware file to be used by phones of the specified Cisco IP phone type when they register.
6. Router(config-telephony)# ip source-address ip-address [port port]	Identifies the IP address and port number that the Cisco CME router uses for IP phone registration. The default port is 2000.
7. Router(config-telephony)# create cnf-files	Builds the XML configuration files that are required for Cisco CME phones.
8. Router(config-telephony)# keepalive seconds	Sets the time interval, in seconds, between keepalive messages that are sent to the router by Cisco IP phones. Default is 30.
9. Router(config-telephony)# reset all	Performs a fast reboot on one or all IP phones associated with the Cisco CME router.
9. Router(config-telephony)# exit	Exits telephony-service configuration mode.

3.3 Setting Up Initial Extensions

There are three ways to create an initial phone setup in an Cisco CME system:

- Automated Phone Setup Using the Cisco CME Setup Tool
- Partially Automated Setup Using the Router CLI
- Manual Setup Using the Router CLI

In project we use manual set up using router CLI .

3.3.1 - Manual Setup Using the Router CLI

The manual process for setting up an extension involves defining an ephone-dn and the manual process for setting up a Cisco CME phone involves defining an ephone. These terms are defined as follows:

- *Ephone-dn*—A software construct that represents a line that connects a voice channel to a phone instrument where a user can receive and make calls. An ephone-dn represents a virtual voice port in the Cisco CME system, so the maximum number of ephone-dns in a Cisco CME system is the maximum number of simultaneous call connections that can occur.

- * *Ephone*—A software construct that represents a physical telephone instrument. The maximum number of ephones in a Cisco CME system is the maximum number of physical instruments that can be connected to the system.

3.3.1 a - Configuration Steps

Command	Purpose
1. Router(config)# ephone-dn <i>dn-tag</i>	Enters ephone-dn configuration mode.
2. Router(config)# number <i>number</i>	Configures a valid extension number for this ephone-dn instance.
3. Router(config-ephone-dn)# name <i>name</i>	Associates a name with this ephone-dn instance. This name is used for caller-ID displays and in the local directory listings.
4. Router(config-ephone-dn)# exit	Exits ephone-dn configuration mode.
5. Router(config)# ephone <i>phone-tag</i>	Enters ephone configuration mode.
6. Router(config-ephone)# mac-address <i>mac-address</i>	Specifies the MAC address of the IP phone that is being configured.
7. Router(config-ephone)# type <i>type</i>	Specifies the type of phone.
8. Router(config-ephone)# button <i>button-number</i>	Associates a button number and line characteristics with an extension (ephone-dn).
9. Router(config-ephone)# keepalive <i>seconds</i>	Sets the length of the time interval between successive keepalive messages from the Cisco CallManager Express router to a particular IP phone.
10 . . Router(config-ephone)# exit	Exits ephone configuration mode.

3.4 Cisco IP Phones

The majority of Cisco IP Phones provide the following features:

- Display-based user interface
- Straightforward user customization
- Inline Power over Ethernet (PoE)
- Support for the G.711 and G.729 audio codecs

3.4.1 – Types of Cisco IP Phones

Entry-Level Cisco IP Phones

Cisco has produced a number of low-cost, entry-level IP Phones for a variety of business functions. Depending on user requirements, these IP Phones may function well for employees



Figure 3.1

Entry-level Cisco phones provide the following common features:

- Display-based user interface (except Cisco IP Phone 7902G)
- G.711 and G.729 codec
- Single line (directory number [DN])
- Cisco inline power, powered patch panel, or local power option support via a power cube (the same power supply as the Cisco IP Phone 7910, 7940, or 7960)
- Visual message waiting indicator (MWI)
- One-way speakerphone (no built-in microphone) and no headset port

Midrange and Upper-End Cisco IP Phones

Cisco designed the IP Phones 7940G, 7941G and 7941G-GE, 7960, 7961G and 7961G-GE, 7970G and 7971G-GE, and 7985G to meet the demand for a corporate-level, full-featured IP Phone for medium-to-high telephone use.



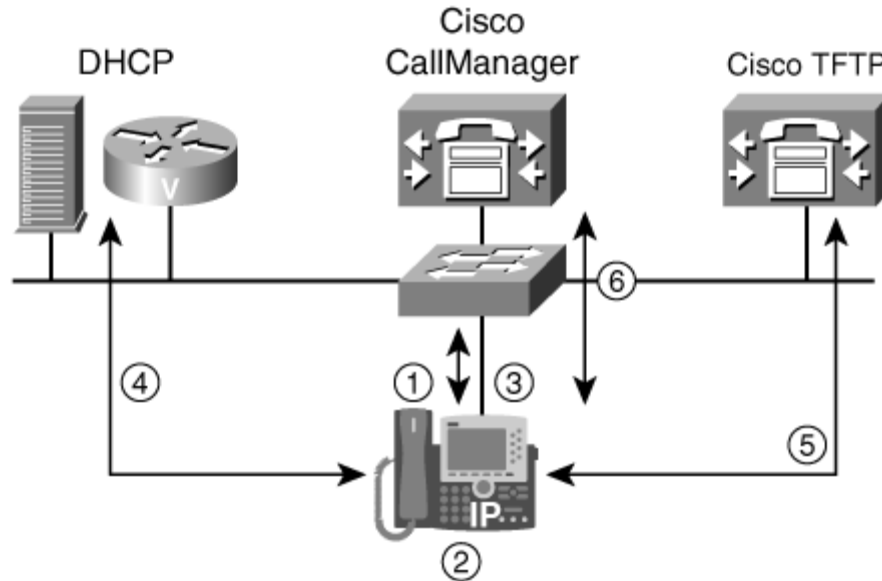
Figure 3.2

A description of features that are common to all midrange and upper-end IP Phones follows:

- Multiline capability
- Large pixel-based displays, which allow for the inclusion of XML and future features
- Integrated two-port Fast Ethernet or Gigabit Ethernet switch.
- Built-in headset connection and quality full-duplex speakerphone (does not come with a headset)
- Information key for "online" help with features
- A minimum of 24 user-adjustable ring tones
- Adjustable foot stand (flat to 60 degrees) for desktop use or appropriate kit included for wall mounting
- SCCP and SIP support
- XML service support
- An EIA/TIA-232 port for options, such as line expansion and security access

3.5 IP Phone Startup Process

Figure 3.3 provides an overview of the startup process for a Cisco IP Phone if we are using a Cisco Catalyst switch that is capable of providing Cisco pre standard Power over Ethernet (PoE).



- | |
|---|
| <ol style="list-style-type: none"> 1. IP Phone obtains power from the switch 2. Phone loads stored image 3. Switch provides VLAN information to IP Phone 4. Phone sends DHCP request; receives IP information and TFTP server address 5. IP Phone gets configuration from TFTP server 6. IP Phone registers with Cisco CallManager server |
|---|

Figure 3.3

These are the steps that normal boot of a CISCO IP Phone follows :

1. Obtain power from the switch
2. Load the stored phone image
3. Configure the VLAN
4. Obtain the IP address and TFTP server address
5. Contact the TFTP server.
6. Register with Cisco CME

3.6 Cisco IP Phone Codec Support

Before a VoIP device is able to stream audio, the analog audio signal must be converted to a digitized format. This is accomplished by an audio codec, which digitizes the audio input at the transmitting end and converts the digital stream back to analog audio at the receiving end. The International Telecommunication Union Telecommunication Standardization Sector (ITU-T) standards committee specifies several standards (called recommendations) for audio codecs. Cisco IP Phones natively support two primary codecs: G.711 and G.729. We used G.729 codecs. A G.711 conversation consumes 64 Kbps of network bandwidth (not including packet header overhead), whereas a G.729

conversation consumes 8 Kbps of network bandwidth (not including packet header overhead).

3.7 Summary

This chapter dealt with CISCO Call Manager Express and CISCO IP Phones. This chapter explains step by step procedure, to configure IP Phones in CISCO Call Manager Express. We used manual setup with router CLI, instead of automatic, so just manual procedure is explained in this chapter. In the last section of the chapter, types of CISCO IP Phones and codec used by those phones are explained.

Chapter 4

Configurations

Here are the running configurations of the router's. Till here there is no traffic shaping implemented.

4.1 Telus2's Running Configuration

```
telus2#sh run
Building configuration...

Current configuration : 1601 bytes
!
version 12.4
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname telus2
!
boot-start-marker
boot-end-marker
!
!
no aaa new-model
!
resource policy
!
ip subnet-zero
!
!
ip cef
!
!
voice-card 0
no dspfarm
!
!
interface GigabitEthernet0/0
ip address 100.0.0.2 255.255.255.252
duplex auto
speed auto
!
```

```

interface GigabitEthernet0/1
  no ip address
  shutdown
  duplex auto
  speed auto
!
interface Serial0/0/0
  bandwidth 10000000
  no ip address
  encapsulation frame-relay
  no fair-queue
!
interface Serial0/0/0.1 point-to-point
  ip address 100.2.10.1 255.255.255.252
  frame-relay interface-dlci 16
!
interface Serial0/0/0.2 point-to-point
  ip address 100.2.20.1 255.255.255.252
  frame-relay interface-dlci 18
!
router bgp 100
  no synchronization
  bgp log-neighbor-changes
  network 100.2.10.0 mask 255.255.255.252
  neighbor 100.0.0.1 remote-as 100
  neighbor 100.2.10.2 remote-as 1001
  neighbor 100.2.10.2 description Head-Office Voice Gateway
  neighbor 100.2.10.2 default-originate
  neighbor 100.2.10.2 route-map med4backup out
  neighbor 100.2.20.2 remote-as 1001
  neighbor 100.2.20.2 description Head-Office Video Server
  neighbor 100.2.20.2 default-originate
  neighbor 100.2.20.2 route-map med4backup out
  no auto-summary
!
ip classless
!
!
ip http server
no ip http secure-server
!
!
route-map med4backup permit 10
  set metric 200
!
!
control-plane
!
!
!
gateway
  timer receive-rtp 1200
!
line con 0
line aux 0
line vty 0 4
  login

```

```
!  
scheduler allocate 20000 1000  
!  
end
```

4.2 Telus1's Running Configuration

```
telus1#sh run  
Building configuration...  
  
Current configuration : 2129 bytes  
!  
version 12.2  
service timestamps debug uptime  
service timestamps log uptime  
no service password-encryption  
!  
hostname telus1  
!  
!  
ip subnet-zero  
!  
!  
ip cef  
call rsvp-sync  
!  
!  
!  
interface Serial0/0  
  bandwidth 10000000  
  no ip address  
  encapsulation frame-relay  
  no fair-queue  
  clock rate 128000  
!  
interface Serial0/0.1 point-to-point  
  bandwidth 10000000  
  ip address 100.1.20.1 255.255.255.252  
  frame-relay interface-dlci 22  
!  
interface Serial0/0.2 point-to-point  
  ip address 100.1.10.1 255.255.255.252  
  frame-relay interface-dlci 20  
!  
interface Serial0/1  
  bandwidth 10000000  
  ip address 100.5.0.1 255.255.255.252  
!  
interface Serial0/2  
  bandwidth 10000000  
  ip address 150.0.0.1 255.255.255.252
```

```

    clock rate 128000
    !
interface Serial0/3
    no ip address
    shutdown
    clock rate 64000
    !
interface Serial0/4
    no ip address
    shutdown
    !
interface Serial0/5
    no ip address
    shutdown
    !
interface Serial0/6
    no ip address
    shutdown
    !
interface Serial0/7
    no ip address
    shutdown
    !
interface Ethernet1/0
    ip address 100.0.0.1 255.255.255.252
    half-duplex
    !
interface ATM2/0
    no ip address
    shutdown
    no atm ilmi-keepalive
    !
router bgp 100
    bgp log-neighbor-changes
    network 100.0.0.0 mask 255.255.255.252
    network 100.1.10.0 mask 255.255.255.252
    network 100.1.20.0 mask 255.255.255.252
    network 100.5.0.0 mask 255.255.255.252
    network 150.0.0.0 mask 255.255.255.252
    aggregate-address 100.0.0.0 255.0.0.0 summary-only
    neighbor 100.0.0.2 remote-as 100
    neighbor 100.1.10.2 remote-as 1001
    neighbor 100.1.10.2 description Head-Office Voice Gateway (Primary
Link)
    neighbor 100.1.10.2 default-originate
    neighbor 100.1.10.2 route-map med4primary out
    neighbor 100.1.20.2 remote-as 1001
    neighbor 100.1.20.2 description Head-Office Server Gateway (Primary
Link)
    neighbor 100.1.20.2 default-originate
    neighbor 100.1.20.2 route-map med4primary out
    neighbor 100.5.0.2 remote-as 1002
    neighbor 100.5.0.2 default-originate
    neighbor 150.0.0.2 remote-as 200
    !
ip classless
ip http server

```

```

!
route-map med4primary permit 10
  set metric 100
!
!
!
dial-peer cor custom
!
!
!
gateway
!
!
line con 0
line aux 0
line vty 0 4
  login
!
end

telus1#

```

4.3 Shaw's Running Configuration

```

shaw#sh run
Building configuration...

Current configuration : 1055 bytes
!
version 12.4
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname shaw
!
boot-start-marker
boot-end-marker
!
!
no aaa new-model
!
resource policy
!
ip subnet-zero
!
!
ip cef
!
!
!
voice-card 0
  no dspfarm

```

```

!
interface GigabitEthernet0/0
  no ip address
  shutdown
  duplex auto
  speed auto
!
interface GigabitEthernet0/1
  ip address 200.0.0.1 255.255.255.252
  duplex auto
  speed auto
!
interface Serial0/0/0
  ip address 150.0.0.2 255.255.255.252
!
router bgp 200
  no synchronization
  bgp log-neighbor-changes
  network 150.0.0.0 mask 255.255.255.252
  network 200.0.0.0 mask 255.255.255.252
  aggregate-address 200.0.0.0 255.255.0.0 summary-only
  neighbor 150.0.0.1 remote-as 100
  neighbor 200.0.0.2 remote-as 1002
  neighbor 200.0.0.2 default-originate
  no auto-summary
!
ip classless
!
!
ip http server
no ip http secure-server
!
!
control-plane
!
!
line con 0
line aux 0
line vty 0 4
  login
!
scheduler allocate 20000 1000
!
end

shaw#

```

4.4 MINT-EDM1's Running Configuration

```

MINT-EDM1#sh runn
Building configuration...

Current configuration : 3153 bytes

```



```

!
version 12.2
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname MINT-EDM1
!
logging queue-limit 100
!
clock timezone mst -7
clock summer-time pst recurring
ip subnet-zero
!
!
no ip dhcp conflict logging
ip dhcp excluded-address 100.4.0.1 100.4.0.5
!
ip dhcp pool MINT
    network 100.4.0.0 255.255.0.0
    default-router 100.4.0.1
    option 150 ip 100.4.0.1
    domain-name Head-Office
!
mpls ldp logging neighbor-changes
!
!
no voice hpi capture buffer
no voice hpi capture destination
!
!
mta receive maximum-recipients 0
!
!
interface Loopback1
    ip address 100.10.0.1 255.255.255.0
!
interface FastEthernet0/0
    ip address 100.4.0.1 255.255.0.0
    duplex auto
    speed auto
!
interface Serial0/0
    no ip address
    shutdown
    no fair-queue
!
interface FastEthernet0/1
    no ip address
    shutdown
    duplex auto
    speed auto
!
interface Serial0/1
    bandwidth 10000000
    no ip address
    encapsulation frame-relay

```

```

!
interface Serial0/1.1 point-to-point
 ip address 100.2.10.2 255.255.255.252
 frame-relay interface-dlci 17
!
interface Serial0/1.2 point-to-point
 ip address 100.1.10.2 255.255.255.252
 frame-relay interface-dlci 21
!
interface Serial0/1.3 point-to-point
 ip address 100.3.0.2 255.255.255.252
 frame-relay interface-dlci 24
!
router bgp 1001
 no synchronization
 bgp log-neighbor-changes
 network 100.2.10.0 mask 255.255.255.252
 network 100.3.0.0 mask 255.255.255.252
 network 100.4.0.0 mask 255.255.0.0
 network 100.10.0.0 mask 255.255.255.0
 neighbor 100.1.10.1 remote-as 100
 neighbor 100.1.10.1 route-map lpref4primary out
 neighbor 100.2.10.1 remote-as 100
 neighbor 100.2.10.1 route-map lpref4backup out
 neighbor 100.3.0.1 remote-as 1001
 no auto-summary
!
ip http server
ip classless
!
!
!
!
route-map lpref4primary permit 10
 set local-preference 200
!
route-map lpref4backup permit 10
 set local-preference 90
!
!
tftp-server flash:P00303020209.bin
call rsvp-sync
!
voice-port 1/0/0
!
voice-port 1/0/1
!
!
mgcp profile default
!
dial-peer cor custom
!
!
!
dial-peer voice 1 pots
 destination-pattern 1002
 port 1/0/0

```

```

!
dial-peer voice 1000 voip
  preference 1
  destination-pattern 2001
  session target ipv4:200.0.1.1
!
dial-peer voice 2 pots
  destination-pattern 1012
  port 1/0/0
!
dial-peer voice 3 pots
  destination-pattern 1001
  port 1/0/1
!
dial-peer voice 4 pots
  destination-pattern 1011
  port 1/0/1
!
dial-peer voice 2000 voip
  preference 1
  destination-pattern 2011
  session target ipv4:200.0.1.1
!
gateway
!
!
telephony-service
  load 7960-7940 P00303020209
  max-ephones 4
  max-dn 4
  ip source-address 100.4.0.1 port 2000
  create cnf-files version-stamp Jan 01 2002 00:00:00
  keepalive 45
!
!
ephone-dn 1
  number 1001
  name Head Office 1
!
!
ephone-dn 2
  number 1011
  name neyota, kenny
!
!
ephone-dn 3
  number 1002
  name Head Office
!
!
ephone-dn 4
  number 1012
  name gill, gurvir
!
!
ephone 1
  keepalive 45

```

```

mac-address 0007.8505.631D
type 7960
button 2:2 3:1
!
!
!
!
ephone 2
keepalive 45
mac-address 0007.50D5.4FA6
type 7960
button 1:3 2:4
!
!
!
line con 0
line aux 0
line vty 0 4
login
!
ntp server 100.4.0.1
!
end

MINT-EDM1#

```

4.5 MINT-EDM2's Running Configuration

```

MINT-EDM2#sh run
Building configuration...

Current configuration : 1769 bytes
!
version 12.2
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname MINT-EDM2
!
logging queue-limit 100
!
ip subnet-zero
!
!
!
mpls ldp logging neighbor-changes
!
!
no voice hpi capture buffer
no voice hpi capture destination
!
!

```

```

mta receive maximum-recipients 0
!
!
!
interface Loopback1
 ip address 100.7.0.1 255.255.0.0
!
interface FastEthernet0/0
 ip address 100.6.0.1 255.255.0.0
 duplex auto
 speed auto
!
interface Serial0/0
 no ip address
 shutdown
 no fair-queue
!
interface Serial0/1
 bandwidth 10000000
 no ip address
 encapsulation frame-relay
 clockrate 128000
!
interface Serial0/1.1 point-to-point
 bandwidth 10000000
 ip address 100.1.20.2 255.255.255.252
 frame-relay interface-dlci 23
!
interface Serial0/1.2 point-to-point
 ip address 100.2.20.2 255.255.255.252
 frame-relay interface-dlci 19
!
interface Serial0/1.3 point-to-point
 ip address 100.3.0.1 255.255.255.252
 frame-relay interface-dlci 25
!
router bgp 1001
 no synchronization
 bgp log-neighbor-changes
 network 100.2.20.0 mask 255.255.255.252
 network 100.3.0.0 mask 255.255.255.252
 network 100.6.0.0 mask 255.255.0.0
 network 100.7.0.0 mask 255.255.0.0
 neighbor 100.1.20.1 remote-as 100
 neighbor 100.1.20.1 route-map lpref4primary out
 neighbor 100.2.20.1 remote-as 100
 neighbor 100.2.20.1 route-map lpref4backup out
 neighbor 100.3.0.2 remote-as 1001
 no auto-summary
!
ip http server
ip classless
!
!
route-map lpref4primary permit 10
 set local-preference 200
!

```

```

route-map lpref4backup permit 10
  set local-preference 90
!
!
call rsvp-sync
!
voice-port 1/0/0
!
voice-port 1/0/1
!
!
mgcp profile default
!
dial-peer cor custom
!
!
line con 0
line aux 0
line vty 0 4
  login
!
!
end

MINT-EDM2#

```

4.6 MINT-CAL's Running Configuration

```

MINT-CAL#sh runn
Building configuration...

Current configuration : 3012 bytes
!
version 12.2
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname MINT-CAL
!
logging queue-limit 100
!
clock timezone mst -7
clock summer-time pst recurring
ip subnet-zero
!
!
no ip dhcp conflict logging
ip dhcp excluded-address 100.8.0.1 100.8.0.5
!
ip dhcp pool MINT2

```

```

    network 100.8.0.0 255.255.0.0
    default-router 100.8.0.1
    option 150 ip 100.8.0.1
    domain-name Branch-Office
!
mpls ldp logging neighbor-changes
!
!
!
no voice hpi capture buffer
no voice hpi capture destination
!
!
mta receive maximum-recipients 0
!
!
interface Loopback0
  ip address 4.4.4.4 255.255.255.255
!
interface Loopback1
  ip address 200.0.1.1 255.255.255.0
!
interface FastEthernet0/0
  ip address 100.8.0.1 255.255.0.0
  duplex auto
  speed auto
!
interface Serial0/0
  no ip address
  shutdown
  no fair-queue
!
interface FastEthernet0/1
  ip address 200.0.0.2 255.255.255.252
  duplex auto
  speed auto
!
interface Serial0/1
  bandwidth 10000000
  ip address 100.5.0.2 255.255.255.252
  clockrate 128000
!
router ospf 1
  log-adjacency-changes
  network 4.4.4.4 0.0.0.0 area 2
  network 192.168.2.0 0.0.0.255 area 2
!
router bgp 1002
  no synchronization
  bgp log-neighbor-changes
  network 100.5.0.0 mask 255.255.255.252
  network 100.8.0.0 mask 255.255.0.0
  network 200.0.0.0 mask 255.255.255.252
  network 200.0.1.0
  neighbor 100.5.0.1 remote-as 100
  neighbor 100.5.0.1 route-map lpref4telus out
  neighbor 200.0.0.1 remote-as 200

```

```

neighbor 200.0.0.1 route-map lpref4shaw out
no auto-summary
!
ip http server
ip classless
!
!
!
!
route-map lpref4telus permit 10
  set local-preference 200
!
route-map lpref4shaw permit 10
  set local-preference 90
!
!
tftp-server flash:P00303020209.bin
call rsvp-sync
!
voice-port 1/0/0
!
voice-port 1/0/1
!
!
mgcp profile default
!
dial-peer cor custom
!
!
!
dial-peer voice 1 pots
  destination-pattern 2001
  port 1/0/0
!
dial-peer voice 2000 voip
  preference 1
  destination-pattern 1002
  session target ipv4:100.10.0.1
!
dial-peer voice 2 pots
  destination-pattern 2011
  port 1/0/0
!
dial-peer voice 3000 voip
  preference 2
  destination-pattern 1012
  session target ipv4:100.10.0.1
!
dial-peer voice 4000 voip
  preference 3
  destination-pattern 1001
  session target ipv4:100.10.0.1
!
dial-peer voice 5000 voip
  preference 4
  destination-pattern 1011
  session target ipv4:100.10.0.1

```



```

!
!
telephony-service
 load 7960-7940 P00303020209
 max-ephones 4
 max-dn 4
 ip source-address 100.8.0.1 port 2000
 create cnf-files version-stamp Jan 01 2002 00:00:00
 keepalive 45
!
!
ephone-dn 1
 number 2001
 name Branch Office
!
!
ephone-dn 2
 number 2011
 name neyota, kenny
!
!
ephone-dn 3
 number 2002
 name Branch Office
!
!
ephone-dn 4
 number 2012
 name gill, gurvir
!
!
ephone 1
 keepalive 45
 mac-address 0007.5079.BF61
 type 7960
 button 1:1 2:2
!
!
!
!
ephone 2
 keepalive 45
 mac-address 0007.8513.13B8
 type 7960
 button 1:3 2:4
!
!
!
!
line con 0
line aux 0
line vty 0 4
 login
!
ntp server 100.8.0.1
!
end

```

MINT-CAL#

4.7 FR-SW's Running Configuration

```
FR-SW#sh run
Building configuration...

Current configuration : 1579 bytes
!
version 12.2
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname FR-SW
!
!
ip subnet-zero
!
!
!
frame-relay switching
call rsvp-sync
!
!
!
interface Ethernet0/0
  no ip address
  shutdown
  half-duplex
!
interface Serial1/0
  bandwidth 10000000
  no ip address
  encapsulation frame-relay
  no fair-queue
  frame-relay intf-type dce
  frame-relay route 20 interface Serial1/2 21
  frame-relay route 22 interface Serial1/3 23
!
interface Serial1/1
  bandwidth 10000000
  no ip address
  encapsulation frame-relay
  clock rate 128000
  frame-relay intf-type dce
  frame-relay route 16 interface Serial1/2 17
  frame-relay route 18 interface Serial1/3 19
!
interface Serial1/2
```

```

bandwidth 10000000
no ip address
encapsulation frame-relay
clock rate 128000
frame-relay intf-type dce
frame-relay route 17 interface Serial1/1 16
frame-relay route 21 interface Serial1/0 20
frame-relay route 24 interface Serial1/3 25
!
interface Serial1/3
bandwidth 10000000
no ip address
encapsulation frame-relay
frame-relay intf-type dce
frame-relay route 19 interface Serial1/1 18
frame-relay route 23 interface Serial1/0 22
frame-relay route 25 interface Serial1/2 24
!
interface Serial1/4
no ip address
shutdown
!
interface Serial1/5
no ip address
shutdown
!
interface Serial1/6
no ip address
shutdown
!
interface Serial1/7
no ip address
shutdown
!
interface ATM2/0
no ip address
shutdown
no atm ilmi-keepalive
!
ip classless
ip http server
!
!
!
dial-peer cor custom
!
!
!
!
line con 0
line aux 0
line vty 0 4
login
!
end

```

4.7 Summary

This chapter is the backbone of the whole project. Most of our detailed studies are applied in this chapter. This chapter has all the running configurations of routers and switches. It requires our great efforts to make the project as real time and make it running properly.

Chapter 5

Generic Traffic Shaping

5.1 Generic Traffic Shaping

GTS is one of the Traffic Shaping mechanism. Generic Traffic Shaping (GTS) shapes traffic by reducing the outbound traffic flow to avoid congestion. This is achieved by constraining traffic to a particular bit rate using the token bucket mechanism. Token Bucket mechanism was explained in Chapter 1. GTS is applied on a per-interface basis and can use access lists to select the traffic to shape. It works with a variety of Layer-2 technologies, including Frame Relay, ATM, Switched Multi-megabit Data Service (SMDS) and Ethernet.

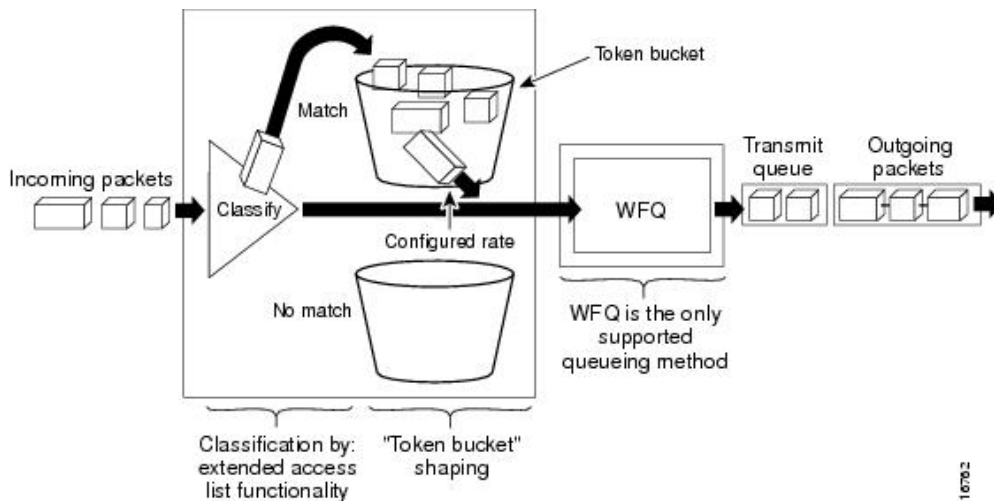


Figure – 5.1

With GTS parameters configured to match the network architecture, downstream congestion can be avoided, eliminating bottlenecks in topologies with data-rate mismatches.

GTS has the following characteristics:

- à Rate enforcement on a per interface, or subinterface, basis—the mean rate can be set to match the circuit CIR or some other value.
- à Traffic selection using Access Control Lists (ACLs).
- à GTS works on many Layer 2 interface types, including Frame Relay, ATM, Switched Multimegabit Data Service (SMDS), and Ethernet.
- à It supports BECN messages for bandwidth throttling.

à It supports WFQ per sub-interface.

GTS works with the token bucket algorithm in the following way. When packets arrive at the router, an interrupt occurs. If the queue is empty, GTS consults the credit manager (token bucket) to see if there is enough credit to send the packet. If there is not, the packet is sent to the queue configured, in this case, WFQ. If there is credit available, the packet is sent to the output interface, and the associated credit value is deducted from the token bucket. Queued packets are serviced at regular time intervals. The credit manager is checked each time interval to determine if there is enough credit to transmit the next packet waiting in queue. If there is, the packet is sent to the output interface, and the VC is charged the appropriate number of credits.

Function(s) of :-

- Classifier – It is used for classification, so that different traffic classes can have different policies applied to them. It can shape multiple classes.
- Meter – It is used for measuring traffic rate of individual classes. Metering, using a token-bucket mechanism, used to distinguish between conforming and exceeding traffic
- Shaper – It is used to shape traffic. It delays packets of exceeding classes and shape it to the configured rate limit.

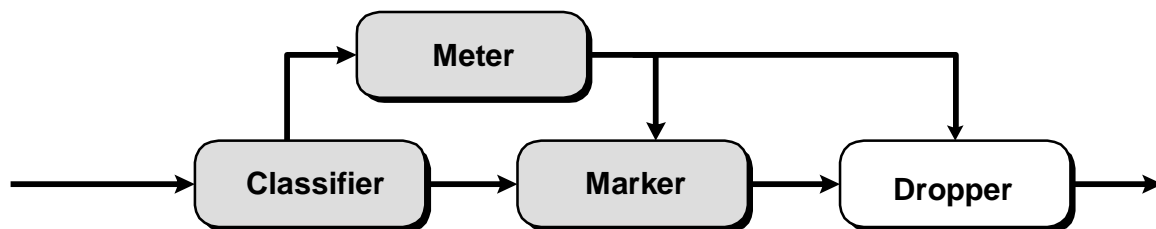


Figure – 5.2

5.2 Features of GTS

- GTS applies parameters per sub-interface
- GTS is multi-protocol
- GTS supports traffic group command
- GTS uses WFQ as the shaping
- GTS can be implemented in any queuing mechanisms:
 - FIFO Queuing
 - Priority Queuing (PQ)
 - Custom Queuing (CQ)
 - Weighted Fair Queuing (WFQ)
- GTS works on output only.
- GTS works on variety of interface types.

The GTS implementation in Cisco IOS supports multiple protocols variety of interface types. WFQ is used as the shaping delay queue, scheduling within a traffic class. Other queuing strategies (FIFO, WFQ) may be employed after GTS to provide traffic scheduling traffic. Also, GTS only works at the output of an interface.

GTS can be used to shape all outbound traffic on an interface or it can separately shape multiple classes. Classification is performed using any type of access list

5.3 Weighted Fair Queuing (WFQ)

WFQ is a simple to implement, dynamic queuing mechanism which ensures that every conversation in the network gets a fair share of the bandwidth. WFQ is turned on by default on links. WFQ dynamically classifies network traffic into individual flows and assigns each flow a fair share of the total bandwidth. Each flow is classified as a high bandwidth or low bandwidth flow. Low bandwidth flows, such as Telnet, get priority over high bandwidth flows such as UTP. If multiple high bandwidth flows occur simultaneously, they share the remaining bandwidth evenly once low bandwidth flows have been serviced. Each of these flows is placed into an individual queue that follows the leaky bucket analogy. If packets from a specific flow exceed the capacity of the queue to which it is allocated, that queue is subject to tail drops like all other queues. WFQ dynamically adapts to changes in the network, including new protocols and applications. If there is no mission-critical traffic that must be given priority over other traffic, WFQ is an easy and efficient method to provide the best level of service to every network user.

5.3.1 How WFQ Work?

WFQ first identifies each individual flow and classifies it as a high or low bandwidth Flow.

Protocol	WFQ Flow Identification Fields
TCP/IP	IP Protocol Source IP address Destination IP address Source port Destination port Type of service (ToS) field
Appletalk	Source network, node and socket Destination network, node and socket Protocol type
IPX	Source network, host and socket Destination network, host and socket Level 2 protocol type
DECnet	Source address Destination address
Frame relay	DLCI value

Transparent Bridging	Source and destination MAC address
CLNS	Source NSAP Destination NSAP
Apollo	Source network, host and socket Destination network, host and socket Level 2 protocol type=
All others	Control protocols (one per queue)

Table 5.1

Once classified, the flows are placed in a fair queue. The default number of dynamic queues is 256. Each queue is serviced in a round-robin fashion, like custom queuing, with service priority being given to low bandwidth queues. Each queue is configured with a default congestive discard threshold that limits the number of messages in each queue. The default congestive discard value for each queue is 64 packets. For high bandwidth flows, messages attempting to enter the queue once the discard threshold is reached are discarded. Low bandwidth messages, however, can still enter the queue even though the congestive discard threshold is exceeded for that queue. The limits for dynamic queues and the congestive discard threshold can both be adjusted up to a value of 4096 packets. Aside from the differentiation between low- and high-speed flows, these conversations are not given any priority or weight over one another.

In WFQ, the priority given to network traffic is inversely proportional to the signal bandwidth. Thus, narrowband signals are passed along first, and broadband signals are buffered. WFQ has little or no effect on the speed at which narrowband signals are transmitted, but tends to slow down the transmission of broadband signals, especially during times of peak network traffic. Broadband signals share the resources that remain after low-bandwidth signals have been transmitted. The resource sharing is done according to assigned weights. In flow-based WFQ, also called standard WFQ, packets are classified into flows according to one of four criteria: the source Internet Protocol address (IP address), the destination IP address, the source Transmission Control Protocol (TCP) or User Datagram Protocol (UDP) port, or the destination TCP or UDP port. Each flow receives an equal allocation of network bandwidth .

5.3.2 Restrictions

Weighted fair queuing is not compatible with :

- à X.25
- à Synchronous Data Link Control (SDLC)
- à Link Access Procedure, Balanced (LAPB)
- à Tunnel, loopback, dialer, bridged, and virtual interfaces

5.3.3 Configuring WFQ

In this scenario WFQ was used as Queue Mechanism. In *Table 5.2* the output of “**show interface interface-type interface-number**” is shown:

```
shaw#sh interfaces gigabitEthernet 0/1
GigabitEthernet0/1 is up, line protocol is up
  Hardware is MV96340 Ethernet, address is 0017.e069.24c1 (bia 0017.e069.24c1)
  Internet address is 200.0.0.1/30
  MTU 1500 bytes, BW 100000 Kbit, DLY 100 usec,
    reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation ARPA, loopback not set
  Keepalive set (10 sec)
  Full-duplex, 100Mb/s, media type is T
  output flow-control is XON, input flow-control is XON
  ARP type: ARPA, ARP Timeout 04:00:00
  Last input 00:00:37, output 00:00:04, output hang never
  Last clearing of "show interface" counters never
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
  Queueing strategy: weighted fair
  Output queue: 0/1000/64/0 (size/max total/threshold/drops)
    Conversations 0/1/256 (active/max active/max total)
    Reserved Conversations 0/0 (allocated/max allocated)
    Available Bandwidth 75000 kilobits/sec
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
    16765 packets input, 12471580 bytes, 0 no buffer
    Received 2995 broadcasts, 0 runts, 0 giants, 0 throttles
    0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored
    0 watchdog, 0 multicast, 0 pause input
    0 input packets with dribble condition detected
  29609 packets output, 5503616 bytes, 0 underruns
    0 output errors, 0 collisions, 1 interface resets
    0 babbles, 0 late collision, 0 deferred
    0 lost carrier, 0 no carrier, 0 pause output
    0 output buffer failures, 0 output buffers swapped out
shaw#
```

5.4 Configuring GTS

There are two methods to configure GTS :

- a) With “ **traffic-shape rate**” commands.
- b) With “ **traffic-shape group**” commands

But they cannot be mixed on the same interface

- a) Traffic-Shape Rate Commands --

```
Router(config-if)#traffic-shape rate bit-rate [burst-size [excess
burst-size]]
```

5.5 Configure GTS for an Access List

Traffic-Shape Group Commands --

```
Router(config-if)#traffic-shape group access-list bit-rate  
[burst[excess-burst]]
```

To configure GTS for outbound traffic on an access list, use the following commands beginning in global configuration mode:

5.5.1 Configuration Steps

Command	Purpose
1. Router(config)# access-list <i>access-list-number</i>	Assign traffic to an access list
2. Router(config)# interface <i>interface-type</i> <i>interface-number</i>	Enter interface configuration mode
3. Router(config-if)# traffic--shape group <i>access-list bit-rate [burst-size [excess-burst-size]]</i>	Configure traffic shaping for outbound traffic on an interface for the specified access list

Repeat Steps 1 through 3 for each type of traffic we want to rate-limit.
Access lists are used to apply traffic shaping to a particular data . (i.e tcp , udp)

5.6 Network Topology for GTS

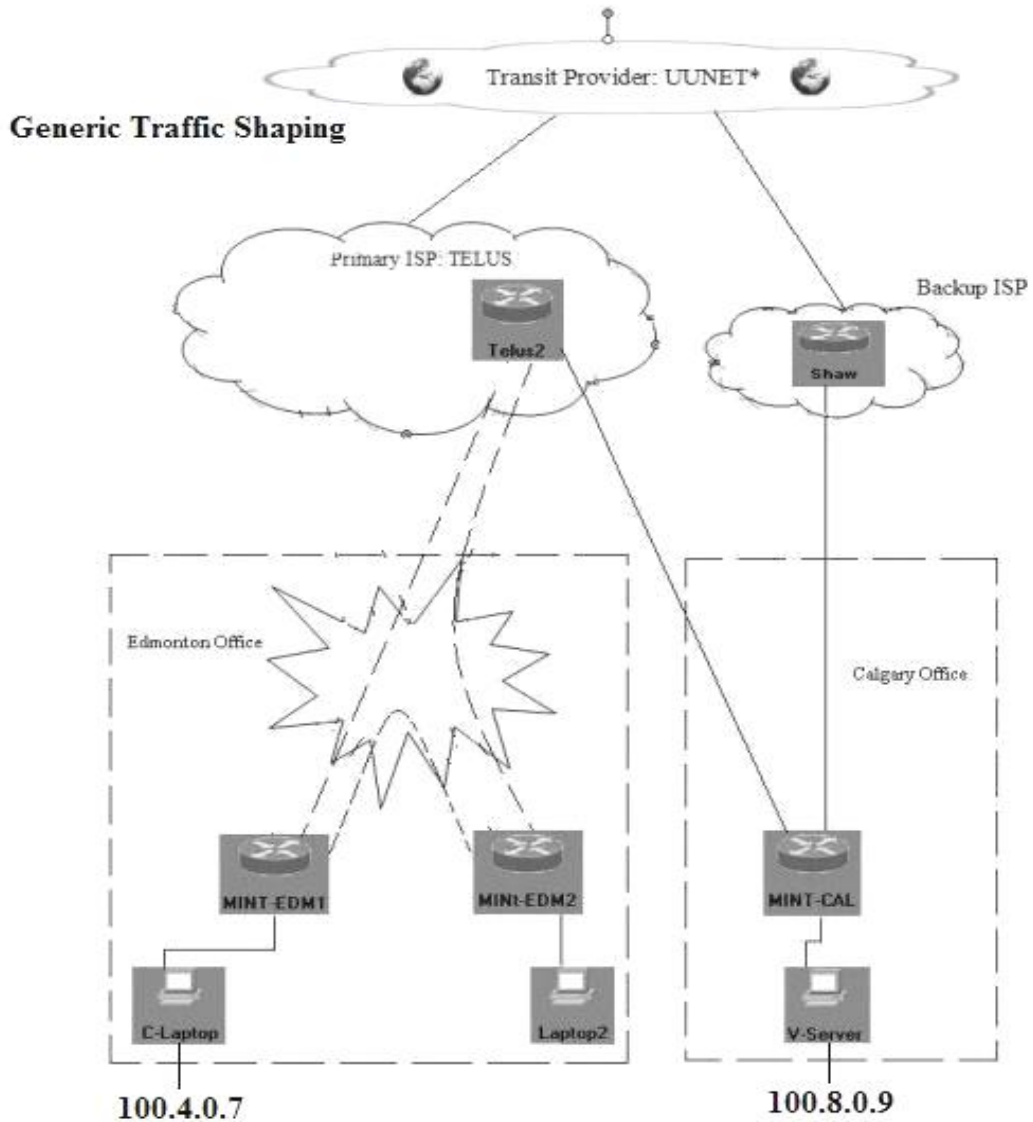


Figure 5.3

5.7 – Configuration of “Shaw” with GTS – Case 1

```
shaw#sh
Building configuration...

Current configuration : 1299 bytes
!
!
```

```

interface GigabitEthernet0/1
 ip address 200.0.0.1 255.255.255.252 → CIR → Be → Buffer Limit
 duplex auto
 speed auto
 traffic-shape rate 12800 7936 0 1000 → Bc
!
!
end

shaw#

```

5.8 – Statistics – Case I

5.8.1 – Statistics 1

We had implemented GTS(Generic Traffic Shaping) and the configuration for that is shown above with GTS parameters (marked with arrows and pointers)

Now here is the captures for GTS :

```
shaw#sh traffic-shape
```

Interface	Access	Target	Byte	Sustain	Excess	Interval	Increment	Adapt
VC	List	Rate	Limit	bits/int	bits/int	(ms)	(bytes)	Active
-		12800	992	7936	0	661	992	-

Table 5.2

5.8.2 – Statistics 2 – No Video Played Till Now

```
shaw#sh traffic-shape statistics
```

I/F	Acc.	Queue	Packets	Bytes	Packets	Bytes	Shaping
	List	Depth			Delayed	Delayed	Active
Gi0/1		0	170	15613	0	0	no

Table 5.3

As shown from the statistics, Shaping was NOT Active. Because till now there was no queue ,as there was no data send till now .

Next statistics was after the data was send in bulk. The queue starts building and traffic shaping becomes active.

5.8.3 – Statistics 3 – Video Streaming Was On

shaw#sh traffic-shape statistics							
I/F	Acc. Queue	Packets	Bytes	Packets	Bytes	Shaping	
Gi0/1	List Depth			Delayed	Delayed	Active	
	64	269	111558	79	92827	yes	
shaw#sh traffic-shape statistics							
I/F	Acc. Queue	Packets	Bytes	Packets	Bytes	Shaping	
Gi0/1	List Depth			Delayed	Delayed	Active	
	64	331	185025	141	166294	yes	
shaw#sh traffic-shape statistics							
I/F	Acc. Queue	Packets	Bytes	Packets	Bytes	Shaping	
Gi0/1	List Depth			Delayed	Delayed	Active	
	55	352	211175	162	192444	yes	
shaw#sh traffic-shape statistics							
I/F	Acc. Queue	Packets	Bytes	Packets	Bytes	Shaping	
Gi0/1	List Depth			Delayed	Delayed	Active	
	25	389	253012	199	234281	yes	

Table 5.4

From the statistics one thing was clear that the traffic shaping becomes active and queue starts building up. Packets are delayed in the network, there is NO priority given to any data.

After the video was stopped , traffic shaping again becomes non-active.

5.8.4 – Statistics 4 – Video Streaming Was Off

shaw#sh traffic-shape statistics							
I/F	Acc. Queue	Packets	Bytes	Packets	Bytes	Shaping	
Gi0/1	List Depth			Delayed	Delayed	Active	
	15	546	453012	457	434281	yes	
shaw#sh traffic-shape statistics							
I/F	Acc. Queue	Packets	Bytes	Packets	Bytes	Shaping	
Gi0/1	List Depth			Delayed	Delayed	Active	
	0	985	948024	777	924968	no	

Table 5.5

5.9 Summary of Statistics 1

S. NO	Queue Depth	Status of Traffic Shaping
1.	0	No
2.	64	Yes
3.	55	Yes
4.	45	Yes
5.	25	Yes
6.	15	Yes
7.	0	No

Table – 5.6

From *Table 5.6* it is clear that when queue start building then traffic shaping became active. For some time the traffic shaping does not became active, the reason is that during that interval of time the queue is building up. When queue becomes full then traffic shaping comes into play. In this the buffer limit was set to “1000”. When there were no packets in the queue, traffic shaping was Non-Active. As soon as queue becomes full, traffic shaping becomes Active. The source in this experiment was Video. When video was stopped then no more new packets came into the queue and after certain interval of time that depth of that queue start decreasing. When queue becomes empty, then status of traffic shaping again becomes Non-Active. This is shown in *Table – 5.6*

5.10 Statistics 1 from Ethereal

Source: 154.0.0.2 (Windows 2003 Server)

Destination: 100.8.0.7 (Dell Notebook)

Player: VLC

No. -	Time	Source	Destination	Protocol	Info
1	0.000000	154.0.0.2	100.8.0.7	UDP	Source port: 1099 Destination port: 1234
2	0.663969	154.0.0.2	100.8.0.7	UDP	Source port: 1099 Destination port: 1234
3	1.327995	154.0.0.2	100.8.0.7	UDP	Source port: 1099 Destination port: 1234
4	2.644136	154.0.0.2	100.8.0.7	UDP	Source port: 1099 Destination port: 1234
5	3.308008	154.0.0.2	100.8.0.7	UDP	Source port: 1099 Destination port: 1234
6	4.628178	154.0.0.2	100.8.0.7	UDP	Source port: 1099 Destination port: 1234
7	5.292047	154.0.0.2	100.8.0.7	UDP	Source port: 1099 Destination port: 1234
8	6.608122	154.0.0.2	100.8.0.7	UDP	Source port: 1099 Destination port: 1234
9	6.656679	Cisco_96:f3:00	Cisco_96:f3:00	LOOP	Reply
10	7.272080	154.0.0.2	100.8.0.7	UDP	Source port: 1099 Destination port: 1234
11	7.877051	100.8.0.1	Broadcast	ARP	who has 100.8.0.6? Tell 100.8.0.1
12	7.936047	154.0.0.2	100.8.0.7	UDP	Source port: 1099 Destination port: 1234
13	9.252194	154.0.0.2	100.8.0.7	UDP	Source port: 1099 Destination port: 1234
14	9.916239	154.0.0.2	100.8.0.7	UDP	Source port: 1099 Destination port: 1234
15	10.017483	100.8.0.7	100.8.255.255	BROWSE	Domain/workgroup Announcement WORKGROUP,
16	11.236195	154.0.0.2	100.8.0.7	UDP	Source port: 1099 Destination port: 1234
17	11.900248	154.0.0.2	100.8.0.7	UDP	Source port: 1099 Destination port: 1234
18	12.561236	154.0.0.2	100.8.0.7	UDP	Source port: 1099 Destination port: 1234
19	13.880215	154.0.0.2	100.8.0.7	UDP	Source port: 1099 Destination port: 1234
20	14.544317	154.0.0.2	100.8.0.7	UDP	Source port: 1099 Destination port: 1234
21	15.864452	154.0.0.2	100.8.0.7	UDP	Source port: 1099 Destination port: 1234
22	16.528301	154.0.0.2	100.8.0.7	UDP	Source port: 1099 Destination port: 1234
23	16.657146	100.8.0.1	100.8.0.1	LOOP	Reply
24	17.573488	100.8.0.1	Broadcast	ARP	who has 100.8.0.6? Tell 100.8.0.1
25	17.848333	154.0.0.2	100.8.0.7	UDP	Source port: 1099 Destination port: 1234
26	18.512244	154.0.0.2	100.8.0.7	UDP	Source port: 1099 Destination port: 1234
27	19.176396	154.0.0.2	100.8.0.7	UDP	Source port: 1099 Destination port: 1234
28	20.492432	154.0.0.2	100.8.0.7	UDP	Source port: 1099 Destination port: 1234
29	21.156292	154.0.0.2	100.8.0.7	UDP	Source port: 1099 Destination port: 1234
30	22.472357	154.0.0.2	100.8.0.7	UDP	Source port: 1099 Destination port: 1234
31	23.136482	154.0.0.2	100.8.0.7	UDP	Source port: 1099 Destination port: 1234
32	23.800405	154.0.0.2	100.8.0.7	UDP	Source port: 1099 Destination port: 1234
33	25.116429	154.0.0.2	100.8.0.7	UDP	Source port: 1099 Destination port: 1234
34	25.780493	154.0.0.2	100.8.0.7	UDP	Source port: 1099 Destination port: 1234
35	26.657529	100.8.0.1	100.8.0.1	LOOP	Reply
36	27.100505	154.0.0.2	100.8.0.7	UDP	Source port: 1099 Destination port: 1234

```

Frame 1 (1370 bytes on wire (1370 bytes captured)
Arrival time: Jul 27, 2007 16:15:43.030419000
[Time delta from previous packet: 0.000000000 seconds]
[Time since reference or first frame: 0.000000000 seconds]
Frame Number: 1
Packet Length: 1370 bytes
Capture Length: 1370 bytes
[Protocols in frame: eth:ip:udp:data]
Ethernet II, Src: 100.8.0.1 (00:08:21:96:f3:00), Dst: Dell_dc:f8:b4 (00:12:3f:dc:f8:b4)
Destination: Dell_dc:f8:b4 (00:12:3f:dc:f8:b4)
Source: 100.8.0.1 (00:08:21:96:f3:00)
Type: IP (0x0800)
Internet Protocol, Src: 154.0.0.2 (154.0.0.2), Dst: 100.8.0.7 (100.8.0.7)
Version: 4
Header length: 20 bytes
Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
  0000 00.. = Differentiated Services Codepoint: Default (0x00)
  .... ..0. = ECN-Capable Transport (ECT): 0
  .... ...0 = ECN-CE: 0
Total Length: 1356
Identification: 0x297a (10618)
Flags: 0x00
Fragment offset: 0
Time to live: 126
Protocol: UDP (0x11)
Header checksum: 0x1016 [correct]
Source: 154.0.0.2 (154.0.0.2)
Destination: 100.8.0.7 (100.8.0.7)
User Datagram Protocol, Src Port: 1099 (1099), Dst Port: 1234 (1234)
Source port: 1099 (1099)
Destination port: 1234 (1234)
Length: 1336
Checksum: 0x777c [correct]
Data (1328 bytes)
0000 00 12 3f dc f8 b4 00 08 21 96 f3 00 08 00 45 00  ..?....!....E.
0010 05 4c 29 7a 00 00 7e 11 10 16 9a 00 00 02 64 08  .L)z...~.....d.
0020 00 07 04 4b 04 d2 05 38 77 7c 80 21 42 86 94 05  ...K...8 w|.!B...
0030 57 b4 00 00 78 1b 47 40 0a 15 00 00 01 e0 02 c8  w...x.G@ .....
0040 80 c0 0a 3d 50 13 d3 67 1d 50 13 d3 67 00 00 01  ...=P..g .P..g...
0050 b6 7f ff ff ff ff ff ff ff ff ff ff ff ff ff
Ethernet (eth), 14 bytes
P: 44 D: 44 M: 0

```

5.11 TELUS1's Running Configuration with GTS -Case 2

```

telus1#sh runn
Building configuration...

Current configuration : 2229 bytes
!
!
interface Serial0/0.2 point-to-point
 ip address 100.1.10.1 255.255.255.252
 traffic-shape rate 12800 7936 0 1000
 frame-relay interface-dlci 20
!
!
End

```

5.12 Statistics – Case 2

5.12.1 – Statistics 1

telus1#sh traffic-shape

Interface	Se0/0.2	Access	Target	Byte	Sustain	Excess	Interval	Increment	Adapt
VC	List	Rate	Limit	bits/int	bits/int	(ms)	(bytes)	Active	
-		12800	992	7936	0	661	992	-	

Table 5.7

5.12.2 – Statistics 2 – Video was played and then stopped

In this case GTS is implemented in the network. Video was played from Calgary Office and Edmonton Office was the client for that video.

telus1#sh traffic-shape statistics									
Shaping	Access	Queue	Packets	Bytes	Packets	Bytes			
I/F	List	Depth			Delayed	Delayed	Active		
Se0/0.2		64	97	108350	81	105169	yes		
telus1#sh traffic-shape statistics									
Shaping	Access	Queue	Packets	Bytes	Packets	Bytes			
I/F	List	Depth			Delayed	Delayed	Active		
Se0/0.2		40	163	194435	147	191254	yes		
telus1#sh traffic-shape statistics									
Shaping	Access	Queue	Packets	Bytes	Packets	Bytes			
I/F	List	Depth			Delayed	Delayed	Active		
Se0/0.2		29	174	209395	158	206214	yes		
telus1#sh traffic-shape statistics									
Shaping	Access	Queue	Packets	Bytes	Packets	Bytes			
I/F	List	Depth			Delayed	Delayed	Active		
Se0/0.2		17	189	225885	173	222704	yes		
telus1#sh traffic-shape statistics									
Shaping	Access	Queue	Packets	Bytes	Packets	Bytes			
I/F	List	Depth			Delayed	Delayed	Active		
Se0/0.2		7	199	239485	183	236304	yes		
telus1#sh traffic-shape statistics									
Shaping	Access	Queue	Packets	Bytes	Packets	Bytes			
I/F	List	Depth			Delayed	Delayed	Active		
Se0/0.2		1	205	247645	189	244464	yes		
telus1#sh traffic-shape statistics									
Shaping	Access	Queue	Packets	Bytes	Packets	Bytes			
I/F	List	Depth			Delayed	Delayed	Active		
Se0/0.2		0	206	249005	190	245824	no		
telus1#sh traffic-shape statistics									
Shaping	Access	Queue	Packets	Bytes	Packets	Bytes			
I/F	List	Depth			Delayed	Delayed	Active		

Se0/0.2	0	206	249005	190	245824	no
---------	---	-----	--------	-----	--------	----

Table 5.8

5.13 Summary of Statistics 2

S.No	Queue Depth	Status of Traffic Shaping
1.	64	Yes
2.	40	Yes
3.	29	Yes
4.	17	Yes
5.	1	Yes
6.	0	No

Table – 5.9

In this experiment video was played for some time and then stopped .When video was running the queue starts building up and packets starts being delayed. Due to queue ,the traffic shaping becomes Active and when the video was stopped ,even then the traffic shaping remain Active, for some time and when the queue depth becomes Zero the traffic shaping becomes Non-Active. This was shown in captures as well as in *Table – 5.9*

5.14 Statistics 2 from Ethereal

No. -	Time	Source	Destination	Protocol	Info
1	0.000000	100.8.0.9	100.4.0.7	UDP	Source port: 1115 Destination port: 1234
2	0.632595	100.8.0.9	100.4.0.7	UDP	Source port: 1115 Destination port: 1234
3	1.918572	100.8.0.9	100.4.0.7	UDP	Source port: 1115 Destination port: 1234
4	2.587860	100.8.0.9	100.4.0.7	UDP	Source port: 1115 Destination port: 1234
5	3.317008	100.8.0.9	100.4.0.7	UDP	Source port: 1115 Destination port: 1234
6	4.603867	100.8.0.9	100.4.0.7	UDP	Source port: 1115 Destination port: 1234
7	5.242843	100.8.0.9	100.4.0.7	UDP	Source port: 1115 Destination port: 1234
8	6.618787	100.8.0.9	100.4.0.7	UDP	Source port: 1115 Destination port: 1234
9	7.272636	100.8.0.9	100.4.0.7	UDP	Source port: 1115 Destination port: 1234
10	7.912015	100.8.0.9	100.4.0.7	UDP	Source port: 1115 Destination port: 1234
11	9.200508	100.8.0.9	100.4.0.7	UDP	Source port: 1115 Destination port: 1234
12	9.836420	100.8.0.9	100.4.0.7	UDP	Source port: 1115 Destination port: 1234
13	10.568428	100.8.0.9	100.4.0.7	UDP	Source port: 1115 Destination port: 1234
14	11.842560	100.8.0.9	100.4.0.7	UDP	Source port: 1115 Destination port: 1234
15	12.493656	100.8.0.9	100.4.0.7	UDP	Source port: 1115 Destination port: 1234
16	13.890365	100.8.0.9	100.4.0.7	UDP	Source port: 1115 Destination port: 1234
17	14.528240	100.8.0.9	100.4.0.7	UDP	Source port: 1115 Destination port: 1234
18	15.165936	100.8.0.9	100.4.0.7	UDP	Source port: 1115 Destination port: 1234
19	16.466283	100.8.0.9	100.4.0.7	UDP	Source port: 1115 Destination port: 1234
20	17.189796	100.8.0.9	100.4.0.7	UDP	Source port: 1115 Destination port: 1234
21	17.834010	100.8.0.9	100.4.0.7	UDP	Source port: 1115 Destination port: 1234
22	19.135324	100.8.0.9	100.4.0.7	UDP	Source port: 1115 Destination port: 1234
23	19.778088	100.8.0.9	100.4.0.7	UDP	Source port: 1115 Destination port: 1234
24	21.110624	100.8.0.9	100.4.0.7	UDP	Source port: 1115 Destination port: 1234
25	21.819125	100.8.0.9	100.4.0.7	UDP	Source port: 1115 Destination port: 1234
26	23.041318	Cisco_94:72:07	CDP/VTP	CDP	Cisco Discovery Protocol
27	23.123944	100.8.0.9	100.4.0.7	UDP	Source port: 1115 Destination port: 1234
28	23.774195	100.8.0.9	100.4.0.7	UDP	Source port: 1115 Destination port: 1234
29	25.122534	100.8.0.9	100.4.0.7	UDP	Source port: 1115 Destination port: 1234
30	25.747188	100.8.0.9	100.4.0.7	UDP	Source port: 1115 Destination port: 1234
31	27.043117	100.8.0.9	100.4.0.7	UDP	Source port: 1115 Destination port: 1234
32	27.695615	100.8.0.9	100.4.0.7	UDP	Source port: 1115 Destination port: 1234
33	29.104636	100.8.0.9	100.4.0.7	UDP	Source port: 1115 Destination port: 1234
34	29.739530	100.8.0.9	100.4.0.7	UDP	Source port: 1115 Destination port: 1234
35	31.076661	100.8.0.9	100.4.0.7	UDP	Source port: 1115 Destination port: 1234

```

# Frame 8 (1370 bytes on wire, 1370 bytes captured)
# Ethernet II, Src: Cisco_96:f2:c0 (00:08:21:96:f2:c0), Dst: Dell_dc:f8:b4 (00:12:3f:dc:f8:b4)
0000 00 12 3f dc f8 b4 00 08 21 96 f2 c0 08 00 45 00  ..?....!.....E.
0010 05 4c 76 69 00 00 7d 11 fa 1b 64 08 00 09 64 04  .Lvj.). .d...d.
0020 00 07 04 5b 04 d2 05 38 27 6d 80 21 38 5f 1b 70  ...[..8 'm.!8..p
0030 64 b0 00 00 2b fa 47 00 0a 1c ff ff ff ff ff ff  d...+.G. ....
0040 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  .....
0050 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff

```

5.15 Conclusion

Generic Traffic Shaping was implemented in this scenario. Weighted Fair Queue was the mechanism that we used in this experiment. Our purpose was to create a queue ,so that traffic shaping becomes active. We used WFQ, according to which, the priority given to network traffic is inversely proportional to the signal bandwidth. In this experiment both video and voice were active. Voice needs less bandwidth as compared to video. So according to the theory there should not be loss of voice traffic. The voice should get the priority and the remaining bandwidth should be used by video, if that bandwidth was not enough then queue must built up and after queue gets filled there will be a delay/dropping of packets. In 1st part of the experiment, GTS was implemented on Primary ISP (Telus1). CIR = 12800 bps. i.e not enough for video. In 1st part there was no video played, just the voice, using IP Phones. The codec that was used in the project is – G.729. This codec needs only 8bps(not including header) of bandwidth. In this part just voice was there . From **Table 5.3** it is clear that there are packets coming but queue is empty so that traffic shaping is non-active. But after that in **Table 5.4**, when video was played between Edmonton and Calgary Office , Queue starts building up and traffic shaping becomes active. The reason for that is, we have configured GTS in network with CIR = 12800bps. This bandwidth was not enough for good quality of video. We use video with 2500kps. As soon as video was played queue builds up and traffic shaping comes to action. But we have WFQ mechanism in this experiment , so voice should not been affected with this as

according to theory of WFQ : Low bandwidth flows get priority over high bandwidth flows . If multiple high bandwidth flows occur simultaneously, they share the remaining bandwidth evenly once low bandwidth flows have been serviced. We have to see that whether the voice was affected (loss/latency) with this or not. From our observations there was no loss/latency in the voice. This can be seen from *Table 5.10* .

```

MINT-EDM1#debug ephone statistics mac address 0007.8505.631D
*Mar  2 20:25:48.556: ephone-1[1]:GetCallStats line 2 ref 16 DN 2: 1011
*Mar  2 20:25:48.808: ephone-1[1]:Call Stats for line 2 DN 2 1011 ref
16
*Mar  2 20:25:48.808: ephone-1[1]:TX Pkts 692 bytes 22144 RX Pkts 685
bytes 21891
*Mar  2 20:25:48.808: ephone-1[1]:Pkts lost 0 jitter 0 latency 0
*Mar  2 20:25:48.808: ephone-1[1]:Src 100.4.0.6 22414 Dst 100.4.0.1
2000 bytes 20 vad 250 G729
*Mar  2 20:25:52.420: STATS: DN 2 Packets Sent 692
*Mar  2 20:25:52.420: STATS: DN 2 Packets Received 685
*Mar  2 20:25:52.420: STATS: DN 2 Latency 0
*Mar  2 20:25:52.420: STATS: DN 2 PacketsLost 0
*Mar  2 20:25:53.600: ephone-1[1]:GetCallStats line 2 ref 16 DN 2: 1011
*Mar  2 20:25:53.852: ephone-1[1]:Call Stats for line 2 DN 2 1011 ref
16
*Mar  2 20:25:53.852: ephone-1[1]:TX Pkts 945 bytes 30240 RX Pkts 938
bytes 29987
*Mar  2 20:25:53.852: ephone-1[1]:Pkts lost 0 jitter 0 latency 0
*Mar  2 20:25:53.852: ephone-1[1]:Src 100.4.0.6 22414 Dst 100.4.0.1
2000 bytes 20 vad 250 G729
*Mar  2 20:25:57.376: STATS: DN 2 Packets Sent 945
*Mar  2 20:25:57.376: STATS: DN 2 Packets Received 938
*Mar  2 20:25:57.376: STATS: DN 2 Latency 0
*Mar  2 20:25:57.376: STATS: DN 2 PacketsLost 0

```

Table 5.10

Table 5.10 shows the call statistics on MINT-EDM1. These statistics were taken at the time when traffic shaping was active and there is a queue and packets were being delayed. But we can see from this table that it does not affect the voice at all. That was the principle of WFQ in GTS.

```

MINT-CAL#debug ephone statistics mac address 0007.5079.BF61
*Mar  5 02:02:19.063: ephone-1[1]:GetCallStats line 2 ref 20 DN 2: 2011
*Mar  5 02:02:19.315: ephone-1[1]:Call Stats for line 2 DN 2 2011 ref
20
*Mar  5 02:02:19.315: ephone-1[1]:TX Pkts 8915 bytes 285165 RX Pkts
6769 bytes 216275
*Mar  5 02:02:19.315: ephone-1[1]:Pkts lost 0 jitter 550 latency 0
*Mar  5 02:02:19.315: ephone-1[1]:Src 100.8.0.6 21406 Dst 100.8.0.1
2000 bytes 20 vad 250 G729
*Mar  5 02:02:23.339: STATS: DN 2 Packets Sent 8915
*Mar  5 02:02:23.339: STATS: DN 2 Packets Received 6769
*Mar  5 02:02:23.339: STATS: DN 2 Latency 0
*Mar  5 02:02:23.339: STATS: DN 2 PacketsLost 0
*Mar  5 02:02:24.107: ephone-1[1]:GetCallStats line 2 ref 20 DN 2: 2011

```

```
*Mar 5 02:02:24.359: ephone-1[1]:Call Stats for line 2 DN 2 2011 ref
20
*Mar 5 02:02:24.359: ephone-1[1]:TX Pkts 9167 bytes 293229 RX Pkts
6771 bytes 216301
*Mar 5 02:02:24.359: ephone-1[1]:Pkts lost 0 jitter 566 latency 0
*Mar 5 02:02:24.359: ephone-1[1]:Src 100.8.0.6 21406 Dst 100.8.0.1
2000 bytes 20 vad 250 G729
*Mar 5 02:02:26.867: STATS: DN 2 Packets Sent 9167
*Mar 5 02:02:26.867: STATS: DN 2 Packets Received 6771
*Mar 5 02:02:26.867: STATS: DN 2 Latency 0
*Mar 5 02:02:26.867: STATS: DN 2 PacketsLost 0
```

Chapter 6

Class Based Traffic Shaping

6.1 Class-Based Traffic Shaping

Class-Based Traffic Shaping is a traffic shaping mechanism. A traffic shaper typically delays excess traffic using a buffer, or queuing mechanism, to hold packets and shape the flow when the data rate of the source is higher than expected. It holds and shapes traffic to a particular bit rate by using the token bucket mechanism. See Token Bucket in section 1.6

6.2 Class-Based Weighted Fair Queuing

CBWFQ is a very powerful congestion management mechanism that is offered by Cisco for its router platforms. WFQ gives certain types of traffic preferential treatment when congestion occurs on a low-speed serial link. WFQ accomplishes this by automatically detecting conversations and guaranteeing that no one conversation monopolizes the link. But WFQ has some scaling limitations.

The algorithm runs into problems as traffic increases or if it is stressed by many conversations. Additionally, it does not run on high-speed interfaces such as ATM. Class-based weighted fair queuing (CBWFQ) was developed to overcome these factors. CBWFQ carries the WFQ algorithm further by allowing user defined classes, which allow greater control over traffic queuing and bandwidth allocation. CBWFQ provides the power and ease of configuration of WFQ, along with the flexibility of custom queuing.

6.2.1 Why CBWFQ – Limitations of WFQ

The limitation of flow-based WFQ is that the flows are automatically determined, and each flow gets a fair share of the bandwidth. This “fair share” of the bandwidth is determined by the size of the flow and moderated by IP precedence. Packets with IP precedences set to values other than the default (zero) are placed into queues that are serviced more frequently, based on the level of IP precedence, and thus get a higher overall bandwidth. Specifically, a data stream’s

weighting is the result of some complex calculations, but the important thing to remember is that weight is a relative number and the lower the weight of a packet, the higher that packet's priority. The weight calculation results in a weight, but the most important thing isn't that number—it's the packet's specific handling. Thus, a data stream with a precedence of 1 is dealt with twice as fast as best-effort traffic. However, even with the action of IP Precedence on WFQ, sometimes a specific bandwidth needs to be guaranteed to a certain type of traffic. CBWFQ fulfills this requirement.

6.2.2 How CBWFQ Works ?

CBWFQ include user-defined classes. These classes can be determined by protocol, Access Control Lists (ACLs), IP precedence, or input interfaces. Each class has a separate queue, and all packets found to match the criteria for a particular class are assigned to that queue. Once the matching criteria are set for the classes, we can determine how packets belonging to that class will be handled. It may be tempting to think of classes as having priority over each other, but it is more accurate to think of each class having a certain guaranteed share of the bandwidth. A guarantee of bandwidth that is active only during periods of congestion. If a class is not using the bandwidth guaranteed to it, other traffic may use it. Similarly, if the class needs more bandwidth than the allocated amount, it may use any free bandwidth available on the circuit. CBWFQ allows the creation of up to 64 individual classes plus a default class. The number and size of the classes are, of course, based on the bandwidth.

Each user-defined class is guaranteed a certain bandwidth, but classes that exceed that bandwidth are not necessarily dropped. Traffic in excess of the class's guaranteed bandwidth may use the "free" bandwidth on the link. "Free" is defined as the circuit bandwidth minus the portion of the guaranteed bandwidth currently being used by all user-defined classes. Within this "free" bandwidth, the packets are considered by fair queuing along with other packets, their weight being based on the proportion of the total bandwidth that was guaranteed to the class.

All packets not falling into one of the defined classes are considered part of the default class (or class-default, as it appears in the router configuration). The default class can be configured to have a set bandwidth like other user-defined classes, or configured to use flow-based WFQ in the remaining bandwidth and treated as best effort. The default configuration of the default class is dependent on the router platform and the IOS revision.

6.2.3 Advantages of CBTS

- Configure traffic shaping on a per-traffic-class basis. It allows we to fine-tune traffic shaping for one or more classes and it allows we to configure traffic shaping on a more granular level.

- Specify average rate or peak rate traffic shaping. Specifying peak rate shaping allows us to make better use of available bandwidth by allowing more data than the configured traffic shaping rate to be sent if the bandwidth is available.
- Configure traffic shaping in a hierarchical policy map structure. That is, traffic shaping is configured in a primary-level (parent) policy map and other QoS features (for instance, CBWFQ and traffic policing) can be configured in the secondary-level (child) policy maps.

6.3 CBTS With CBWFQ

With the Class-Based Traffic Shaping mechanism, traffic shaping can be configured in a hierarchical policy map structure; that is, traffic shaping is enabled in a primary-level (parent) policy map and other QoS features used with traffic shaping, such as CBWFQ and traffic policing, can be enabled in a secondary-level (child) policy map.

By default the queue type in CBTS is WFQ. To configure CBWFQ hierarchical policy has to be created. CBWFQ allows to fine-tune the way traffic is placed in a queue. For instance, we can specify that all voice traffic be placed in a high-priority queue and all traffic from a specific class be placed in a lower-priority queue.

To use CBWFQ with the Class-Based Traffic Shaping mechanism, the following steps has to be done :

1. A secondary-level (child) policy map must be created. This secondary-level (child) policy map is then used to configure CBWFQ by enabling the bandwidth command.
2. Traffic shaping must be configured in the primary-level (parent) policy map.

6.4 Configuring CBWFQ

In this scenario CBWFQ was used as Queue Mechanism. In *Table 6.1* the output of “**show interface interface-type interface-number**” is shown:

```
telus1#sh interfaces serial 0/0
Serial0/0 is up, line protocol is up
  Hardware is CD2430 in sync mode
  MTU 1500 bytes, BW 10000000 Kbit, DLY 20000 usec,
    reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation FRAME-RELAY, loopback not set
  Keepalive set (10 sec)
  LMI enq sent 248, LMI stat rcvd 248, LMI upd rcvd 0, DTE LMI up
  LMI enq rcvd 0, LMI stat sent 0, LMI upd sent 0
  LMI DLCI 1023 LMI type is CISCO frame relay DTE
```

FR SVC disabled, LAPF state down Broadcast queue 0/64, broadcasts sent/dropped 43/0, interface broadcasts 0 Last input 00:00:03, output 00:00:03, output hang never Last clearing of "show interface" counters 00:41:23 Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 6473 Queueing strategy: weighted fair Output queue: 0/1000/64/1893 (size/max total/threshold/drops) Conversations 0/2/256 (active/max active/max total) Reserved Conversations 2/2 (allocated/max allocated) Available Bandwidth 7500000 kilobits/sec 5 minute input rate 0 bits/sec, 0 packets/sec 5 minute output rate 0 bits/sec, 0 packets/sec 41538 packets input, 2708220 bytes, 0 no buffer Received 0 broadcasts, 0 runts, 0 giants, 0 throttles 0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort 47001 packets output, 14295146 bytes, 0 underruns 0 output errors, 0 collisions, 1 interface resets 0 output buffer failures, 0 output buffers swapped out 0 carrier transitions DCD=up DSR=up DTR=up RTS=up CTS=up telus1#

Table 6.1

6.5 Configuring CBTS With CBWFQ

Before configuring traffic shaping, we must use the MQC(Modular Quality of Service Command-Line Interface) to create a policy map and a class map.

Following are the sequence of the steps that are used to for MQC :

1. Creating a traffic Class
2. Creating a traffic policy
3. Attaching a policy to interface.

6.5.1 Configuration – Creating a Traffic Class

Command	Purpose
1. Router(config)# class-map [match-all match-any] class-name	Creates a class to be used with a class map
2. Router(config-cmap)# match ip rtp starting-port-number port-range	Configures a class map to use the Real-Time Protocol (RTP) protocol port as the match criterion.
3. Router(config-cmap)# exit	Exits class-map configuration mode.

6.5.2 Configuration - Creating a Traffic Policy

Command	Purpose
1. Router(config)# policy-map <i>policy-name</i>	Creates or specifies the name of the traffic policy and enters policy-map configuration mode.
2. Router(config-pmap)# class <i>class-name</i>	Specifies the name of a traffic class (previously created in 7.5.1a)
3. Router(config-pmap-c)# priority { <i>bandwidth-kbps</i> percent <i>percentage</i> } [<i>burst</i>]	Gives priority to a class of traffic belonging to a policy map.
4. Router(config-pmap-c)# shape [average peak] <i>mean-rate</i> [[<i>burst-size</i>] [<i>excess-burst-size</i>]]	Uses a service policy as a QoS policy within a policy map i.e hierarchical service policy.
6 . Router(config-pmap)# exit	Returns to privileged EXEC mode.

6.5.3 Attaching a Traffic Policy to an Interface

Command	Purpose
1. Router(config)# interface <i>interface-type</i>	Configures an interface type and enters interface configuration mode.
2. Router(config-if)# service-policy output <i>policy-map-name</i>	Attaches a policy map to an interface.
3. Router (config-if)# exit	Exits interface configuration mode.

6.6 Network Topology for CBTS

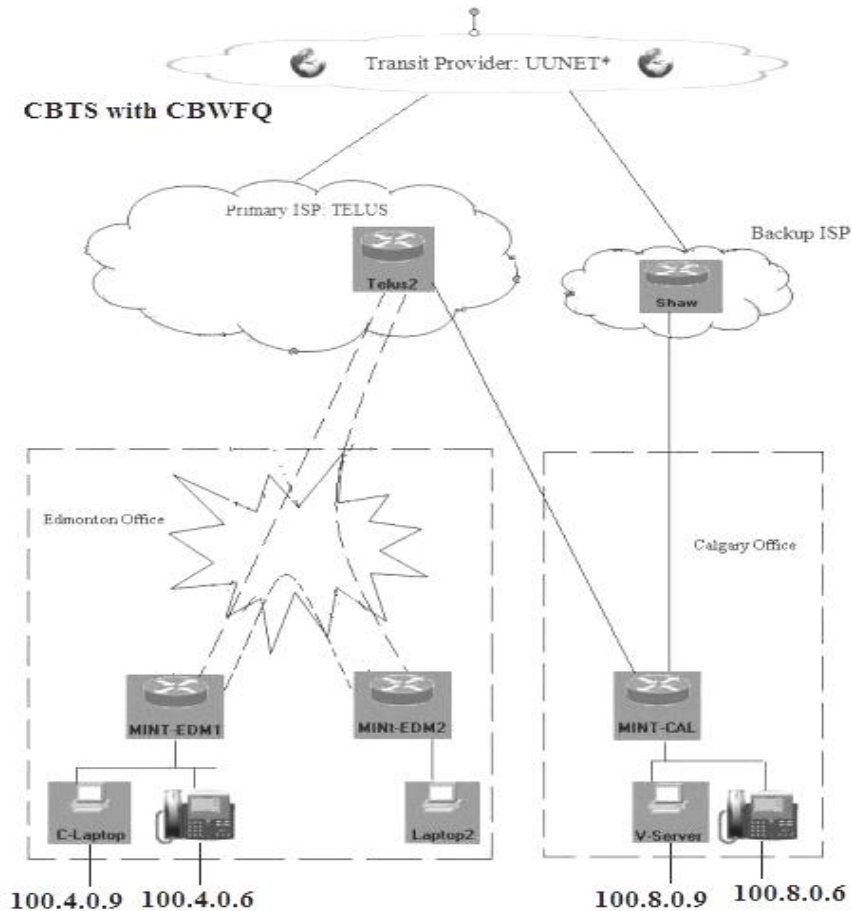


Figure 6.1

6.7 Telus1's Configuration With CBTS – Case 1

In this part of the experiment we had configured two classes, “voice” and “data” under policy named CBWFQ.

Voice class has configured with 50% of the bandwidth, the peak CIR committed to this class was 128000bps. Queue Limit to this class was 10 packets. In this class voice has being configured.

Data has configured with 20% of the bandwidth, the average CIR committed to this class was 12800bps. Queue Limit to this class was 64 packets (default). In this class ,any data, except voice was configured .

Table 6.2 and running configuration confirms this :

```

telus1#sh policy-map

Policy Map CBWFQ
  Class voice
    Bandwidth 50 (%) Max Threshold 10 (packets)
    Traffic Shaping
      Peak Rate Traffic Shaping
        CIR 128000 (bps) Max. Buffers Limit 1000 (Packets)
  Class data
    Bandwidth 20 (%) Max Threshold 64 (packets)
    Traffic Shaping
      Average Rate Traffic Shaping
        CIR 12800 (bps) Max. Buffers Limit 1000 (Packets)

```

Table 6.2

```

telus1#sh runn
Building configuration...

Current configuration : 2574 bytes
!
class-map match-all data
  match input-interface Serial0/1
class-map match-any voice
  match ip rtp 16384 16383
!
!
policy-map CBWFQ
  class voice
    bandwidth percent 50
    queue-limit 10
    shape peak 128000
  class data
    bandwidth percent 40
    queue-limit 20
    shape average 12800
!
!
!
interface Serial0/0
  bandwidth 10000000
  no ip address
  service-policy output CBWFQ
  encapsulation frame-relay
  clock rate 128000
!
!
end

telus1#

```

Policy named "CBWFQ"

Class named "voice"

Class named "data"

6.8 Statistics1

We had implemented CBTS(Class Based Traffic Shaping) and the configuration for that is shown above with CBTS parameters (marked with arrows and pointers)

```
Telus1#sh policy-map interface serial 0/0

Serial0/0

Service-policy output: CBWFQ

Class-map: voice (match-any)
  Target/Average   Byte   Sustain   Excess   Interval   Increment
  Rate            Limit  bits/int  bits/int  (ms)       (bytes)
  256000/128000   1984   7936      7936      62          1984

  Adapt Queue     Packets  Bytes    Packets  Bytes    Shaping
  Active Depth    26743   1711552  0         Delayed  Delayed  Active
  -              0       26743   1711552  0         0        no

Class-map: data (match-all)
  Target/Average   Byte   Sustain   Excess   Interval   Increment
  Rate            Limit  bits/int  bits/int  (ms)       (bytes)
  12800/12800     2000   8000      8000      666         1000

  Adapt Queue     Packets  Bytes    Packets  Bytes    Shaping
  Active Depth    2584    3345577  47        Delayed  Delayed  Active
  -              78      2584    3345577  47        51688   yes

Class-map: class-default (match-any)
  2514 packets, 3153184 bytes
  5 minute offered rate 0 bps, drop rate 0 bps
  Match: any

Telus1#sh policy-map interface serial 0/0

Serial0/0

Service-policy output: CBWFQ

Class-map: voice (match-any)
  Target/Average   Byte   Sustain   Excess   Interval   Increment
  Rate            Limit  bits/int  bits/int  (ms)       (bytes)
  256000/128000   1984   7936      7936      62          1984

  Adapt Queue     Packets  Bytes    Packets  Bytes    Shaping
  Active Depth    27738   1775232  0         Delayed  Delayed  Active
  -              0       27738   1775232  0         0        no

Class-map: data (match-all)
  Target/Average   Byte   Sustain   Excess   Interval   Increment
  Rate            Limit  bits/int  bits/int  (ms)       (bytes)
  12800/12800     2000   8000      8000      666         1000

  Adapt Queue     Packets  Bytes    Packets  Bytes    Shaping
  Active Depth    2611    3376333  74        Delayed  Delayed  Active
  -              91      2611    3376333  74        82444   yes
```

```

Class-map: class-default (match-any)
  2517 packets, 3153273 bytes
  5 minute offered rate 0 bps, drop rate 0 bps
  Match: any

Telus1#sh policy-map interface serial 0/0

Serial0/0

Service-policy output: CBWFQ

Class-map: voice (match-any)

      Target/Average   Byte   Sustain   Excess   Interval   Increment
      Rate             Limit  bits/int  bits/int  (ms)       (bytes)
      256000/128000    1984   7936      7936     62         1984

Adapt  Queue   Packets  Bytes   Packets  Bytes   Shaping
Active Depth                Bytes   Delayed  Delayed  Active
-      0       28275   1809600  0        0       no

Class-map: data (match-all)

      Target/Average   Byte   Sustain   Excess   Interval   Increment
      Rate             Limit  bits/int  bits/int  (ms)       (bytes)
      12800/12800      2000   8000      8000     666        1000

Adapt  Queue   Packets  Bytes   Packets  Bytes   Shaping
Active Depth                Bytes   Delayed  Delayed  Active
-      80       2623   3391453  86       97564   yes

Class-map: class-default (match-any)
  2518 packets, 3153286 bytes
  5 minute offered rate 0 bps, drop rate 0 bps
  Match: any

telus1#

```

Table 6.3

6.9 Statistics1's Summary

As explained before, two class had been configured with different parameters. Class named “voice” was configured in such a way that voice packets should not get delayed/dropped but we have put some restrictions on the class named “data” like bandwidth and peak CIR. That parameters were configured under CBWFQ. Those parameters was not enough for a moderate quality of video to run smoothly through the network. For that reasons, in statistics table , Traffic Shaping becomes active in class name “data” ONLY, but not in class named “voice”. Class named “data” was using the free bandwidth of class named “voice”. Because this was the principle of CBWFQ, that if one class goes out of it's available bandwidth then other class can use it.

6.10 Telus1's Configuration With CBTS – Case 2

In this part of the experiment we had configured two classes, “voice” and “data” under policy named CBWFQ.

Voice class has configured with 50% of the bandwidth, the peak CIR committed to this class was 128000bps. Queue Limit to this class was 10 packets. In this class voice has being configured.

Data has configured with 40% of the bandwidth, the average CIR committed to this class was 128000bps. Queue Limit to this class was 20 packets. In this class ,any data, except voice was configured .

Table 6.4 and running configuration confirms this :

```
Telus1#sh policy-map
Policy Map CBWFQ
  Class voice
    Bandwidth 50 (%) Max Threshold 10 (packets)
    Traffic Shaping
      Peak Rate Traffic Shaping
        CIR 128000 (bps) Max. Buffers Limit 1000 (Packets)
  Class data
    Bandwidth 40 (%) Max Threshold 20 (packets)
    Traffic Shaping
      Average Rate Traffic Shaping
        CIR 128000 (bps) Max. Buffers Limit 1000 (Packets)
```

Table 6.4

6.11 Statistics2

```
Telus1#sh policy-map interface serial 0/0

Serial0/0

Service-policy output: CBWFQ

Class-map: voice (match-any)

      Target/Average   Byte   Sustain   Excess   Interval   Increment
      Rate             Limit  bits/int  bits/int  (ms)       (bytes)
256000/128000        1984   7936     7936     62         1984

Adapt  Queue   Packets  Bytes   Packets  Bytes   Shaping
Active Depth                               Delayed  Delayed  Active
-      0       818     52352   0        0       no

Class-map: data (match-all)

      Target/Average   Byte   Sustain   Excess   Interval   Increment
      Rate             Limit  bits/int  bits/int  (ms)       (bytes)
128000/128000        1984   7936     7936     62         992

Adapt  Queue   Packets  Bytes   Packets  Bytes   Shaping
Active Depth                               Delayed  Delayed  Active
-      0       10      1341   0        0       no

Class-map: class-default (match-any)
```

```

2434 packets, 3149004 bytes
5 minute offered rate 0 bps, drop rate 0 bps
Match: any

Telus1#sh policy-map interface serial 0/0

Serial0/0

Service-policy output: CBWFQ

Class-map: voice (match-any)
  Target/Average   Byte   Sustain   Excess   Interval   Increment
  Rate             Limit  bits/int  bits/int  (ms)       (bytes)
  256000/128000   1984   7936     7936     62         1984

  Adapt Queue    Packets  Bytes    Packets  Bytes    Shaping
  Active Depth    Delayed  Delayed  Active
  -      0         44      6549    0        0        no

Class-map: data (match-all)

  Target/Average   Byte   Sustain   Excess   Interval   Increment
  Rate             Limit  bits/int  bits/int  (ms)       (bytes)
  128000/128000   1984   7936     7936     62         992

  Adapt Queue    Packets  Bytes    Packets  Bytes    Shaping
  Active Depth    Delayed  Delayed  Active
  -      0         8353   534592  0        0        no

Class-map: class-default (match-any)
2457 packets, 3150078 bytes
5 minute offered rate 0 bps, drop rate 0 bps
Match: any

Telus1#

```

Table 6.5

6.12 Statistics2's Summary

As explained before, two class had been configured with almost same parameters. In this case class named “data” had enough bandwidth and CIR value is much higher that in previous case. That parameters was configured under CBWFQ. For that reasons, in statistics table, Traffic Shaping remains non-active after playing the video. It is shown in statistics table, that number of packets keeps on increasing in both of the classes but traffic shaping remains non active.

6.13 Statistics from Ethereal

No. -	Time	Source	Destination	Protocol	Info
1	0.000000	100.8.0.9	100.4.0.9	UDP	Source port: 1977 Destination port: 1234
2	0.703180	100.8.0.9	100.4.0.9	UDP	Source port: 1976 Destination port: 1234
3	2.043491	100.8.0.9	100.4.0.9	UDP	Source port: 1979 Destination port: 1234
4	2.670036	100.8.0.9	100.4.0.9	UDP	Source port: 1978 Destination port: 1234
5	3.372347	100.8.0.9	100.4.0.9	UDP	Source port: 1977 Destination port: 1234
6	4.699065	100.8.0.9	100.4.0.9	UDP	Source port: 1976 Destination port: 1234
7	5.313334	100.8.0.9	100.4.0.9	UDP	Source port: 1980 Destination port: 1234
8	6.014266	100.8.0.9	100.4.0.9	UDP	Source port: 1979 Destination port: 1234
9	7.334863	100.8.0.9	100.4.0.9	UDP	Source port: 1978 Destination port: 1234
10	7.950806	100.8.0.9	100.4.0.9	UDP	Source port: 1977 Destination port: 1234
11	9.367371	100.8.0.9	100.4.0.9	UDP	Source port: 1980 Destination port: 1234
12	9.987074	100.8.0.9	100.4.0.9	UDP	Source port: 1979 Destination port: 1234
13	10.704970	100.8.0.9	100.4.0.9	UDP	Source port: 1978 Destination port: 1234
14	12.034855	100.8.0.9	100.4.0.9	UDP	Source port: 1977 Destination port: 1234
15	12.659971	100.8.0.9	100.4.0.9	UDP	Source port: 1980 Destination port: 1234
16	13.992038	100.8.0.9	100.4.0.9	UDP	Source port: 1979 Destination port: 1234
17	14.612547	100.8.0.9	100.4.0.9	UDP	Source port: 1978 Destination port: 1234
18	15.319141	100.8.0.9	100.4.0.9	UDP	Source port: 1977 Destination port: 1234
19	16.645926	100.8.0.9	100.4.0.9	UDP	Source port: 1980 Destination port: 1234
20	17.346561	100.8.0.9	100.4.0.9	UDP	Source port: 1979 Destination port: 1234
21	17.974033	100.8.0.9	100.4.0.9	UDP	Source port: 1978 Destination port: 1234
22	19.294250	100.8.0.9	100.4.0.9	UDP	Source port: 1977 Destination port: 1234
23	20.003855	100.8.0.9	100.4.0.9	UDP	Source port: 1980 Destination port: 1234
24	20.626505	100.8.0.9	100.4.0.9	UDP	Source port: 1979 Destination port: 1234
25	22.026090	100.8.0.9	100.4.0.9	UDP	Source port: 1978 Destination port: 1234
26	22.644857	100.8.0.9	100.4.0.9	UDP	Source port: 1977 Destination port: 1234
27	23.986836	100.8.0.9	100.4.0.9	UDP	Source port: 1980 Destination port: 1234
28	24.616193	100.8.0.9	100.4.0.9	UDP	Source port: 1979 Destination port: 1234
29	25.952889	100.8.0.9	100.4.0.9	UDP	Source port: 1978 Destination port: 1234
30	26.657235	100.8.0.9	100.4.0.9	UDP	Source port: 1977 Destination port: 1234
31	27.276098	100.8.0.9	100.4.0.9	UDP	Source port: 1980 Destination port: 1234
32	28.650831	100.8.0.9	100.4.0.9	UDP	Source port: 1979 Destination port: 1234
33	29.531864	100.8.0.9	100.4.0.9	IP	Fragmented IP protocol (proto=UDP 0x11, off=0)
34	30.509593	100.8.0.9	100.4.0.9	UDP	Source port: 1980 Destination port: 1234
35	30.686293	100.8.0.9	100.4.0.9	UDP	Source port: 1981 Destination port: 1234
36	31.270001	100.8.0.9	100.4.0.9	UDP	Source port: 1981 Destination port: 1234

Frame 1 (14/0) bytes on wire (14/0) bytes captured)																	
0020	00	09	07	b9	04	d2	05	38	c2	1e	80	21	2c	a9	33	9c	..b....8...!.,.3.
0030	3f	b8	00	00	7c	b0	47	00	0a	1f	fb	ff	4b	68	3b	49	?... .G.kh;I
0040	ea	29	0b	f2	6d	49	d3	ea	40	00	70	5f	f5	fe	5f	04	.)...mI... @.p..._
0050	8a	55	ef	a2	13	ff	f5	de	a9	fb	40	89	42	64	c7	56	.U..... ..@.Bd.V
0060	53	13	9b	e5	0d	a5	29	54	7b	09	d7	48	00	24	c0	e7	S.....)T {...H.S..
0070	28	e7	e7	7f	00	ff	40	f4	f0	e4	e0	f4	4f	b7	6a	d4	s.....T

6.14 Conclusion

Class Based Traffic Shaping was implemented in this scenario. Class Based Weighted Fair Queue was the mechanism that we used in this experiment. We divided that experiment into two separate parts. In the first part two classes had been configured, from which one had configured with fewer bandwidth and CIR value than required. So traffic shaping becomes active in the class with fewer bandwidth, because we had send a stream of video between the end station i.e between Edmonton and Calgary Office. This clarifies the function of CBTS. But we used CBWFQ as queuing mechanism, so according to that if one class has some free bandwidth then other class can use that, if other class gone out of the available bandwidth. This thing can be seen by comparing *Table 6.3* and *Table 6.5*. The number of packets in voice class – 27738 (*Table 6.3*), when data class gone out of bandwidth, it started using free bandwidth of voice class. This is the basic principle of CBWFQ. But in second part of the experiment we implemented two classes, with almost same bandwidth and CIR value. In that case after streaming the video the traffic shaping did not come to active state. Because the bandwidth available in video class was enough for moderate quality of video. In comparison to number of packets in *Table 6.3*, *Table*

6.5 has just 44 packets in voice class. This was due to the CBWFQ. In second case CBTS does not comes to active state so that data class didn't use any of bandwidth of voice class.

Chapter 7

Frame Relay Traffic Shaping

7.1 FRTS

Cisco's Frame Relay Traffic Shaping feature provides many parameters for handling users' traffic and managing congestion on Frame Relay networks. With Frame Relay Traffic Shaping, users are able to configure rate enforcement to either the CIR level or to other user-defined values on a per-VC basis. In this way, bandwidth can be flexibly allocated to each VC and tailored to specific requirements. This allows the feature to provide a mechanism for sharing a common Frame Relay connection by multiple VCs over the same line.

In addition, Frame Relay Traffic Shaping allows traffic queuing to be configured at the VC level. Presently, Frame Relay Traffic Shaping supports priority queuing, custom queuing, and First-In-First-Out (FIFO) queuing (default). In our scenario we implemented priority queue as queue mechanism in FRTS.

Besides administering bandwidth, Frame Relay Traffic Shaping can be used to manage network congestion. The Frame Relay specifications support a congestion notification mechanism with the implementation of BECN and FECN bits inside the Frame Relay headers. Before Frame Relay Traffic Shaping, there was no mechanism to allow the Cisco Frame Relay Access Device (FRAD) to react to BECN-tagged packets received from the network. With Frame Relay Traffic Shaping, a Cisco router can dynamically throttle traffic as soon as it detects congestion, signaled by the presence of BECN tagged packets. When a Cisco router is configured to respond to BECN, it holds the overload packets in the queues to reduce the flow of data into a congested Frame Relay network. The throttling is performed at a per-VC level, and the transmission rate is adjusted dynamically based on the number of BECN-tagged packets received. This allows a Cisco router to actively adapt to downstream congestion conditions in the Frame Relay network.

In *Figure 7.1*, we see that Router 1 is connected to a Frame Relay switch network (shown as the cloud) via a T1 interface with three virtual circuits. Routers 2 and 3 are connected to different parts of the network, each with a 64 Kbps port speed and a CIR of 16 Kbps. Router 4 is connected with a port speed of 256 Kbps and a 64 Kbps CIR. It is possible that the traffic flowing out of Router 1 might overload any one of the three other routers. Enabling FRTS can solve this problem. Enable FRTS at the hub location and set the

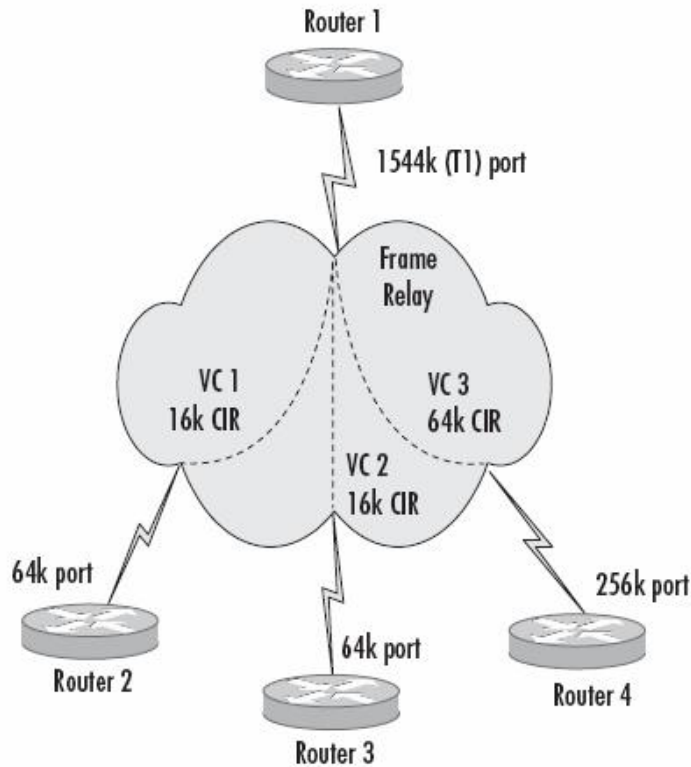


Figure 7.1

CIR parameter equal to the port speed at the far end of the VC. Thus, for the first VC from Router 1 to Router 2, we would have a CIR set to 64 Kbps. The same configuration would apply to the second VC. We would set the CIR of the third VC to 256 Kbps. This overcomes the data-rate mismatch, the traffic becomes well behaved, and unnecessary packet drops are eliminated.

7.2 How FRTS Works?

This discussion tells how Frame Relay Traffic Shaping generally works. *Figure 7.1* illustrates a flow diagram of a packet arriving on an interface of a router that has Frame Relay Traffic Shaping enabled on the packet's outgoing interface.

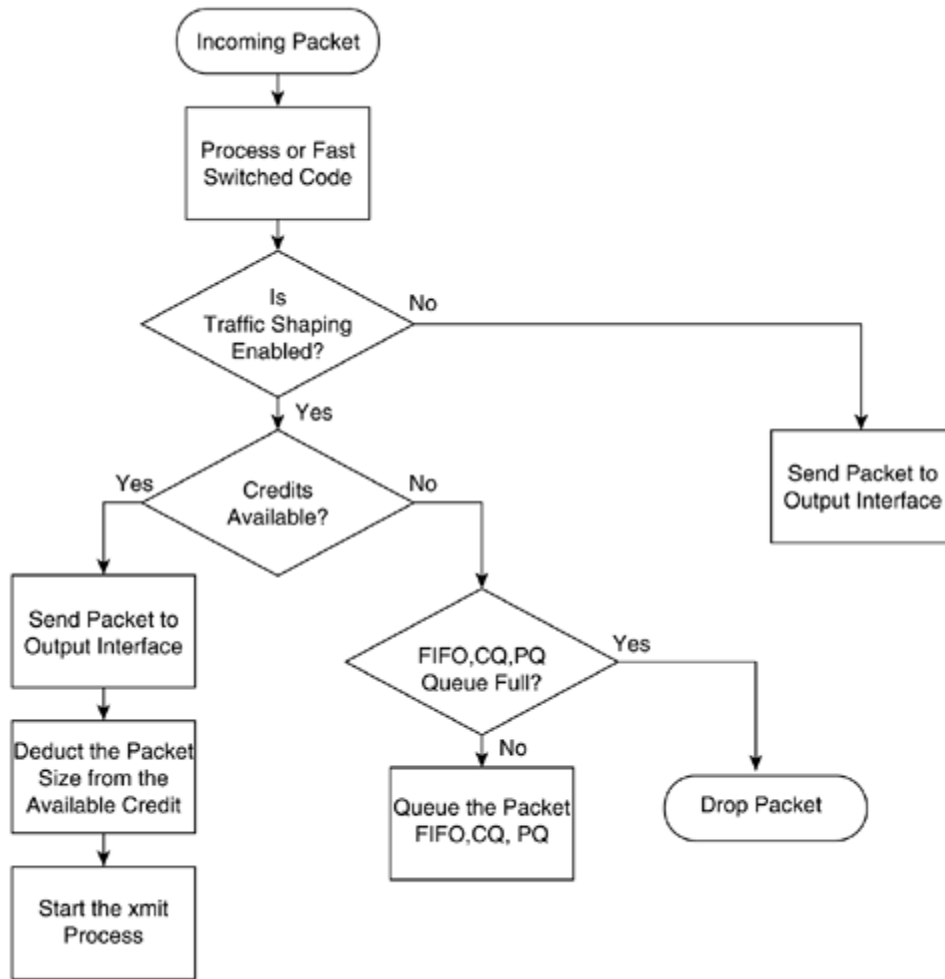


Figure 7.2. Flow Diagram of Packet with Frame Relay Traffic Shaping

7.3 Difference between VoFR and VoIP

Voice over IP (VoIP) and voice over frame relay (VoFR) are two of the popular solutions being considered for providing voice/data solution. As from the name implies VoIP uses Internet Protocol and VoFR uses Frame Relay Protocol . The Internet protocol (IP) is the network layer standard for communication over the Internet. Frame relay is a popular layer-2 protocol designed primarily for data transmission across enterprise WANs.

Internet Protocol

The Internet protocol (IP) provides a connectionless, best-effort service to applications. Several enhancements have been made to the protocol to improve the QoS it provides. These include the next generation IP (IPv6) and the real-time transport protocol (RTP).

Frame Relay

Frame relay (FR) is a connection-oriented, packet-switching network protocol, designed to provide reliable, high-speed network service. FR uses virtual circuits instead of the datagram service used by VoIP systems.

7.3.1 Which One is Better?

The technical feasibility of new solution depends on the extent to which changes have to be made to the current network infrastructure. For example, will it be necessary to replace existing equipment and wires? In a typical VoFR application, a customer's PBX is connected to a voice capable frame relay access device (VFRAD), which is then connected to the data network. In such configuration, the only additional equipment necessary is the VFRAD. The current infrastructure remains more or less the same. However, depending on the size of the organization, additional permanent virtual circuits (PVCs) may be needed. In addition, SLAs will have to be modified to accommodate voice traffic. This could prove quite costly.

On the other hand, a VoIP solution, using IPv4 would not require any major changes to the infrastructure. However, IP-based QoS mechanisms are required to guarantee performance and reliability.

One area in which IP has a clear advantage over FR is that of cost. Many organizations will, therefore, find it more cost-effective to graduate to a VoIP solution, than to invest in new FR technology. In addition, FR normally uses permanent virtual circuits, which are expensive and can be wasteful of bandwidth. This makes FR less attractive for occasionally used circuits. FR had an advantage over IP with respect to delays and bandwidth utilization. However, mechanisms has been made to improve QoS in IP networks (IPv6)

7.4 Reliability and QoS Guarantees

Voice is a mission-critical application. It requires a high degree of reliability to function properly. For VoP applications, the network must be able to guarantee certain minimum levels of QoS. In particular, there must be a bound on the end-to-end delay as well as the delay variation or jitter. End-to- end delay refers to the time that elapses between a signal being sent by the speaker, and it being received by the listener.

Parameter	Optimal network settings	Minimum network settings
<i>Latency</i>	<= 100ms	<= 150ms
<i>Jitter</i>	<= 40ms	<= 75ms
<i>Packet Loss</i>	<= 1%	<= 75ms

Table 7.1

7.5 Differences Between FRTS and GTS

This section explains the differences between generic traffic shaping and Frame Relay Traffic Shaping. Generic traffic shaping shapes traffic by reducing outbound traffic flow to avoid congestion. Generic traffic shaping is applied on a per-interface basis, and access lists can be used to selectively filter the type of traffic to shape. Generic traffic shaping works with Frame Relay, ATM, Switched Multimegabit Data Service (SMDS), and Ethernet. Both generic traffic shaping and Frame Relay Traffic Shaping are similar in implementation, but there are some notable differences. Generic traffic shaping mainly differs from Frame Relay Traffic Shaping with regard to CLI configuration and the queue types used. *Table 7.2* sums up the differences between generic traffic shaping and Frame Relay Traffic Shaping.

Mechanism	Generic Traffic Shaping	Frame Relay Traffic Shaping
CLI	Applies parameters on a per-interface basis traffic group command used	Classes of parameters configured in map class Applies parameters to all VCs on an interface through inheritance No traffic group command supported
Queues supported	Weighted Fair Queuing (WFQ) per subinterface	Weighted Fair Queuing (WFQ), Priority Queuing, Custom Queuing, First-In-First-Out (FIFO) per VC supported

Table 7.2

Both traffic shaping methods are similar in implementation, though their command-line interfaces differ somewhat and they use different types of queues to contain and shape traffic that is deferred. In particular, the underlying code that determines whether there is enough credit in the token bucket for a packet to be sent or whether that packet must be delayed is common to both features. If a packet is deferred, GTS uses a weighted fair queue to hold the delayed traffic. FRTS uses either a custom queue or a priority queue for the same, depending on what we have configured.

7.6 Priority Queuing

Priority queuing (PQ) enables network administrators to prioritize traffic based on specific criteria. These criteria include protocol or sub-protocol types, source interface, packet size, fragments, or any parameter identifiable through a standard or extended access list. PQ offers four different queues:

- à low priority
- à normal priority
- à medium priority

à high priority

Each packet is assigned to one of these queues. If no classification is assigned to a packet, it is placed in the normal priority queue.

7.6.1 How PQ Works ?

The priority of each queue is absolute. As packets are processed, PQ examines the state of each queue, always servicing higher priority queues before lower priority queues. This means that as long as there is traffic in a higher priority queue, lower priority queues will not be processed. PQ, therefore, does not use a fair allocation of resources among its queues. It services them strictly on the basis of the priority classifications configured by the network administrator. *Figure 7.3* shows PQ in action.

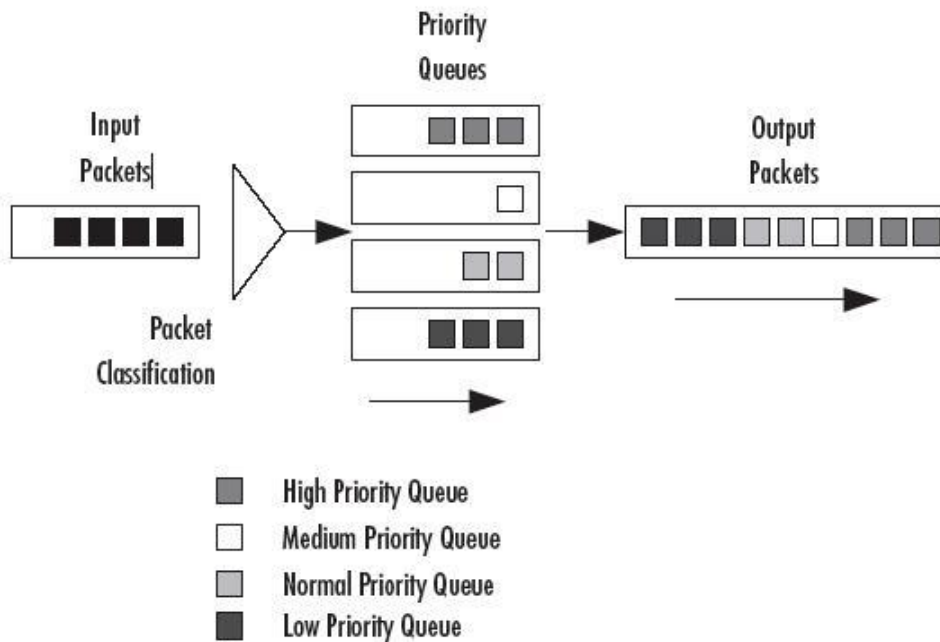


Figure 7.3 – Priority Queue in Operation

7.6.2 Queue Sizes

The default queue sizes for PQ are shown in *Table 7.3*. These queue sizes can be manually adjusted from 0 to 32,767 packets.

Limit	Size
High priority queue limit	20 packets
Medium priority queue limit	40 packets

Normal priority queue limit	60 packets
Low priority queue limit	80 packets

Table 7.3

7.6.3 Configuring PQ

In this scenario PQ was used as Queue Mechanism.

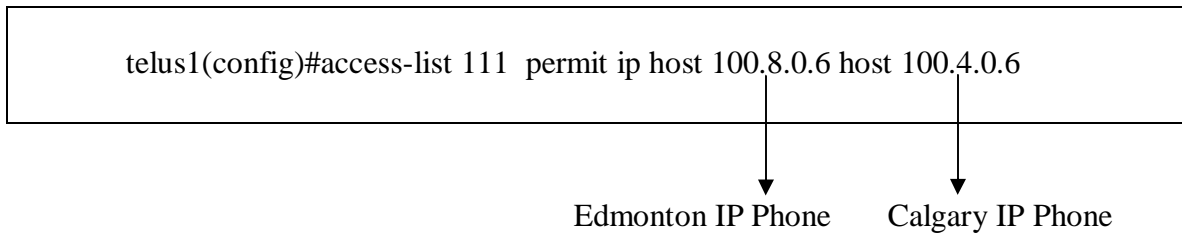
The tasks involved in configuring priority queuing are as follows:

- à Define a priority list
- à Assign the priority list to an interface

In this experiment we use “priority list 1”, (Table 7.4) which was defined as :

- Low Priority – Default
- Normal Priority – Normal
- High Priority – Access List 111

Access List 111 was defined as :



```
telus1#sh queueing priority
Current DLCI priority queue configuration:
Current priority queue configuration:

List   Queue  Args
1      low    default
1      normal protocol ip
1      high   protocol ip          list 111
telus1#
```

Table 7.4

In **Table 7.5** the output of “**show interface interface-type interface-number**” is shown:

```
telus1#sh interfaces serial 0/0
Serial0/0 is up, line protocol is up
  Hardware is CD2430 in sync mode
  MTU 1500 bytes, BW 10000000 Kbit, DLY 20000 usec,
    reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation FRAME-RELAY, loopback not set
  Keepalive set (10 sec)
  LMI enq sent 270, LMI stat recvd 270, LMI upd recvd 0, DTE LMI up
  LMI enq recvd 0, LMI stat sent 0, LMI upd sent 0
  LMI DLCI 1023 LMI type is CISCO frame relay DTE
```



```

FR SVC disabled, LAPF state down
Broadcast queue 0/64, broadcasts sent/dropped 58/0, interface
broadcasts 9
Last input 00:00:01, output 00:00:00, output hang never
Last clearing of "show interface" counters 00:45:02
Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
Queueing strategy: priority-list 1
Output queue (queue priority: size/max/drops):
  high: 0/20/0, medium: 0/40/0, normal: 0/60/0, low: 0/80/0
5 minute input rate 0 bits/sec, 0 packets/sec
5 minute output rate 123000 bits/sec, 11 packets/sec
29284 packets input, 2249427 bytes, 0 no buffer
  Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
  32 input errors, 32 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
19187 packets output, 16255672 bytes, 0 underruns
  0 output errors, 0 collisions, 2 interface resets
  0 output buffer failures, 0 output buffers swapped out
  0 carrier transitions
DCD=up

```

Table 7.5

7.7 BECN

Its stands for Backward Explicit Congestion Notification . In Frame Relay frames, BECN bits are set to indicate that congestion is occurring in the direction opposite to the direction in which the traffic is The devices receiving the BECN bits should slow (or stop) their traffic until the congestion is relieved to avoid buffer overflows and lost data. The good part about BECN bits is that they are sent to the appropriate party - the one that's supposed to slow down.

In addition to adaptive shaping to BECN, Frame Relay Traffic Shaping supports adaptive shaping to Foresight, an additional functionality that allows a Cisco router to actively communicate with a Cisco Frame Relay switch using timed Foresight messages. When adaptive shaping to Foresight is enabled, a Cisco router can dynamically adjust its transmission rate based on congestion conditions reported in Foresight messages received from the switch without relying on remote devices to send BECN tagged packets. Figure illustrates the small difference between adaptive shaping to BECN and Foresight. *Figure 7.4. Using BECN and ForeSight as Congestion Notification Mechanisms*

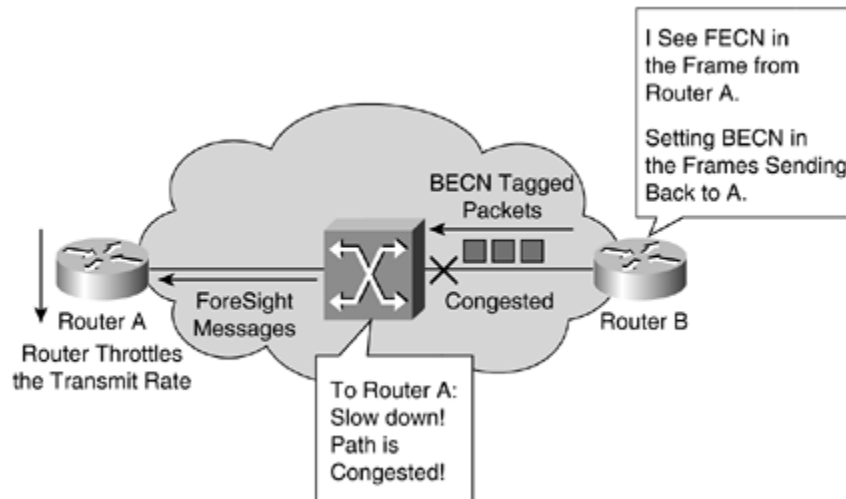


Figure 7.4

In summary, the Frame Relay Traffic Shaping feature provides functionalities in three major components:

- Rate Enforcement configured on a per-VC basis
- Priority queuing, custom queuing, or default FIFO queuing at the VC level
- BECN and Foresight support for congestion management.

7.8 Configuring FRTS

Configuration Steps : Before applying these commands, policy and class have to be configured.

Command	Purpose
1 Router(config)# map-class frame relay class-name	Configure static map class
2. Router(config-map-class)# frame-relay class-name bandwidth percent number	Configures Voice Options
3. Router(config-map-class)# frame-relay adaptive-shaping becn	Configures rate adjustment in response to BECN
4. Router(config-map-class)# frame-relay cir cir	Configures Committed Information Rate (CIR)
5. Router(config-map-class)# frame-relay bc bc(bps)	Configures Committed burst size (Bc)
6. Router(config-map-class)# service-policy output policy-name	Configures QoS Service Policy

7.9 Network Topology for FRTS

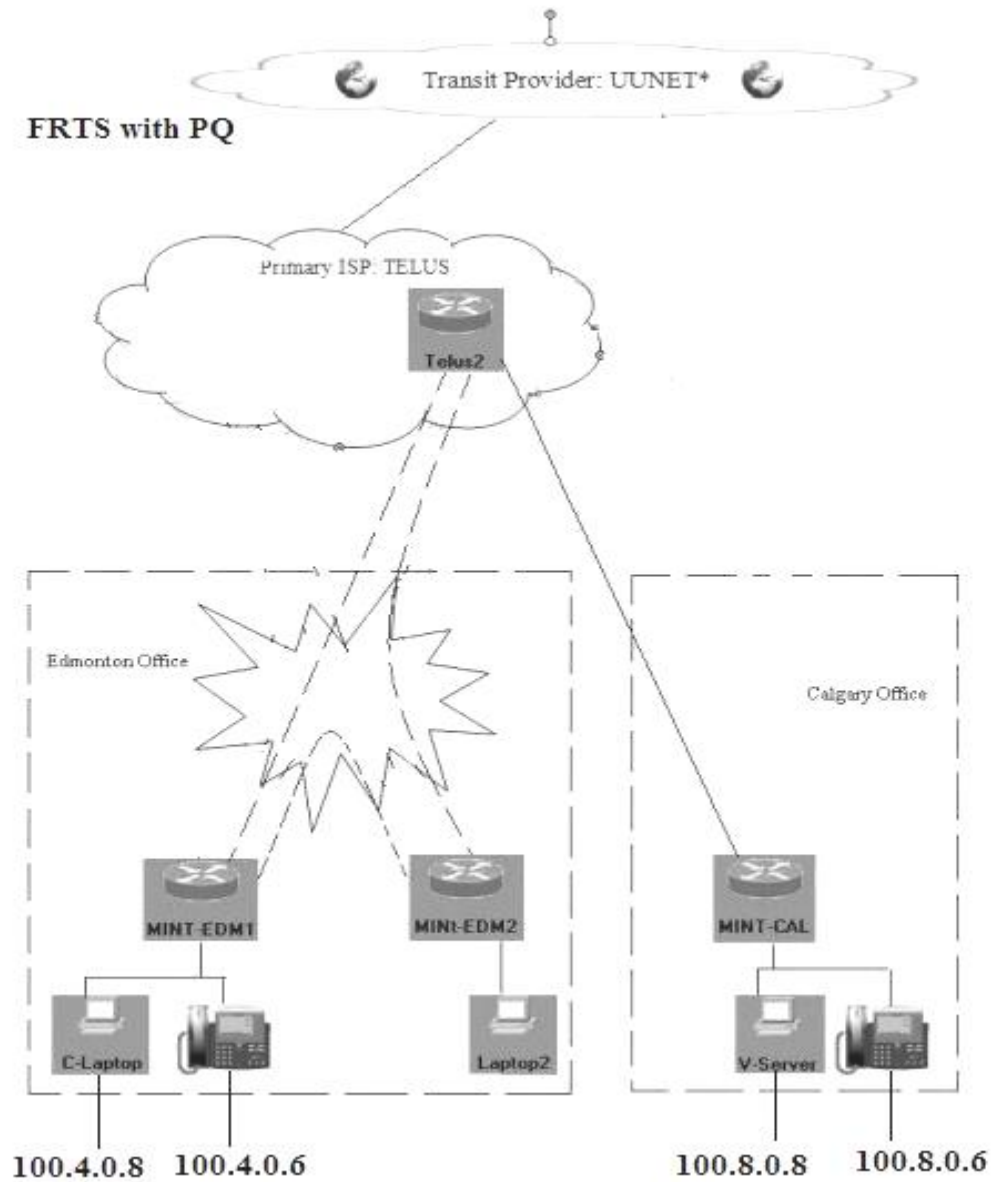


Figure 7.5

In this scenario Frame Relay Traffic Shaping was configured in the network .Below are the running configuration of Telus1 and EDM-MINT1 and EDM-MINT2. FRTS was implemented on serial 0/0.2 interface of Telus1. The Value of CIR ,Bc ,Be ,MinCir is shown in the configuration. A policy named “policy1” and a classes named “class1” and “class2” were configured. “Class1” was configured inside “policy1” and “class1” was implemented on sub-interface serial 0/0.2 on Telus1 side and that same “class1” was

defined on MINT-EDM1 side at interface serial 0/1.2 .“Class2” was configured inside “policy1” and “class2” was implemented on sub-interface serial 0/0.1 on Telus1 side and that same “class2” was defined on MINT-EDM2 side at interface serial 0/1.1 The 45% of the bandwidth is reserved for “class1” is given to class1 and that class is used for VoIP in frame relay , because port range 16384 32767 is used for Voice in Cisco devices. BECN is used in this case. On MINT-EDM2 side 55% bandwidth” is allocated to interface s0/1.1.MINT-EDM2 is used as video client. PQ was used in the experiment .

7.10 Telus1’s Running Configuration with FRTS

```

telus1#
telus1#sh runn
Building configuration...

Current configuration : 2624 bytes
!
!
class-map match-all class2
  match access-group 102
class-map match-all class1
  match access-group 101
!
!
policy-map policy1
  class class1
    bandwidth percent 45
  class class2
    bandwidth percent 55
!
!
!
!
interface Serial0/0.1 point-to-point
  bandwidth 10000000
  ip address 100.1.20.1 255.255.255.252
  frame-relay class class2
  frame-relay interface-dlci 22
!
interface Serial0/0.2 point-to-point
  ip address 100.1.10.1 255.255.255.252
  frame-relay class class1
  frame-relay interface-dlci 20
!
map-class frame-relay class2
  frame-relay cir 64000
  frame-relay bc 8000
  frame-relay be 1000
  frame-relay mincir 64000
  service-policy output policy1
!
map-class frame-relay class1
  frame-relay cir 64000
  frame-relay bc 8000

```

Class Named "class2"

Class Named "class1"

Policy Named "policy1"

Bandwidth associated to class1

Bandwidth allocated to class2

Class2 implemented on s0/0.1

Class1 implemented on s0/0.2

```

frame-relay mincir 32000
frame-relay adaptive-shaping becn Voice Port Range In Cisco Devices
service-policy output policyl
!
access-list 101 permit udp any any range 16384 32767
access-list 102 permit tcp any any
route-map med4primary permit 10
  set metric 100
!
end

telus1#

```

7.11 MINT-EDM1'S Running Configuration with FRTS

```

MINT-EDM1#
MINT-EDM1#sh run
Building configuration...

!
interface Serial0/1.1 point-to-point
 ip address 100.2.10.2 255.255.255.252
 frame-relay interface-dlci 17
!
interface Serial0/1.2 point-to-point
 ip address 100.1.10.2 255.255.255.252 class1 defined on client side
 frame-relay class class1
 frame-relay interface-dlci 21
!
!
!
map-class frame-relay class1
 frame-relay cir 64000
 frame-relay bc 8000
 frame-relay mincir 32000
 frame-relay adaptive-shaping becn
!
route-map lpref4primary permit 10
  set local-preference 200
!
end

MINT-EDM1#

```

7.12 MINT-EDM2'S Running Configuration with FRTS

```

MINT-EDM2#sh run
Building configuration...

```

```

Current configuration : 1769 bytes
!
!
interface Serial0/1.1 point-to-point
 bandwidth 10000000
 ip address 100.1.20.2 255.255.255.252
 frame-relay class class2
 frame-relay interface-dlci 23
!
!
!
map-class frame-relay class2
 frame-relay cir 64000
 frame-relay bc 8000
 frame-relay be 1000
 frame-relay mincir 64000
 frame-relay adaptive-shaping becn
!
end

MINT-EDM2#

```

class2 defined on client side

7.13 Statistics

7.13.1 Statistics 1

```

telus1#sh policy-map interface serial 0/0.2
Serial0/0.2: DLCI 20 -
Service-policy output: policy1
Class-map: class1 (match-all)
 27089 packets, 1744260 bytes
 5 minute offered rate 0 bps, drop rate 0 bps
Match: access-group 101
Queueing
  Output Queue: Conversation 25
  Bandwidth 75 (%) Max Threshold 64 (packets)
  (pkts matched/bytes matched) 0/0
  (depth/total drops/no-buffer drops) 0/0/0

Class-map: class-default (match-any)
 18551 packets, 15645233 bytes
 5 minute offered rate 117000 bps, drop rate 53000 bps
Match: any

```

```

telus1#sh policy-map interface serial 0/0.2
Serial0/0.2: DLCI 20 -
Service-policy output: policy1
Class-map: class1 (match-all)
 28526 packets, 1836801 bytes
 5 minute offered rate 6000 bps, drop rate 0 bps
Match: access-group 101
Queueing
  Output Queue: Conversation 25

```

```

Bandwidth 75 (%) Max Threshold 64 (packets)
(pkts matched/bytes matched) 0/0
(depth/total drops/no-buffer drops) 0/0/0

Class-map: class-default (match-any)
  20201 packets, 17871872 bytes
  5 minute offered rate 111000 bps, drop rate 53000 bps
Match: any

```

```

telus1#sh policy-map interface serial 0/0.2
Serial0/0.2: DLCI 20 -
Service-policy output: policy1
Class-map: class1 (match-all)
  28778 packets, 1853044 bytes
  5 minute offered rate 8000 bps, drop rate 0 bps
Match: access-group 101
Queueing
  Output Queue: Conversation 25
  Bandwidth 75 (%) Max Threshold 64 (packets)
  (pkts matched/bytes matched) 0/0
  (depth/total drops/no-buffer drops) 0/0/0

Class-map: class-default (match-any)
  22294 packets, 17998352 bytes
  5 minute offered rate 109000 bps, drop rate 54000 bps
Match: any

```

```

telus1#sh policy-map interface serial 0/0.2
Serial0/0.2: DLCI 20 -
Service-policy output: policy1
Class-map: class1 (match-all)
  28786 packets, 1853749 bytes
  5 minute offered rate 8000 bps, drop rate 0 bps
Match: access-group 101
Queueing
  Output Queue: Conversation 25
  Bandwidth 75 (%) Max Threshold 64 (packets)
  (pkts matched/bytes matched) 0/0
  (depth/total drops/no-buffer drops) 0/0/0

Class-map: class-default (match-any)
  28432 packets, 18184951 bytes
  5 minute offered rate 109000 bps, drop rate 55000 bps
Match: any

```

Table 7.6

7.13.2 Statistics 2

```

telus1#show frame-relay pvc interface serial 0/0.2 20
PVC Statistics for interface Serial0/0.2 (Frame Relay DTE)

```

DLCI = 20, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE = Serial0/0.2

```
input pkts 65796          output pkts 56190          in bytes 68277558
out bytes 18249713        dropped pkts 0             in pkts dropped 0
out pkts dropped 9533     out bytes dropped 12922082
late-dropped out pkts 9533   late-dropped out bytes 12922082
in FECN pkts 0           in BECN pkts 0           out FECN pkts 0
out BECN pkts 0          in DE pkts 0             out DE pkts 0
out bcast pkts 208       out bcast bytes 50076
Shaping adapts to BECN
pvc create time 02:48:13, last time pvc status changed 02:28:23
cir 64000      bc 8000      be 0          byte limit 1000  interval 125
mincir 32000   byte increment 1000 Adaptive Shaping BECN
pkts 55724     bytes 18132824 pkts delayed 21966   bytes delayed
15843113
shaping active
traffic shaping drops 9533
service policy policy1
Serial0/0.2: DLCI 20 -
```

Service-policy output: policy1

```
Class-map: class1 (match-all)
  37008 packets, 2383233 bytes
  5 minute offered rate 0 bps, drop rate 0 bps
  Match: access-group 101
  Queueing
    Output Queue: Conversation 25
    Bandwidth 75 (%) Max Threshold 64 (packets)
    (pkts matched/bytes matched) 0/0
    (depth/total drops/no-buffer drops) 0/0/0

Class-map: class-default (match-any)
  28313 packets, 28773489 bytes
  5 minute offered rate 117000 bps, drop rate 53000 bps
  Match: any
  Output queue size 64/max total 600/drops 9581
```

telus1#show frame-relay pvc interface serial 0/0.2 20

PVC Statistics for interface Serial0/0.2 (Frame Relay DTE)

DLCI = 20, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE = Serial0/0.2

```
input pkts 65799          output pkts 56327          in bytes 68277709
out bytes 18433420        dropped pkts 0             in pkts dropped 0
out pkts dropped 9646     out bytes dropped 13075762
late-dropped out pkts 9646   late-dropped out bytes 13075762
in FECN pkts 0           in BECN pkts 0           out FECN pkts 0
out BECN pkts 0          in DE pkts 0             out DE pkts 0
out bcast pkts 208       out bcast bytes 50076
Shaping adapts to BECN
pvc create time 02:48:36, last time pvc status changed 02:28:46
```



```

cir 64000      bc 8000      be 0          byte limit 1000  interval 125
mincir 32000   byte increment 1000 Adaptive Shaping BECN
pkts 55862     bytes 18317891  pkts delayed 22104   bytes delayed
16028180
shaping active
traffic shaping drops 9646
service policy policy1
Serial0/0.2: DLCI 20 -

Service-policy output: policy1

Class-map: class1 (match-all)
37008 packets, 2383233 bytes
5 minute offered rate 0 bps, drop rate 0 bps
Match: access-group 101
Queueing
Output Queue: Conversation 25
Bandwidth 75 (%) Max Threshold 64 (packets)
(pkts matched/bytes matched) 0/0
(depth/total drops/no-buffer drops) 0/0/0

Class-map: class-default (match-any)
28564 packets, 29112236 bytes
5 minute offered rate 117000 bps, drop rate 53000 bps
Match: any
Output queue size 64/max total 600/drops 9658
telus1#

```

Table 7.7

7.14 Statistics's Summary

Statistics 1		Class1	Class-Default
	Packets	27089 packets	18557 packets
	Drop Rate	0 bps	53000 bps
Statistics 1			
	Packets	28526 packets	20201 packets
	Drop Rate	0 bps	53000 bps
Statistics 1			
	Packets	28778 packets	22294 packets
	Drop Rate	0 bps	54000 bps
Statistics 1			
	Packets	28786 packets	28432 packets
	Drop Rate	0 bps	55000 bps

Table 7.8

Packets in both of the classes keeps on increasing. “class1” was having VoIP and “class-default” was having video and other data. There was no dropping in class1, because class1 was allocated with bandwidth of “45 percent “ of that interface. In this project the codec used by the Cisco IP Phones is G.729, that uses 8bps (not including header).The thing that shows FRTS in this scenario was that video can’t use “class1” even though it was using the same sub-interface . The reason for that was FRTS, there was an access-list that permits only UDP (port range 16384 - 32767) .There was a packet loss in “class-default”. Video was used as source in that class. Packets keep on increasing at great rate. Drop rate boots from 53000 bps to 55000 bps.

7.15 – Statistics from Ethereal

In this experiment, FRTS had been implemented in the network. On serial interface of “telus1” “policy1” is implemented. “Class1” is implemented on sub-interface s0/0.2 , DLCI - 20 and “Class2” is implemented on sub-interface s0/0.1 ,DLCI – 22.

7.15.1 – *Statistics1 from Ethereal* – This capture was taken when call is just made from MINT-EDM1 to MINT-CAL office. “Class1” is implemented on s0/0.2 with bandwidth of 75% of that interface, with “mincir” = 32000bps.This capture shows the initial process of call.

No. -	Time	Source	Destination	Protocol	Info
1	0.000000	100.4.0.6	100.4.0.1	SKINNY	OffHookMessage
2	0.021438	100.4.0.1	100.4.0.6	SKINNY	SetRingerMessage
3	0.022281	100.4.0.1	100.4.0.6	SKINNY	CallStateMessage
4	0.022916	100.4.0.1	100.4.0.6	SKINNY	DisplayPromptStatusMessage
5	0.023461	100.4.0.1	100.4.0.6	SKINNY	SelectSoftkeysMessage
6	0.029819	100.4.0.6	100.4.0.1	TCP	50384 > 2000 [ACK] Seq=12 Ack=40 win=1400 Len=0
7	0.033204	100.4.0.1	100.4.0.6	SKINNY	ConnectionStatisticsReq
8	0.033827	100.4.0.1	100.4.0.6	SKINNY	OpenReceiveChannel
9	0.049680	100.4.0.6	100.4.0.1	TCP	50384 > 2000 [ACK] Seq=12 Ack=96 win=1400 Len=0
10	0.059737	100.4.0.6	100.4.0.1	TCP	50384 > 2000 [ACK] Seq=12 Ack=124 win=1400 Len=0
11	0.062028	100.4.0.6	100.4.0.1	SKINNY	ConnectionStatisticsRes
12	0.063293	100.4.0.6	100.4.0.1	SKINNY	OpenReceiveChannelAck
13	0.259749	100.4.0.1	100.4.0.6	TCP	2000 > 50384 [ACK] Seq=204 Ack=112 win=3484 Len=0
14	0.278858	100.4.0.1	100.4.0.6	SKINNY	StartMediaTransmission
15	0.279785	100.4.0.6	100.4.0.1	TCP	50384 > 2000 [ACK] Seq=112 Ack=256 win=1400 Len=0
16	0.281250	100.4.0.1	100.4.0.6	RTP	Payload type=ITU-T G.729, SSRC=1844842639, Seq=6, Time=26584461
17	0.301278	100.4.0.1	100.4.0.6	RTP	Payload type=ITU-T G.729, SSRC=1844842639, Seq=7, Time=26584621


```

# Frame 11 (126 bytes on wire (126 bytes captured)
# Ethernet II, Src: Cisco_02:00:0c:1a:00:00, Dst: Cisco_02:00:0c:1a:00:00, Enc: Cisco_90:12:10 (00:00:21:90:12:10)
# Internet Protocol, Src: 100.4.0.6 (100.4.0.6), Dst: 100.4.0.1 (100.4.0.1)
# Transmission Control Protocol, Src Port: 50384 (50384), Dst Port: 2000 (2000), Seq: 12, Ack: 168, Len: 72
# Skinny client control Protocol
  Data Length: 64
  Reserved: 0x00000000
  Message ID: ConnectionStatisticsRes (0x00000023)
  Directory Number: 1011
  Call Identifier: 6
  StatsProcessingType: clearStats (0)
  Packets Sent: 0
  Octets Sent: 0
  Packets Received: 0
  Octets Received: 0
  Packets Lost: 0
  Jitter: 0
  Latency(ms): 0
0000  00 08 21 96 f2 c0 00 07 85 05 63 1d 08 00 45 68  ..!.....c...Eh
0010  00 70 37 a2 00 00 40 06 7a 6f 64 04 00 06 64 04  .p7...#.zod...d.
0020  00 01 c4 d0 07 d0 a8 5c 49 45 ac 80 40 f8 50 18  ..... \IE..#.P.
0030  05 78 e2 89 00 00 40 00 00 00 00 00 00 00 23 00  .x...#. ....#.
0040  00 00 31 30 31 31 00 00 00 00 00 00 00 00 00 00  ..1011.. ....
0050  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ..0000.. ....

```

7.15.2 - *Statistics 2 from Ethereal* – This capture was taken when voice and video is running at same time. “Class1” is for voice and at this point of time video was also on the link . Bandwidth for video is not enough, as video or any other data has to take the “class-default”. “Class-default” has 25% of the total bandwidth. When video is played from server (100.8.0.8) then queue starts building up. When queue becomes full then there will be a packets loss. That was shown in *Table 7.8* .

No. -	Time	Source	Destination	Protocol	Info
987	11.814220	100.4.0.1	100.4.0.6	SKINNY	CloseReceiveChannel
988	11.814752	100.4.0.1	100.4.0.6	SKINNY	StopMediaTransmission
989	11.815665	100.4.0.1	100.4.0.6	SKINNY	ConnectionStatisticsReq
990	11.816214	100.4.0.1	100.4.0.6	SKINNY	SetSpeakerModeMessage
991	11.816643	100.4.0.6	100.4.0.1	SKINNY	ConnectionStatisticsRes
992	11.817164	100.4.0.1	100.4.0.6	SKINNY	CallStateMessage
993	11.817794	100.4.0.1	100.4.0.6	SKINNY	ClearPromptStatusMessage
994	11.818424	100.4.0.1	100.4.0.6	SKINNY	SelectSoftkeysMessage
995	11.818989	100.4.0.1	100.4.0.6	SKINNY	SetSpeakerModeMessage
996	11.819605	100.4.0.6	100.4.0.1	TCP	50374 > 2000 [ACK] Seq=216 Ack=212 win=1400 Len=0
997	11.839627	100.4.0.6	100.4.0.1	TCP	50374 > 2000 [ACK] Seq=216 Ack=276 win=1400 Len=0
998	12.022369	100.4.0.1	100.4.0.6	TCP	2000 > 50374 [ACK] Seq=276 Ack=216 win=2868 Len=0
999	12.420560	100.8.0.8	100.4.0.8	UDP	Source port: 4239 Destination port: 1234
1000	12.556712	100.8.0.8	100.4.0.8	UDP	Source port: 4239 Destination port: 1234
1001	12.733739	100.8.0.8	100.4.0.8	UDP	Source port: 4239 Destination port: 1234
1002	12.926279	100.8.0.8	100.4.0.8	UDP	Source port: 4239 Destination port: 1234
1003	13.118457	100.8.0.8	100.4.0.8	UDP	Source port: 4239 Destination port: 1234
1004	13.214995	100.8.0.8	100.4.0.8	UDP	Source port: 4239 Destination port: 1234
1005	13.486263	100.8.0.8	100.4.0.8	UDP	Source port: 4239 Destination port: 1234
1006	13.576520	100.8.0.8	100.4.0.8	UDP	Source port: 4239 Destination port: 1234
1007	13.879166	100.8.0.8	100.4.0.8	UDP	Source port: 4239 Destination port: 1234
1008	13.977295	100.8.0.8	100.4.0.8	UDP	Source port: 4239 Destination port: 1234
1009	14.074685	100.8.0.8	100.4.0.8	UDP	Source port: 4239 Destination port: 1234
1010	14.346858	100.8.0.8	100.4.0.8	UDP	Source port: 4239 Destination port: 1234
1011	14.443454	100.8.0.8	100.4.0.8	UDP	Source port: 4239 Destination port: 1234
1012	14.628212	100.8.0.8	100.4.0.8	UDP	Source port: 4239 Destination port: 1234
1013	14.820335	100.8.0.8	100.4.0.8	UDP	Source port: 4239 Destination port: 1234
1014	14.906448	cisco_96:f2:c0	cisco_96:f2:c0	LOOP	Reply
1015	14.913371	100.8.0.8	100.4.0.8	UDP	Source port: 4239 Destination port: 1234
1016	15.044509	100.8.0.8	100.4.0.8	UDP	Source port: 4239 Destination port: 1234
1017	15.303776	100.8.0.8	100.4.0.8	UDP	Source port: 4239 Destination port: 1234
1018	15.425920	100.8.0.8	100.4.0.8	UDP	Source port: 4239 Destination port: 1234
1019	15.689006	100.8.0.8	100.4.0.8	UDP	Source port: 4239 Destination port: 1234
1020	15.813132	100.8.0.8	100.4.0.8	UDP	Source port: 4239 Destination port: 1234
1021	15.939080	100.8.0.8	100.4.0.8	UDP	Source port: 4239 Destination port: 1234

Frame 1 (74 bytes on wire, 74 bytes captured)	
Ethernet II, Src: cisco_05:63:1d (00:07:85:05:63:1d), Dst: cisco_96:f2:c0 (00:08:21:96:f2:c0)	
0000	00 08 21 96 f2 c0 00 07 85 05 63 1d 08 00 45 b8 ..!.....E.
0010	00 3c 54 61 00 00 40 11 5d 89 64 04 00 06 64 04 .<T...&].d...d.
0020	00 01 42 4e 07 40 00 28 00 00 80 12 3d f7 01 7f ..BN...(-----f
0030	2e d0 1d 63 05 85 78 2b 00 a0 00 fa c2 00 07 d6 ...c..X+
0040	78 16 c0 a0 00 fa c2 00 07 d6 x.....

17.15.3 - *Statistics 3 from Ethereal* – This capture was taken when call is going to end. These are the final steps when the call is going to terminate. From *Table 7.8* it is pretty clear that there was a huge loss of packets on class-default, but there was no loss in “class1”. Even though the destination comes to same interface of the router . Video can’t use “class1” for video transmission because of policy implemented in network. *Table 8.1* shows that there was no loss on “class 1” that was used for voice and we can see that this from this capture too. In this capture packets lost = 0, for voice, because SCCP is used for voice only.


```

MINT-EDM1#debug ephone statistics mac-address 0007.8505.631D
EPHONE statistics debugging is enabled for phone 0007.8505.631D
MINT-EDM1#
*Mar  1 00:37:28.079: ephone-1[1]:Call Stats for line 3 DN 1 1001 ref 3
*Mar  1 00:37:28.079: ephone-1[1]:TX Pkts 7824 bytes 250091 RX Pkts
1595 bytes 43999
*Mar  1 00:37:28.079: ephone-1[1]:Pkts lost 0 jitter 0 latency 0
*Mar  1 00:37:28.079: ephone-1[1]:Src 100.4.0.6 27928 Dst 100.4.0.1
2000 bytes 20 vad 250 G729
*Mar  1 00:37:30.763: STATS: DN 1 Packets Sent 7824
*Mar  1 00:37:30.763: STATS: DN 1 Packets Received 1595
*Mar  1 00:37:30.763: STATS: DN 1 Latency 0
*Mar  1 00:37:30.763: STATS: DN 1 PacketsLost 0
*Mar  1 00:37:32.619: ephone-1[1]:GetCallStats line 3 ref 3 DN 1: 1001
*Mar  1 00:37:32.871: ephone-1[1]:Call Stats for line 3 DN 1 1001 ref 3
*Mar  1 00:37:32.871: ephone-1[1]:TX Pkts 8064 bytes 257771 RX Pkts
1608 bytes 44168
*Mar  1 00:37:32.871: ephone-1[1]:Pkts lost 0 jitter 0 latency 0
*Mar  1 00:37:32.871: ephone-1[1]:Src 100.4.0.6 27928 Dst 100.4.0.1
2000 bytes 20 vad 250 G729
*Mar  1 00:37:34.819: STATS: DN 1 Packets Sent 8064
*Mar  1 00:37:34.819: STATS: DN 1 Packets Received 1608
*Mar  1 00:37:34.819: STATS: DN 1 Latency 0
*Mar  1 00:37:34.819: STATS: DN 1 PacketsLost 0
*Mar  1 00:37:37.663: ephone-1[1]:GetCallStats line 3 ref 3 DN 1: 1001
*Mar  1 00:37:37.915: ephone-1[1]:Call Stats for line 3 DN 1 1001 ref 3
*Mar  1 00:37:37.915: ephone-1[1]:TX Pkts 8316 bytes 265835 RX Pkts
1619 bytes 44311
*Mar  1 00:37:37.915: ephone-1[1]:Pkts lost 0 jitter 0 latency 0
*Mar  1 00:37:37.915: ephone-1[1]:Src 100.4.0.6 27928 Dst 100.4.0.1
2000 bytes 20 vad 250 G729
*Mar  1 00:37:41.443: STATS: DN 1 Packets Sent 8316
*Mar  1 00:37:41.443: STATS: DN 1 Packets Received 1619
*Mar  1 00:37:41.443: STATS: DN 1 Latency 0
*Mar  1 00:37:41.443: STATS: DN 1 PacketsLost 0

```

Table 7.9

Chapter 8

Theory v/s Experiment

In this chapter we will discuss in short the result that we were expecting and results what we got from the experiments.

8.1 GTS With WFQ

According to the theory, Generic Traffic Shaping (GTS) shapes traffic by reducing the outbound traffic flow to avoid congestion. WFQ dynamically classifies network traffic into individual flows and assigns each flow a fair share of the total bandwidth. Each flow is classified as a high bandwidth or low bandwidth flow. Low bandwidth flows, get priority over high bandwidth flows. WFQ is that the flows are automatically determined, and each flow gets a fair share of the bandwidth.

We proved this by applying GTS with WFQ on Gigabit Ethernet Interface. We applied parameters that are scant for a good quality of video. Working of WFQ was proved with voice data. Data with higher bandwidth (video) requirements start dropping after queue gets full and data with lower bandwidth (voice) requirements didn't drop at all. The result are already shown in *Table 5.4, 5.8 and 5.10*. Hence we proved what theory says.

8.2 CBTS With CBWFQ

According to the theory, in CBTS a traffic shaper typically delays excess traffic using a buffer, or queuing mechanism, to hold packets and shape the flow when the data rate of the source is higher than expected. CBWFQ differs from WFQ in allocating the bandwidth to different applications. In WFQ that is done automatically, But in CBWFQ that part is done manually. We can configure as much as bandwidth to certain applications.

To prove the theoretical results we divided this experiment into two parts. In the first part, we provided less than what was desired for the video class. We implemented less bandwidth as well as very fewer value of CIR. But we provided more than enough bandwidth and CIR value to other class, that was used by voice. Just as the theoretical results, traffic shaping becomes active in video class only but not in voice class. In the second part, we implemented almost same parameters. So due to that there was no

dropping in any of the classes. The principle of CBWFQ is to include user-defined classes. Each class has a separate queue, and all packets found to match the criteria for a particular class are assigned to that queue.

The most important thing that has to be noted is about the working of CBWFQ. As according to theory in CBWFQ each user-defined class is guaranteed a certain bandwidth, but classes that exceed that bandwidth are not necessarily dropped. Traffic in excess of the class's guaranteed bandwidth may use the "free" bandwidth on the link. This thing was proved with the statistics results of *Table 6.3* and *Table 6.5*. In *Table 6.3* class "voice" had free bandwidth, so class "data" will use that. This thing can be seen by comparing the number of packets in *Table 6.3* and *Table 6.5*. Hence we proved what theory says.

8.3 FRTS With PQ

According to the theory, with Frame Relay Traffic Shaping, users are able to configure rate enforcement to either the CIR level or to other user-defined values on a per-VC basis. In this way, bandwidth can be flexibly allocated to each VC and tailored to specific requirements. This allows the feature to provide a mechanism for sharing a common Frame Relay connection by multiple VCs over the same line. And Priority queuing (PQ) enables network administrators to prioritize traffic based on specific criteria. These criteria include protocol or sub-protocol types, source interface, packet size, fragments, or any parameter identifiable through a standard or extended access list.

To prove the theoretical results we implemented two classes and third class named as "class-default" was by default. Both classes were implemented on different DLCI's numbered 20 and 22. Class1 was configured to have the voice, with the help of access-list, and class-default had all the other data. When video was played between two end stations, the queue starts building in class-default only. This thing was proved in *Table 7.6* and *Table 7.7*. From *Table 7.9* and *Section 7.16.1*, this thing has been proved that priority was given to voice, due to that there was no dropping/latency to voice traffic, but there was delay and dropping of packets in class-default. . Hence we proved what theory says.

8.4 Summary

This chapter contains almost whole summary of the project. This chapter compares the expected results with results we got. It requires detailed study of the concepts and hard work to make the results accurate as desired.

Chapter 9

Conclusion and Future Directions

The Network Model used in the report was a manifestation of a real-time. This project was carried out to study the performance of traffic with traffic shaping configured on the network. Different types of traffic shaping had been implemented on the network, with different types of Queuing Mechanism. Two ISP's had been configured, with one a primary and one as backup ISP.

In GTS, WFQ had been configured. Video was streamed into the network and after that traffic shaping becomes active and according to WFQ, data with low bandwidth will be given priority over the data which requires high bandwidth usage. So in our experiment voice data was not effected by the queue, as voice is considered as low bandwidth usage data. But on the contrary video traffic experiences a delay and then dropping of packets.

In CBTS, CBWFQ had been configured. Two classes had been configured with different parameters. One with enough bandwidth(voice) and other with fewer bandwidth(video). As expected traffic shaping becomes active in data class only. Now we have to prove the functionality of CBWFQ, as according to CBWFQ one class can use the free bandwidth of other class, if it's own bandwidth is used up. We can see that thing from the statistics of two classes. When queue becomes full for data class, it used the bandwidth of voice class. It was proved in chapter 6.

In FRTS, PQ had been used. We configured rate enforcement to CIR level. In this way, bandwidth can be flexibly allocated to each VC. Classes had been configured on different DLCI's .PQ had been used in this experiment and voice was the data which got high priority over all other data. The class which was carrying data had been allocated sufficient amount of bandwidth. Video stream had been played and data start queuing up, but voice data had not been affected with that. There was no latency or loss in voice packets.

This study could be very useful in analyzing traffic shaping and its various effects on different types of data. Now a days many of the major ISP's are using one of the traffic shaping mechanisms. A subsequent migration to IPv6 offers better QoS guarantees for voice applications. The deployment of IPv6 in conjunction with RSVP facilitates QoS guarantees for real-time applications. These solutions, however, are still emerging and are not widely deployed.

Chapter 10

Bibliography

- 1.** Author: Michael E. Flannagan, Benoit Durand, Jerry Sommerville, Mark Buchmann and Ron Fuller. Title: “Queuing and Congestion Avoidance Overview”.
- 2.** Author: Carl Timm and Wade Edwards. Title: CCNP: Building Scalable Cisco Internetworks Study Guide.
- 3.** Author: Todd Lammle. Title: CCNP: Routing Study Guide.
- 4.** Author: Wade Edwards. Title: CCNP Complete Study Guide.
- 5.** Author: Danelle Au, Baldwin Choi, Rajesh Haridas, Christina Hattingh, Ravi Koulagi, Mike Tasker, Lillian Xia. Title: Cisco IP Communications Express: CallManager Express with Cisco Unity Express (Networking Technology).
- 6.** Author: Jonathan Davidson, James Peters, Manoj Bhatia, and Satish Kalidindi. Title: Voice over IP Fundamentals (2nd Edition).
- 7.** Author: Ramesh Kaza and Salman Asadullah. Title: Cisco IP Telephony: Planning, Design, Implementation, Operation, and Optimization (Networking Technology). ISBN : 1-58705-157-5
- 8.** Author: Giralt, Addis Hallmark, and Anne Smith. Title: Troubleshooting Cisco IP Telephony.
- 9.** Author: Benoit Durand, Jerry Sommerville, Mark Buchmann, Ron Fuller. Title: Adminstring Cisco QoS in IP Networks . ISBN: 1-928994-21-0
- 10.** Author: Pauline P. Francis-Cobley and Adrian D. Coward. Title: Voice over IP versus Voice over Frame Relay.
- 11.** Author: Fong, Paul J. Title: Configuring Cisco Voice over IP. [Electronic Resource]. 2nd Edition. ISBN: 1931836647
- 12.** Author: Anand, Vijay. Title: Cisco IP Routing Protocols. 1st Edition. ISBN: 1584503416
- 13.** Author: Donohue, Denise. Title: Cisco Voice Gateways and Gatekeepers. ISBN: 9781587052583
- 14.** Author: White, Russ. Practical BGP [Electronic Resource]. ISBN: 0321127005

- 15.** Author: Christina Hattingh. Title: Cisco Call Manager Express [Electronic Resource]
- 16.** Author: James Boney. Title: Cisco IOS in a Nutshell.
- 17.** Author: Cisco Systems, Inc. Title: Cisco IOS™ 12.0 Quality of Service. ISBN-10: 1-57870-161-9
- 18.** Author: Dave Hucaby, Steve McQuerry. Title: Cisco® Field Manual: Router Configuration. ISBN-10: 1-58705-024-2
- 19.** Cisco Documentation – Configuring Generic Traffic Shaping.
- 20.** Cisco Documentation – Configuring Class Based Traffic Shaping.
- 21.** Cisco Documentation – Configuring Frame Relay Traffic Shaping.
- 22.** Cisco Documentation – Configuring the MQS with Command-Line Interface.
- 23.** Cisco Documentation – Setting Up a Cisco CME System.
- 24.** Online Resources : <http://en.wikipedia.org>
- 25.** Online Resources : <http://proquest.safaribooksonline.com>