University of Alberta

Faculty of Electrical and Computer Engineering

Master of science in internetworking

QoS and Over-subscription for IP/MPLS Networks

Submitted by:
Wei Chen

Contents

Abstract

QoS (quality of service) as its name, it utilizes basic networking technologies such as classifying, queuing, policing, packet drop, and shaping to maintain the certain network quality based on SLA (Service-Level Agreement), and also supply better service and performance for networking communication. As the beginning of establishment of internet, the QoS was not to be recognized its necessity for the whole internet, it was only best effort service mode. When the traffic is getting dramatically increased, the best effort mechanism cannot reach the requirement of fast internet level any more and the transmitting data gets dropped. Therefore, the main point of QoS is to resolve the traffic delay and congestion issue. In accordance with limited the network resource and different requests of service, we need to set up respective rule to certain service to more efficiently utilize network resource such as based on packet priority level. QoS contains three diverse service mode: Best-Effort, Integrated service, and Differentiated service. In this paper, we will focus on each of them with different OSI layers (datalink layer, network layer, MPLS technology), and implement labs scenarios based on real-life scheme using Cisco devices and configuration concept. Furthermore, over-subscription will be as a business case to be introduced, and it is implemented by policing and shaping mechanism,

Acknowledgement

I would like to express my gratitude to Dr. Mike MacGregor and MR. Shawnawaz Mir for providing me the great opportunity to do this project" QoS and over-subscription for IP/MPLS Networks".

I am greatly thankful to Mr. Juned Noonari for his guidance and cooperation to work on this project and help me in overcoming the difficult I encountered during processing this project.

My sincere thanks to my friend who gave me helps and encouragement in the completion of this report successfully.

1. Introduction

1.1 Background

In the conventional IP network architecture, all the packets and data have been transmitted without any differentiation and been treated as equal. Every forwarding device (IP routers and LAN switches) without QoS mechanism addresses all the packets and data only by taking FIFO (first in first out) policy; furthermore, packets and data transmission to the destination is on Best Effort basis. Therefore, the reliability of the data transmission, data propagation latency, and the possibility of losing packet cannot be fully controlled and guaranteed.

As the development of internet and dramatically increased internet applications and complexity, it brings more new requirements and challenges to the quality of service of internet. For instance, VOIP and some similar real-time services are more sensitive depending on the data propagation delay. If there is a huge delay between end-to-end customers, customers will not be satisfied with what its function is supposed to be, such as facetime, video streaming. Hence, to meet the different requests of service and avoid the network congestion on limited bandwidth, it demands the network itself to have a capability distinguishing the type of communications and give them a priority to decide what kind of data should be transmitted first along the network path or supply more bandwidth when the congestion occurs.

Before converged networks: traditional telephone service using individual line.

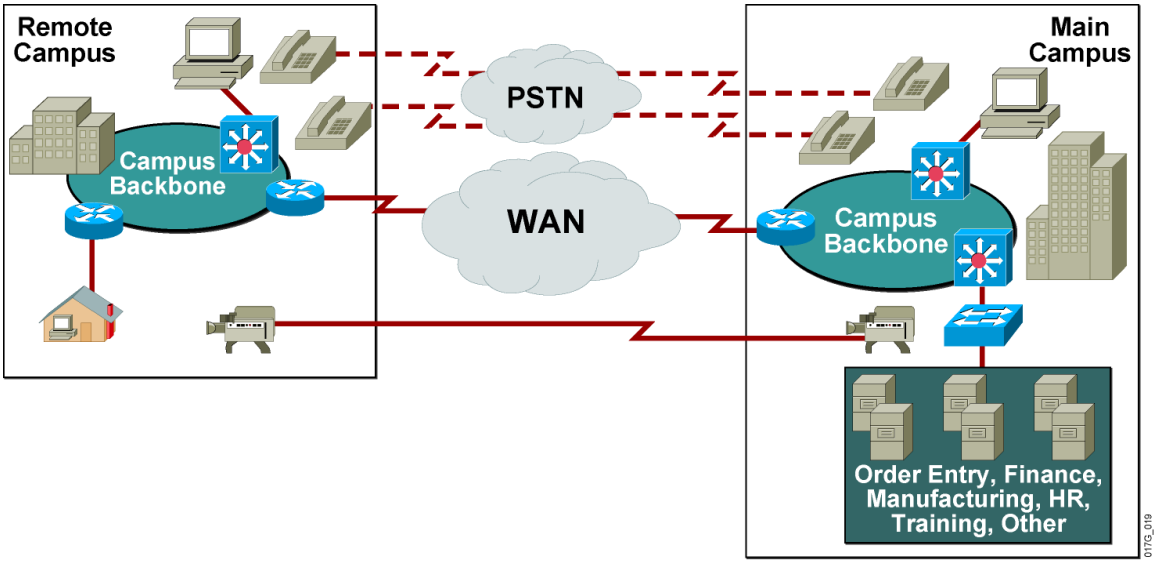


Figure 1-1

http://www.slideshare.net/proydesa/cisco-qos

Data traffic characteristics: bursty data flow, first in first on, not sensitive for delay, acceptable brief outages.

After converged networks: all of service are sharing one line.

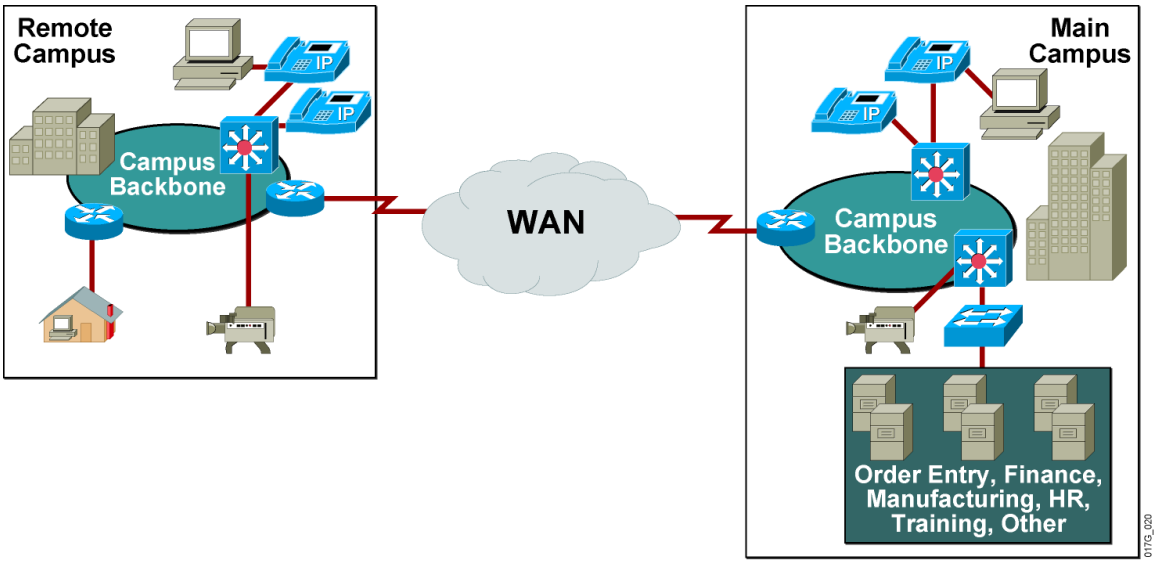


Figure 1-2

http://www.slideshare.net/proydesa/cisco-qos

Data Traffic characteristics: voice packet over IP competes with bursty data flow, critical traffic has high priority, time-sensitive for voice and video, brief outages not acceptable.

The common issues are listed as following:

- Lack of bandwidth – bottleneck of the link, $bandwidth_{max}$=min (bandwidth along the link)
- End-to-End delay – the sum of all the propagation, processing, and queueing delay
- Variation delay (jitter)
- Packet loss

As a consequence, the appearance of QoS is to devote to solve the above issues.

1.2 QoS definition

What is QoS?

QoS is a set of techniques managing the network and working on serval layers for providing predictable performance and improving better service and capacity to particular network communication by using kinds of basic computer or telephony network techniques, also it is for solve issues such as network congestion, packet loss, jitter, bandwidth shortage, and delay. Quality of Service is all about providing managed unfairness to certain forms of traffic in the network compared to others.

Under normal circumstances, if the network only works for particular no timeout application system like web application, E-mail, it doesn't require QoS necessarily. However, when network overloads and congestion occurs for some real-time and critical applications, QoS can guarantee the important data and packets will not get dropped and delayed; meanwhile, it insures the network running high-efficiency.

1.3 The advantages of QoS

- Fully customize the traffic, prioritize the important packets to be transmitted
- Control over the distribution of network resource, something like limiting the bandwidth consumption of FTP transfers to give resource to more important data access.
- More efficient use of network resource, making policies to efficiently utilize resource when they are being vacant. For example, one type of service has been assigned appropriate bandwidth (less than total link bandwidth) to guarantee its performance. When there are no any other services using the link resource, that type of service can take more bandwidth to facilitate the performance.
- Provide low latency and sufficient bandwidth to time sensitive data, even other traffic transmitting occupies the link resource simultaneously on the path.

1.4 QoS service mode

- Best-Effort service
- Integrated service (IntServ)
- Differentiated service (DiffServ)

1.4.1 Best-Effort Service

Best-Effort is a simplest service mode, and this approach is not applying any congestion management or avoidance tools. Application can send any size of packets and data anytime without any pre-process, classification. For Best-Effort service, the network is sending packets and data in best capacity, but it cannot insure the low packet delay and transmission reliability. In fact, the default congestion management in most device is used FIFO (first in first out). The very first packets to arrive, are those packets that are sent out first. Notice the problem is voice and video packets might be stuck behind many less important packets in the limited bandwidth as packets accumulating in the queue causes extreme congestion and insufficient resource on the link. It applies for most of network application through default mode FIFO as above such as FTP, E-mail, etc.

1.4.2 Integrated Service (IntServ)

Another approach to QoS called IntServ which is a comprehensive service mode that can fulfil various QoS requests. Under this IntServ approach, prior to application sending actual packets and data, application on the link needs to send

a signal to network along the path to each device to request particular reserved path for certain service. The implementation of the request by protocol RSVP (resource reservation protocol) to establish the reserved path on the network. At the very beginning, application advertise its own flow parameter and particular service quality request including bandwidth reservation, latency and so on. After the device on network received the request of the application, they started to check resource distribution based on application request and resource availability of the network to decide whether deploy the appropriate resource or not. Since the network confirm the request, it will be for every flows (consist of IP address and port number) carrying out packet classification, traffic policing and shaping, queueing, and scheduling. After the reservation for flows, the application starts to communicate along the path, as long as the packet in the range of the flow parameter bound, the network will be guarantee to meet QoS requirement of the application.

IntServ supplies two service as flowing:

- Guarantee service: insure to provide bandwidth and delay restriction to implement the application requirement. For instance, VoIP application reserves 10M bandwidth and less than 1 second delay.
- Overload control service: when overload or network congestion occurs, it ensures the particular application packets have low latency and high priority to be forwarded.

This approach to QoS seems definitely ideal; nevertheless, IntServ mode is almost not scalable comparing with DiffServ. All device and system in the environment must support RSVP and configured appropriately in order to make the reservation. This is really difficult to guarantee as the network system and configuration from enterprise to enterprise.

Furthermore, there is a disadvantage associated with device hardware requirement, to maintain the appropriate reservation demands CPU and memory consumption. In order to keep the resource reserved, it also brings more resource consumption along the path. The traffic only can be separated to flows rather than each packet invoked by queuing mechanism. It has less flexibility of process various traffic.

In the paper, IntServ will be significantly covered, but I will give a template for lab design environment.

1.4.3 Differentiated Service (DiffServ)

DiffServ is multi-function service mode which can implement different type of QoS requirement, and it does not require open a path for resource reservation against IntServ. This approach is the major focus of QoS because its main features of DiffServ are effectivity and scalability. It not only simply serves for single type of flow, which means the traffic would be divided into various types of packets,

and the packets will be forwarded sequentially and put into appropriate queues based on their identification (DSCP). Different types of traffic are recognized by network and provide different levels of service to these traffic. Every packets are using DSCP (Differentiated Service code point) in IP packet header or IP precedence, CoS in frame to supply particular service respectively. For instance, DiffServ provide low-latency service to prioritize time-sensitive traffic such as VoIP or video conferencing but providing less prioritized service for less critical traffic. In response to these classifications and markings, routers and switches use various queueing strategies to tailor performance to expectations. We will break down of DiffServ into serval categories and expatiate on each of them such as classification, marking, queueing, policing, shaping, and so on.

Differentiated Service Architecture:

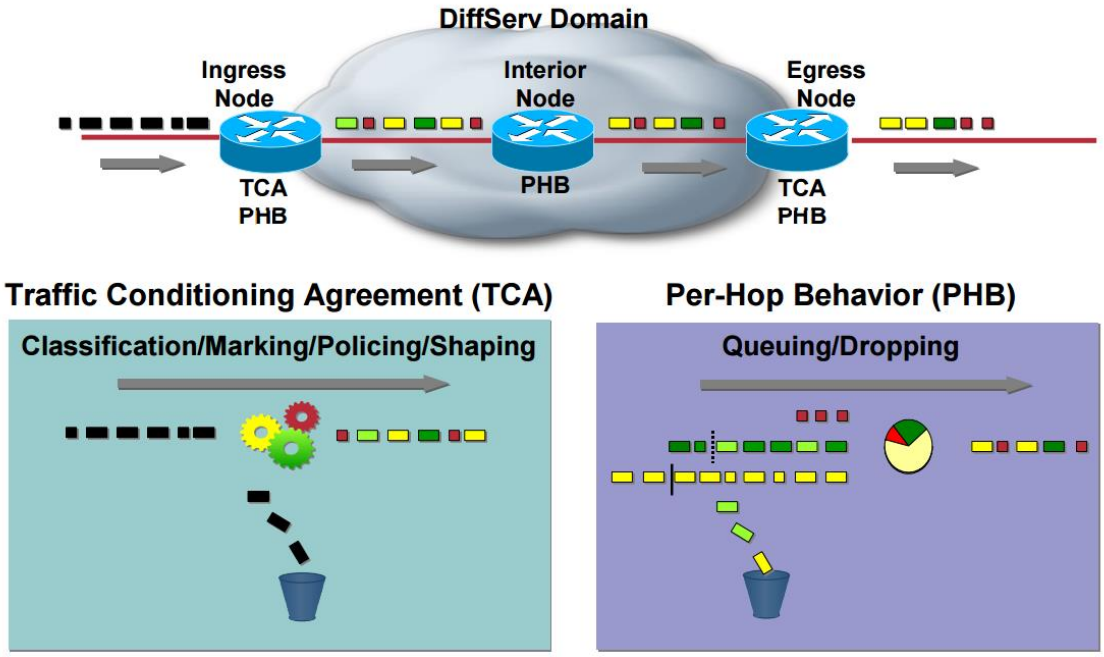


Figure 1-3

https://www.nanog.org/meetings/nanog36/presentations/sathiamurthi.pdf

- Classification: The component of a QoS feature that recognizes and distinguishes between different traffic flows. Without classification, all packets and traffic will be treated as the same.
- Marking: The component of a QoS feature that can color the packet(frame) or flows in order to be differentiate from other packets. Usually markers in IP header or frame header: CoS, DSCP/IPP, MPLS EXP.

- Queueing: congestion management, default as FIFO. When congestion occurs, it facilitated to determent which packets will be de-queued first based on the packet parameters.
- Dropping: default as tail drop. When the queue is overloaded, in order to keep the efficiency of data transmission properly, some of packet will get dropped. Random early detection and weighted RED can be customized enabled.
- Policing: introducing token bucket algorithm, conformed packets being limited in CIR to transmit, and drop incompetent packets if insufficient tokens to fulfil incoming packets.
- Shaping: smooth bursty traffic to reach CIR, also using token bucket algorithm.

2. QoS traffic requirements

2.1 SLA – Service-level agreement

SLA is a signed contract between customers and service provider including standardized service parameters – scope, quality, responsibilities, and performance, and it usually get two or more parties involved. The agreement commonly defines detail content containing minimum service assurance, performance measurement, problem management, warranties, disaster recovery, and customer duties. According to SLA requirement, the service provider applies to the agreement to monitor and manage network capacity and flows. SLAs have been used since late 1980s by fixed line telecom companies as part of their contracts with their customers. This approach has generalized such that now it is common for a customer to engage a service provider within a service level agreement in a wide range of service contracts in practically all industries and markets.

2.2 Voice traffic

Characteristics of Voice traffic:

- Smooth and benign traffic
- Sensitive to packet dropping
- Sensitive to delay
- UDP
- Real-time

The requirements for one-way Voice traffic:

- Latency ≤ 150 ms of one-way from sender to receiver
- Jitter ≤ 30 ms
- Packet loss ≤ 1%
- 17-106 kbps guaranteed priority bandwidth per call

- 150bps plus layer 2 overhead guaranteed bandwidth for voice-control traffic per call

In accordance with the detail of needs on IP SLA, John Evans and Clarence Filsfils wrote G.114 states that 150 ms end-to-end one-way delay meet the most of telephony service without a perceivable degradation in voice quality.

Some carries make their target latency as 100 ms, a usual maintenance is 150 ms delay, and Enterprise VoIP networks tend to have a looser target as 250 ms.

2.3 Video traffic

Characteristics for video traffic:

- Bursty
- Lager bandwidth occupied
- More bandwidth required
- Videoconferencing for drop and delay sensitive
- UDP

There are two types of video traffic:

- streaming video: only receiving - IP multicast, real-time training activities, and executive broadcasts, such as watching videos on YouTube, software training Webinar
- videoconferencing: requiring two-way traffic as receiving data and responding data - either one-to-one or one-to-multipoint conference chatting such as facetime.

The requirements for one-way streaming video as following:

- allows 4 to 5 seconds delay depending on application's buffer size
- 2% packet loss acceptable depending on the encoding and the rate of the video stream
- Insensitive for delay and jitter
- Video content duplicated to distribute
- Large file transfers
- Restrict to distribution to less-busy times of day

The requirements of videoconferencing:

- Latency $\leq$ 150 ms
- Jitter $\leq$ 30 ms
- Packet loss $\leq$ 1%
- Minimum priority bandwidth assurance: video stream + 20%, for example, a 400 kbps stream might require 480 kbps of priority bandwidth

2.4 Data traffic

Except Voice and Video data, we also consider other traffic. Different applications have different traffic characteristics; different versions of the same application also might have different traffic characteristics.

- Bursty or smooth
- Less bandwidth demand or greedy for bandwidth
- Insensitive of packet dropping and delay
- TCP retransmission

Data are classified into relative priority level:

- Mission critical: transactional, IP routing, network management, and locally defined high-priority applications
- Guaranteed bandwidth: streaming video. Interactive traffic, and intranet
- Best-effort: web browsing, e-mail, unspecified applications
- Less-than-best-effort: FTP, backups, and peer-to-peer applications

Now the traffic has been classified into different priority levels, the network will create specific policies depending on these privilege levels to identify and forward the packets. The network is not only easily transmitting the packets by FIFO without classification along the path. Furthermore, a critical role in the service provider delivery is the SLA as above, affecting the ability to support drop and delay-sensitive classes, as seen in voice and video requirements. To implement the best throughput for each type of traffic, the service providers have to be strictly in compliance with SLA as detail mentioned in the contract.

2.5 Delay

The data propagated along the path might generate all kinds of delays. In general, the delays can classify into: transmitting delay, processing delay, queueing delay. The transmitting delay is the time consumption of the data propagating along the path, which depends on the bandwidth, distance, and physical connection. In Best-Effort network, the transmission delay is fixed, processing and queueing delays are unpredictable. To reduce the transmitting delay, it requires the link updating which is actually physical cable upgrade. For example, the ISP would renew the cable through cities or long distance of territory with new physical connection. Currently, using optical fiber carrying data replaces the old one coaxial cable. However, it brings a huge difficulty-financial and time consumption. Therefore, using bandwidth efficiently become more considerable and is always significant to fully deploy the resource. Furthermore, processing and queueing delay occurs in the device. When the data has been arrived at the input interface of device, the device has to process the data to decide which packet will be transferred to specified output interface and might mark the data to put into appropriate queue. Also, since the data has been at queue,

the queue has its own policy to prioritise the critical data to be transmitted first. There are many methods to avoid and reduce those delay such as congestion management (queueing), congestion avoidance (packet dropping mechanism), and traffic policing and shaping. We will take a deeper look on them later on this paper.

3. Classification and marking

3.1 Classification

Prior to marking the data, the traffic has been to be classified in order to provide a prerequisite for data obtaining diverse treatments and queueing policy in network devices and along the network path. Classification is one of QoS components and the first sept to identify different flows of traffic from different applications and protocols. In the typical network environment, the traffic consists of all kinds of flows generated by multiple applications and devices. Many of these applications have their own unique characteristics. Therefore, to classify the traffic based on their characteristics, it allows that each class of traffic to obtain a different level of treatment other than other traffic. Without classification, all traffic and packets will be treated equally. After the traffic has sorted into different classification, the network administrator can apply various policies according to the requirement of the network. The network devices distinguish the level of service by policies and take actions for forwarding and processing the traffic.

Classification options will be applied into Cisco devices command configuration. A class-map is created to match each criterion by the MQC (Modular of Quality of Service CLI). Traffic classification by Differentiated service implementation as figure:

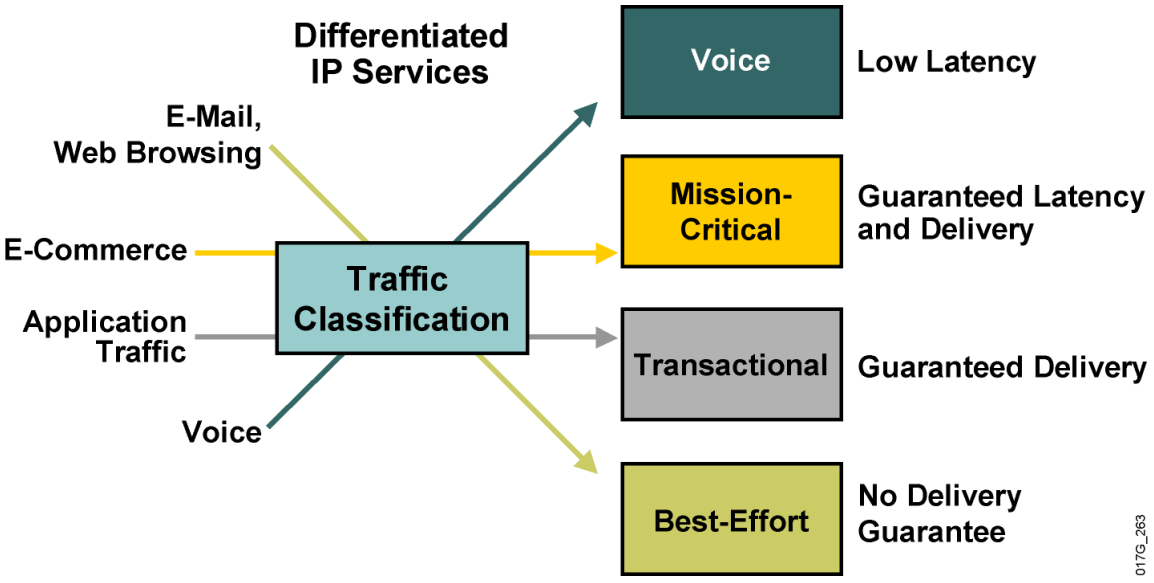


Figure 3-1

http://www.slideshare.net/proydesa/cisco-qos

There are many methods to implement the classification and identification mainly based on field in packet, flow parameters, and IPP/DSCP, as following options of classification include:

- Access list: classifying the traffic by source host/port, destination host/port, TCP, UDP, ICMP, and so on.
- IP precedence/DSCP value: DSCP is backward compatible with IPP. In IPv4 packet TOS byte field, IPv4 header defines the first 3 bits for IPP which has 8 types of service with value 0 – 7 and the first 6 bit for DSCP. The related detail of IPv4 packet, IPP/DSCP value will be showed in the marking section.
- IEEE 802.1Q/ISL CoS/Priority value (Layer 2): 3 bits PRI in 802.1q Ethernet frame header encode 8 types of service level
- QoS group number
- MPLS experimental bits: 3 bits EXP in 32-bit MPLS header for QoS, 8 classes of service as well.
- Protocol/NBAR (Network Based Application Recognition): class-map has lots of options matching different protocols and Applications.
  NBAR performs serval functions such as traffic statistic provision, protocol discovery, and identifying application and protocol through layer 4 to layer 7. It can classify applications that use statically and dynamically assigned TCP or UDP numbers, Non-TCP and Non-UDP IP protocol such as ICMP, EIGRP, and deep packet inspection such as HTTP traffic carrying a certain type of payload (.wav)
- Frame Relay DE bit
- Input interface: in class-map to match all traffic received through input interface.
- Source/Destination MAC address

3.2 Marking

Since the traffic is classified, the marking will be taking place in the traffic so that it is easy to be distinguished and treated with appropriate QoS policy at each device in the network infrastructure. In order to know how and the traffic actually been marked and where the marking been made in the frame or packet, we will go through and analyse each of layer 2, layer 2.5, and layer 3 packet and frame.

3.2.1 Layer 2 - data link layer marking

In Ethernet frame header of VLAN Tag (802.1q) defines an 8 levels service (CoS, Class of service). Meanwhile, a 3-bit field called Priority Code Point specifies a carrying CoS value between 0 and 7 that can be used by QoS rules to differentiate traffic, and these bits are also commonly referred to 802.1p. A sample of frame structure as below figure:

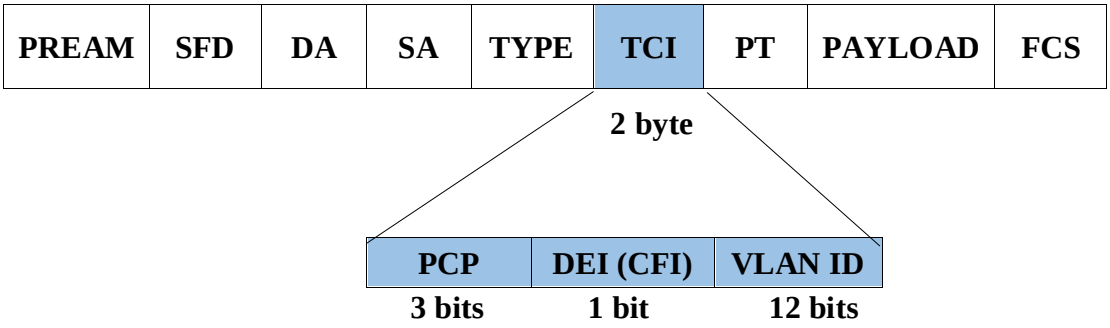


Figure 3-2

http://www.cisco.com/c/dam/en/us/products/collateral/ios-nx-os-software/quality-of-service-qos/prod_presentation0900aecd80312b59.pdf

TCI (tag control information): 2 bytes in Ethernet frame including PCP (PRI), DEI (CFI), and VLAN ID.

DEI (Drop eligible indicator): a 1-bit field. Formerly called CFI, usually working with PCP to indicate frames whether should be dropped when the congestion occurs.

VLAN identifier: a 12-bit field to indicate which VLAN the frame belongs to.

PCP (priority control point): 3 bits used for 802.1p class of service referring to the frame priority level. 8 priority values to be used for prioritizing different traffic. 8 class of service definition in figure:

Traffic type	Service characteristic	CoS Marking	Protocol
Network Control	**Applied to network maintaining and data reliability, and requires low drop probability**	**7 reserved**	**BGP, PIM, SNMP**
Internet work control	**Requires low latency and drop probability**	**6 reserved**	**STP, OSFP, RIP**
Voice	**Applied to voice service, normally requires latency and jitter < 10 ms**	**5**	**SIP, MGCP**
Video	**Video service requires latency and jitter < 100 ms**	**4**	**RTP**
Critical Applications	**Applied to guarantee minimum bandwidth service**	**3**	**NFS, SMB, RPC**

Excellent Effort	**information organization sending out message to important costumers**	**2**	**SQL**
Best Effort	**Default class of service, not require prioritizing service, takes its best effort to transfer the traffic**	**0**	**HTTP**
Background	**Applied to bulk transfer service**	**1**	**FTP, SMTP**

Table 3-3

One thing has to be aware of is CoS value of 7 and 6 are usually reserved on Cisco devices and these are never assign to traffic. These marking only been used by devices themselves and reserved these values for traffic control.

3.2.2 Layer 2.5 – MPLS Marking

In here, we consider MPLS is referred to layer 2.5, because its header inserted between layer 3 and layer 2. In nowadays technology, MPLS is commonly been used in ISP core network, and it brings more efficient and faster service for packet switching with label pushing and popping. As same as frame header, it reserves 3 bits in MPLS header class EXP (the experimental bits) to carry the CoS or DSCP value as marking purpose within prioritizing the service level. As below shown the figure MPLS header structure:

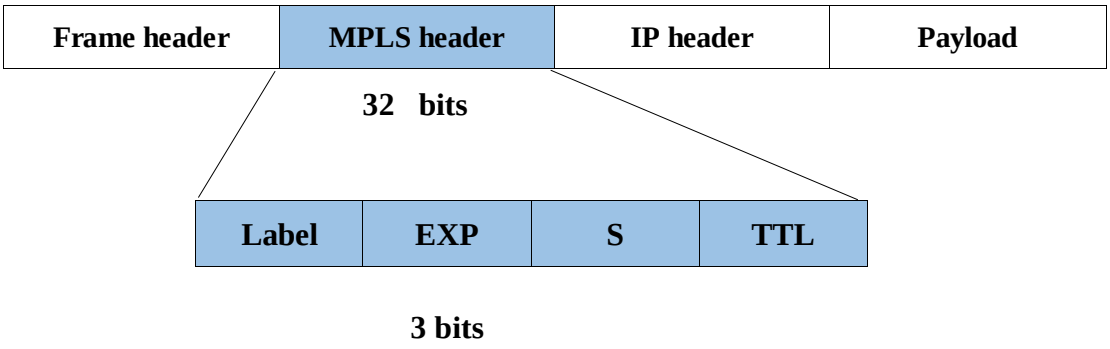


Figure 3-4
http://www.cisco.com/c/dam/en/us/products/collateral/ios-nx-os-software/quality-of-service-qos/prod_presentation0900aecd80312b59.pdf

Experimental: 3 bits used for class of service consists of 8 service levels. By default, Cisco device copies the three most significant bits of the DSCP or IPP to EXP field.

The allocation of EXP value in MPLS header:

Traffic type	EXP
Network control	7 reserved
Internetwork control	6 reserved
Voice	5
Video	4
Critical data	3
Transaction data	2
Bulk data	1
Best effort	0

Table 3-5

http://babaawesam.com/2013/08/07/mpls-vpn-qos-with-gns3-and-virtualbox/

3.2.3 Layer 3 - IP Precedence/DSCP marking

In traditional network, a 3-bit field is used for different levels of service in 1 byte ToS field. As the network infrastructure is becoming more complicated, there are more classifications of service to fulfil the requirement of the traffic complexity in the network. Thus, DiffServ with extensions is implemented within adding up to 6 bit for DSCP. DSCP is backward compatible with IP Precedence. IPv4 packet header as figure:

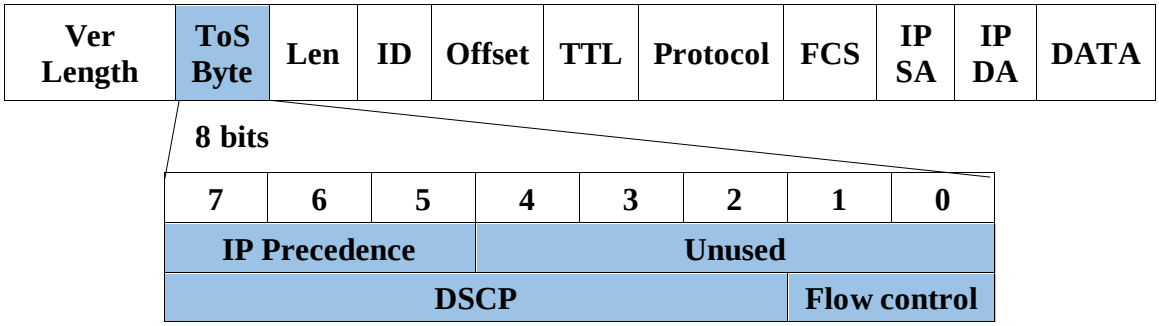
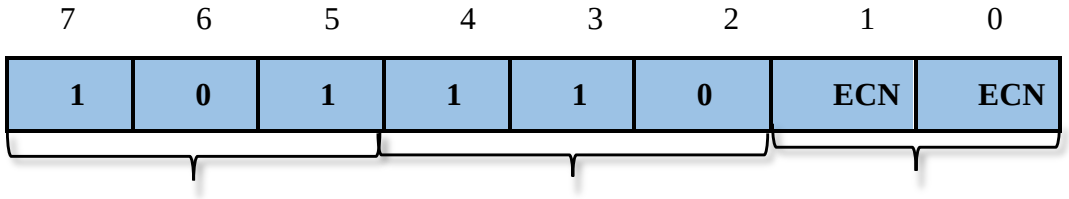


Figure 3-6

http://www.slideshare.net/proydesa/cisco-qos

DSCP is taking 6 bits for different type of service. CS is compatible with IPP.

In implementing of DiffServ network, each forwarding device decides its forwarding behavior in accordance with DSCP value in IP packet. It also can be called Per-Hop behavior. ToS field in IPv4 packet defines PHB in figure:



Default=000000 **Class Selector(IPP)=000** **Flow Control**

Expedited forwarding=101110

Assured forwarding=001xx0,010xx0,011xx0,100xx0

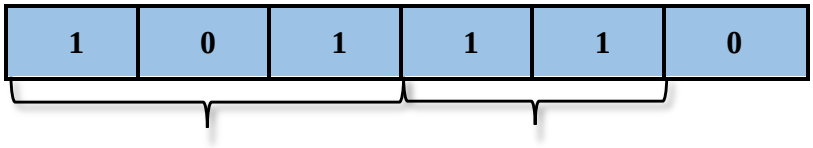
Figure 3-7

https://en.wikipedia.org/wiki/Type_of_service

The first 3 bits define IP Precedence, and the first 6 bits are for DSCP. DSCP is backward compatible with IP Precedence.

There are four mainly used forwarding behavior (traffic categories) as following:

- Expedited forwarding: dedicated to low latency, jitter, and packet loss service. These characteristics are usually using in voice, video and other real-time service. The reserved bandwidth is guaranteed. DSCP value for EF is 101110.



5 **No Drop probability**

Figure 3-8

https://en.wikipedia.org/wiki/Type_of_service

- Assured forwarding: in this approach, the service using this forwarding behavior provides assurance of delivery as long as the traffic does not exceed maximum allowed bandwidth. Once the traffic that exceeds the maximum subscribed rate and congestion occurs, the forwarding behavior separates 4 AF classes and each of them subdivided three drop precedence (low, med, or high, high precedence means more probability of dropping packet). If the congestion occurs between different class, the traffic in the high class is assigned high priority and more balanced queueing algorithms are likely to be used such as fair queueing or weighted fair queueing. Also, if congestion occurs within a class, the packets with

higher precedence are dropped first. The first 3 bits are assigned for queueing, and the 2 bits behind are for congestion avoidance.

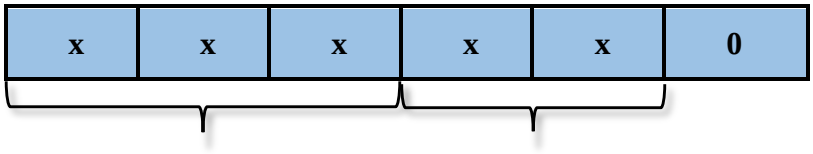


Figure 3-9
https://en.wikipedia.org/wiki/Type_of_service

AF behavior group figure:

	Class 1	Class 2	Class 3	Class 4
Low drop probability	**AF11(DSCP 10)**	**AF21(DSCP 18)**	**AF31(DSCP 26)**	**AF41(DSCP 34)**
Medium drop probability	**AF12(DSCP 12)**	**AF22(DSCP 20)**	**AF32(DSCP 28)**	**AF42(DSCP 36)**
High drop probability	**AF13(DSCP 14)**	**AF23(DSCP 22)**	**AF33(DSCP 30)**	**AF43(DSCP 38)**

Table 3-10
https://en.wikipedia.org/wiki/Differentiated_services

- Class Selector: CS value is a continuation of definition of Precedence field and equivalent to IP precedence, CS value is being used in TOS field in IPv4 header to prioritize traffic and in order to make it compatible backwardly with other old network devices. In addition, in TOS byte, the class selector code is taking first 3 bit to encode CS value. It also respectively maps to IP precedence such as cs1 to IP precedence 1. CS6 and CS7 are default pointed to network control protocols such as OSPF, BGP because maintaining network connection is always top priority.
- Best Effort: As the default forwarding behavior (FIFO tail drop), it mainly uses for the service that is not sensitive about delay, jitter, and packet loss. Typically, the default forwarding behavior has best-effort forwarding characteristics. Any traffic and packets that does not match the defined classes is belong to default forwarding behavior. This approach has been applied to most of network such as FTP, E-mail and so on by FIFO to implement. We will talk about DiffServ more later in this paper. DSCP value is 000000.

Here is IPP and DSCP value to mark certain type of service, and the mapping between CoS, IPP and DSCP:

Traffic type	PHB	IPP	DSCP	CoS
Network Control	**CS7**	**7**	**56**	**7**
Internetwork Control	**CS6**	**6**	**48**	**6**
Voice	**EF**	**5**	**46**	**5**
	AF43	**4**	**38**	**4**
	AF42	**4**	**36**	**4**
Video conferencing	**AF41**	**4**	**34**	**4**
Video streaming	**CS4**	**4**	**32**	**4**
	AF33	**3**	**30**	**3**
	AF32	**3**	**28**	**3**
Critical Data	**AF31**	**3**	**26**	**3**
Call Signaling	**CS3**	**3**	**24**	**3**
	AF23	**2**	**22**	**2**
	AF22	**2**	**20**	**2**
Transaction Data	**AF21**	**2**	**18**	**2**
Network Management	**CS2**	**2**	**16**	**2**
	AF13	**1**	**14**	**1**
	AF12	**1**	**12**	**1**
Bulk Data	**AF11**	**1**	**10**	**1**

Scavenger	**CS1**	**1**	**8**	**1**
Best Effort	**0**	**0**	**0**	**0**

Table 3-11

https://www.tucny.com/Home/dscp-tos

Packet can be classified and marked at layer 2 and layer 3 using different mechanisms such as IPP, DSCP MPLS EXP, and CoS. The packet classification is a QoS mechanism responsible for distinguishing different traffic flows, and the marking is to color the packets so it can carry the distinguishable mark against other different packets during the application of QoS to decide PHB.

4. Congestion management

In the data communication network, the links are shared by many of computers for data propagation. In addition, normally, the bandwidth of WAN is less than the bandwidth of LAN. As result of it, the data cannot always maintain the same data rate when data transmission is from one LAN to another through WAN, so it might cause a congestion. As the reduction of bandwidth along the path, some of packet arrived at device and cannot be fully transmitted due to the congestion occurring. For example, as figure shown below:

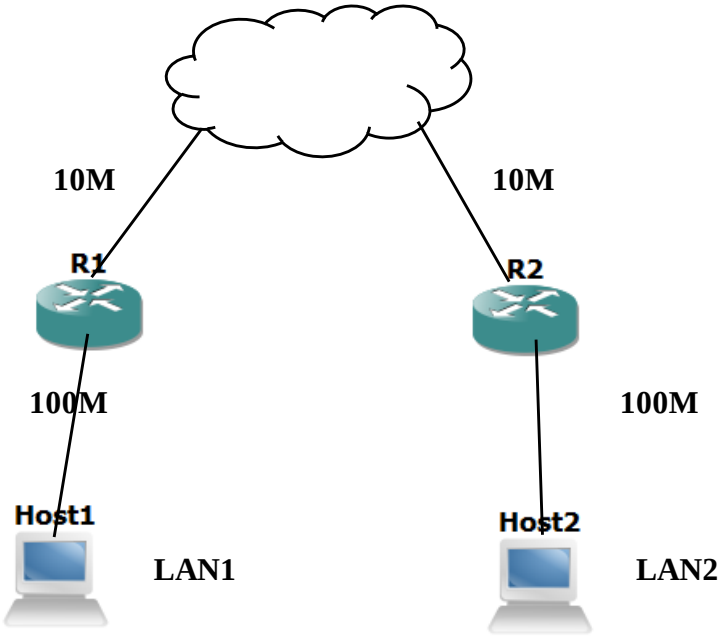


Figure 4-1

The congestion occurs on R1 when traffic is propagated with speed mismatch or traffic aggregation from 100M link to 10M link. 10M link is also called bottleneck with minimum bandwidth on whole path. We have to notice that a typical cisco network device interface having a software queue and a hardware queue for outbound traffic propagation. The hardware queueing system always uses FIFO queueing, and the software depends on how you configure and perform it. Only if the hardware queue is fully occupied, then the software queueing will be used to manage the congestion. The software queue works as a buffer, where the packets store temporarily until they are transferred. Actually, the length of the hardware queue can be adjusted with *tx-ring-limit* command. These two queueing components consist: hardware queueing and software queueing.

Congestion management means using approaches to administrate and control when the device cannot process that much flow. The approach to facilitate solving the congestion issue is queueing concept, the explicit procedures of this method include queue creation, packet classification, and packet allocation to appropriate queue, and scheduling queues. When the data rate exceeds the bandwidth of the interface, congestion management mechanism will classify the traffic and allocate them into different queues with priority value in packets or frames to process traffic sequentially, which the packets have higher priority will be treated first forwarding. FIFO (First in First out), PQ (Priority Queueing), CQ (Custom queueing), FQ (Fair queueing), WFQ (Weighted fair queueing), CBWFQ (Class-based Weighted fair queueing), LLQ (Low latency queueing).

4.1 FIFO

First in First out queueing is a default best effort approach in software and hardware queue, and all traffic receive equal treatment regardless of whether the traffic been classified or not. When the time-sensitive and important traffic arrived the outbound traffic interface with a period of congestion occurring, the packets might get stuck far behind the less important packets. As result of it, a huge delay would be easily generated. Furthermore, FIFO queueing only use one queue to process packets, all packets will be put into one single queue. If the queue has been filled up, it will take tail-drop method to avoid overload with incoming packets. The packets can be transferred first that really depends on the sequencing of the packets arrival. As default in cisco devices, the FIFO queueing is a default applying to interfaces except the serial interface, which uses Fair-queueing instead. We will cover FQ in next section. Now, here is the illustration of FIFO:

Figure 4-2

http://h3c.com/portal/Products___Solutions/Products/Switches/H3C_S5120-EI_Series_Switches/White_Paper/200812/689004_57_0.htm#_Toc215923183

P1: non-important packet P2: normal packet P3: time-sensitive packet

The critical traffic is not receiving any special treatment, the quality of that traffic cannot get guaranteed.

In cisco device, the default FIFO is using in the interfaces which the bandwidth is more than 2.048 Mbit/sec. Otherwise, the default setting will be Fair queueing.

Example as the following:

```
R1(config)#do sh int f0/0
FastEthernet0/0 is administratively down, line protocol is down
  Hardware is DEC21140, address is ca01.31e4.0000 (bia ca01.31e4.0000)
  MTU 1500 bytes, BW 100000 Kbit/sec, DLY 100 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation ARPA, loopback not set
  Keepalive set (10 sec)
  Half-duplex, 100Mb/s, 100BaseTX/FX
  ARP type: ARPA, ARP Timeout 04:00:00
  Last input never, output never, 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: fifo
  Output queue: 0/40 (size/max)
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     0 packets input, 0 bytes
     Received 0 broadcasts (0 IP multicasts)
     0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored
     0 watchdog
     0 input packets with dribble condition detected
     0 packets output, 0 bytes, 0 underruns
     0 output errors, 0 collisions, 0 interface resets
     0 unknown protocol drops
     0 babbles, 0 late collision, 0 deferred
     1 lost carrier, 0 no carrier
     0 output buffer failures, 0 output buffers swapped out
```

Figure 4-3

4.2 Priority Queueing

Priority queueing identifies the classification of traffic and put them into certain queue. Eventually, all packets will be classified into up to 4 classes and allocate the packets respectively in one of these queues. Four PQs are high, medium, normal, low priority queue. PQ always schedules the packets in the high priority queue to be sent out first until this queue becomes empty. In turn, the medium queue is starting processing packets, and rest of queue does same thing. The packets belong to high priority queue always are transferred first when the congestion takes place, which is benefit for the critical traffic. However, it will cause a starvation of other queue. The packets in the lower priority queue might get stuck if the higher queue never being empty.

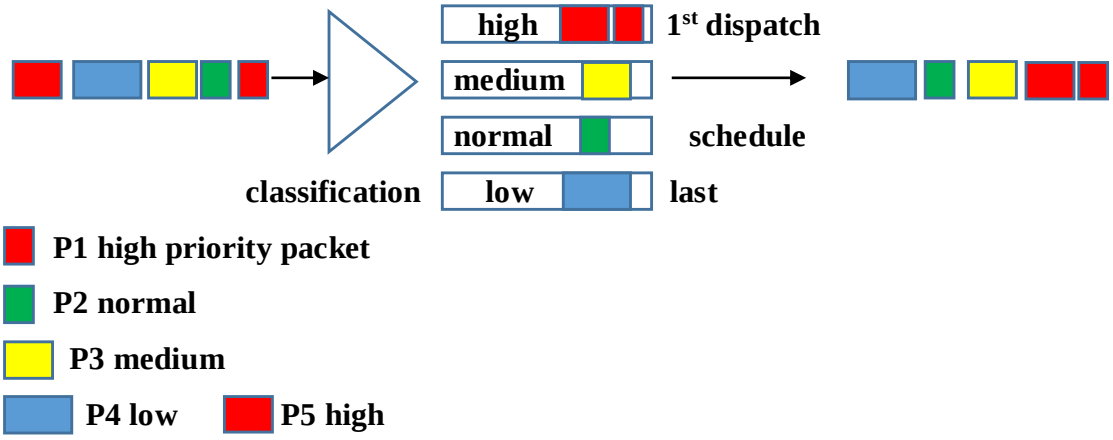


Figure 4-4

http://h3c.com/portal/Products___Solutions/Products/Switches/H3C_S5120-EI_Series_Switches/White_Paper/200812/689004_57_0.htm#_Toc215923183

The PQ mechanism as above illustration. Prior to queueing of packets, the order of packets: P1 P2 P3 P4 P5. The packets go through classification and put into certain queue. Afterward, the sequence of packets is rearranged to P1 P5 P3 P2 P4.

The example by using Cisco configuration:

Versus PQ always scheduling the packets in high priority queue, there are extensions of queue scheduling mechanism as Round Robin algorithm and Weighted Round Robin algorithm.

Round Robin is a scheduling fashion which has multiple queues which don't have prioritization, and dispatch one packet from each of queues in turn and perform a same scheduling.

Weighted Round Robin embeds a weight algorithm to each queue, the packets are dispatched by weight. For example, four packets dispatch from queue 1, two from queue 2, five from queue 3, 1 from queue 4, and it all depends on how the queue being configured and the weight assigned.

Here I will give a quick lab to show how PQ works:

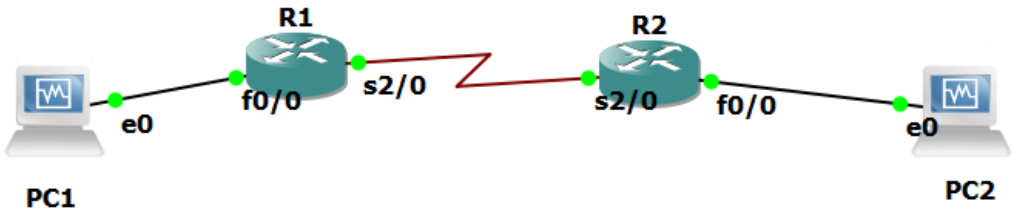


Figure 4-5

The packets should be scheduled first in the high queue. If there is no high queue, then process medium queue. the same thing is for low queue. In this scenario, PC1(192.168.1.1/24) will send traffic to PC2(192.168.2.1/24) with 4 different types of traffic flow. The bandwidth between PC and routers is 100M and between routers is 1.54M. The interface s2/0 on R1 will execute PQ and monitor the bandwidth distribution.

1st creates 4 access-list to match different traffic:

```
R1#sh access-lists
Extended IP access list 101
    10 permit tcp host 192.168.1.1 host 192.168.2.1 eq 5001 (973 matches)
Extended IP access list 102
    10 permit tcp host 192.168.1.1 host 192.168.2.1 eq 5002 (908 matches)
Extended IP access list 103
    10 permit tcp host 192.168.1.1 host 192.168.2.1 eq 5003 (221 matches)
Extended IP access list 104
    10 permit icmp host 192.168.1.1 host 192.168.2.1
```

Figure 4-6

2nd defines priority lists to assign the traffic to relative queue

```
R1#sh queueing priority
Current DLCI priority queue configuration:
Current priority queue configuration:

List   Queue  Args
1      high   protocol ip          list 101
1      medium protocol ip          list 102
1      normal protocol ip          list 103
1      low    protocol ip          list 104
R1#
```

Figure 4-7

By default, the queue size for each queue is high – 20 packet, medium – 40 packets, normal – 60 packets, low – 80 packet. The queue-limit sets in this way because high queue can be scheduled first and need less time to de-queue, the queue size does not need to be large. Also, it brings a balance for others queue scheduling. The queue-limit can be change with command priority-list 1 queue-limit x x x x.

```
R1#sh queueing priority
Current DLCI priority queue configuration:
Current priority queue configuration:

List   Queue  Args
1      high   protocol ip          list 101
1      medium protocol ip          list 102
1      normal protocol ip          list 103
1      low    protocol ip          list 104
1      high   limit 10
1      medium limit 20
1      normal limit 30
1      low    limit 40
R1#
```

Figure 4-8

3rd assign the priority list to interface s2/0

```
interface Serial2/0
 ip address 1.1.1.1 255.255.255.0
 priority-group 1
 serial restart-delay 0
!
```

Figure 4-9

Show interface s2/0 command to see the PQ applied and whether it takes the effect.

```
R1#sh interfaces s2/0
Serial2/0 is up, line protocol is up
  Hardware is M4T
  Internet address is 1.1.1.1/24
  MTU 1500 bytes, BW 1544 Kbit/sec, DLY 20000 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation HDLC, crc 16, loopback not set
  Keepalive set (10 sec)
  Restart-Delay is 0 secs
  Last input 00:00:07, output 00:00:06, output hang never
  Last clearing of "show interface" counters never
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 118
  Queueing strategy: priority-list 1
  Output queue (queue priority: size/max/drops):
     high: 0/20/94, medium: 0/40/24, normal: 0/60/0, low: 0/80/0
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     7372 packets input, 351470 bytes, 0 no buffer
     Received 229 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     12749 packets output, 18609878 bytes, 0 underruns
     0 output errors, 0 collisions, 3 interface resets
     0 unknown protocol drops
     0 output buffer failures, 0 output buffers swapped out
     4 carrier transitions     DCD=up  DSR=up  DTR=up  RTS=up  CTS=up
```

Figure 4-10

Now we are using iperf which is a tool for measuring the achievable maximum bandwidth on IP network.

PC1 is iperf client and PC2 acts like a server. Let us see the output of the execution. We run iperf3 command concurrently, so we can see the bandwidth occupation.

```
C:\Users\wei\Desktop\iperf-3.0.11-win32>iperf3 -c 192.168.2.1 -t 60 -P 1 -p 5001

Connecting to host 192.168.2.1, port 5001
[  4] local 192.168.1.1 port 49383 connected to 192.168.2.1 port 5001
[ ID] Interval           Transfer     Bandwidth
[  4]   0.00-60.01  sec  3.32 MBytes   464 Kbits/sec                  sender
[  4]   0.00-60.01  sec  3.27 MBytes   457 Kbits/sec                  receiver
```

Figure 4-11

```
C:\Users\wei\Desktop\iperf-3.0.11-win32>iperf3 -c 192.168.2.1 -t 60 -P 1 -p 5002

Connecting to host 192.168.2.1, port 5002
[  4] local 192.168.1.1 port 49386 connected to 192.168.2.1 port 5002
[ ID] Interval           Transfer     Bandwidth
[  4]   0.00-60.01  sec  2.71 MBytes   378 Kbits/sec                  sender
[  4]   0.00-60.01  sec  2.65 MBytes   370 Kbits/sec                  receiver
```

Figure 4-12

```
C:\Users\wei\Desktop\iperf-3.0.11-win32>iperf3 -c 192.168.2.1 -t 60 -P 1 -p 5003

Connecting to host 192.168.2.1, port 5003
[  4] local 192.168.1.1 port 49387 connected to 192.168.2.1 port 5003
[ ID] Interval           Transfer     Bandwidth
[  4]   0.00-60.01  sec  2.58 MBytes   361 Kbits/sec                  sender
[  4]   0.00-60.01  sec  2.54 MBytes   356 Kbits/sec                  receiver
```

Figure 4-13

As the expectation, traffic 1 as high priority queue took more bandwidth more others and got addressed fisrt. The router always process the high queue most if there are packets in queue.

4.3 Weighted Fair queueing

Prior to WFQ, there is a brief introduction of FQ. Fair Queueing is based on the concept of traffic flow to split the bandwidth to assign for each queue and utilize Round Robin algorithm which are against the conventional approach FIFO with one single queue for all traffic flows or PQ without fairly scheduling packets from each queue. The objective of FQ is to implement fairness when a limited link being shared. In cisco routers or switches, the interfaces speed less than 2.048 Mbps set to the default as Fair Queueing; otherwise, FIFO is default applied to all interfaces.

WFQ is a further development of FQ, in order to break the fairness and assign special treatment for particular queue, so the mechanism acts as classifying the traffic into different queue and deciding how much resource needs to be allocated proportionately to each queue regarding other queues. Furthermore, WFQ performs a queueing algorithm to reduce response time for interactive flows by scheduling them to the front of the queue and fairly share the bandwidth among flows.

WFQ classifies the traffic flows by the characteristic of the packet including following parameters: source/destination IP address or MAC address, source/destination TCP or UDP port, IPP/DSCP/EXP value, and protocols, and the packets of the same flow end up in the same queue. For example, the packets having same above parameters will be put into same flow such as a series of HTTP data packets or ICMP ping packets. WFQ can configure N queues up to 256 and assign a weight value for each of flow. The weight determines how bandwidth can be guaranteed. WFQ allocates the bandwidth regarding the IPP/DSCP/EXP to each flow, so this reflects the weight assigned in the queue. For example, there are N flows with respective weight w1, w2, w3, w4, w5…, wn, so the assigned bandwidth proportion of the K flow is (w1+w2+w3+…+wn)/wk. (Note: bandwidth proportion is inversely proportion to weight, the greater weight the less bandwidth allocation). In IOS, using IPP to address the weight. The proportion of the bandwidth assigned to each flow is (IPP+1)/(sumIPP+1). In addition, if two or more flows put into one queue, it might result to lower the bandwidth which is supposed to assign to one queue. Therefore, the importance is that the number of queues configured has to be larger than the number of flows. The packet put into appropriate queue according to packet characteristics calculated as an index of the queue known as hash value by source/destination IP address or MAC address, source/destination TCP or UDP port, IPP/DSCP/EXP value, and protocols, and assign a weight=32384/(IPP+1) value for finish time calculation to determine the scheduling of packet in one flow queue. $FT(P_{k+1})=FT(P_k)+Size(P_{k+1})*32384/(IPP+1)$ (k=0,1,2,3,4…,7), the packet has minimum FT comparing with other queues will be scheduling first to be sent out of software queue and the rest can be done in the same manner regarding FT value. According to the integer issue, that's the reason why we multiple a number 32384 which is least common multiple of 1 to 7. Also there is another thing dropping algorithm, and WFQ has two types of drooping which are early dropping (congestion

discard threshold) and aggressive dropping (hold-queue out limit). HQO is the maximum packets that the WFQ buffer can hold, and CDT is the threshold when WFQ calculate the worst FT packet and start to drop packets early. The dropping policy as illustration shows:

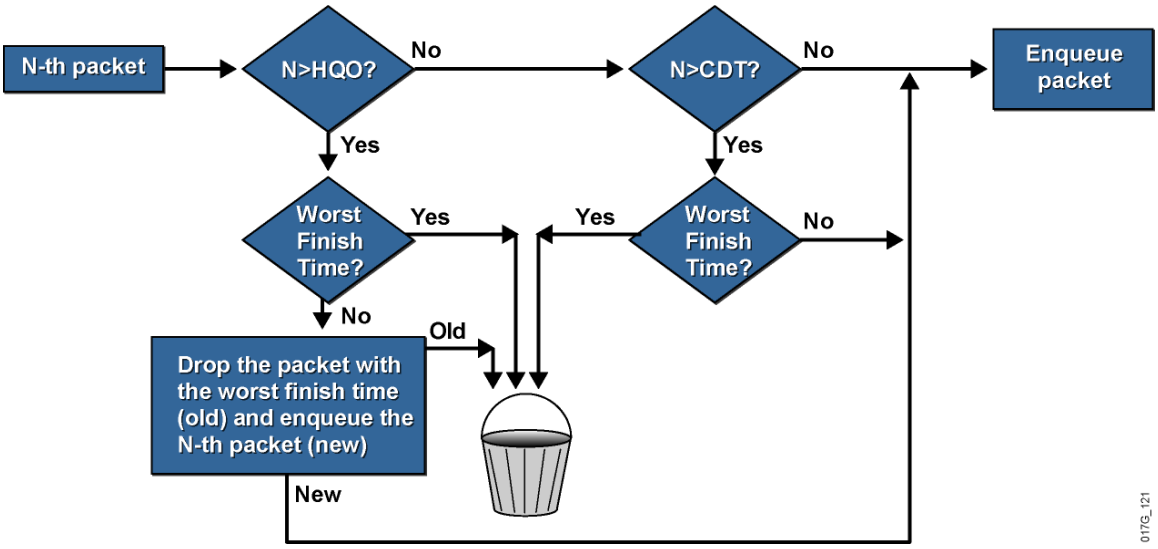


Figure 4-14

http://blog.globalknowledge.com/technology/unified-communications/quality-of-service-part-10-%E2%80%93-weighted-fair-queuing/

Note: Because dropping happens in the same flow, the packet precedence has no effect on the dropping policy. In one flow contains various packets; $FT(P_{k+1})=FT(P_k)+Size(P_{k+1})$, $FT(P_0)=Now+size(P_0)$ (k=0,1,2,3,4,….k)

Here is the illustration of WFQ:

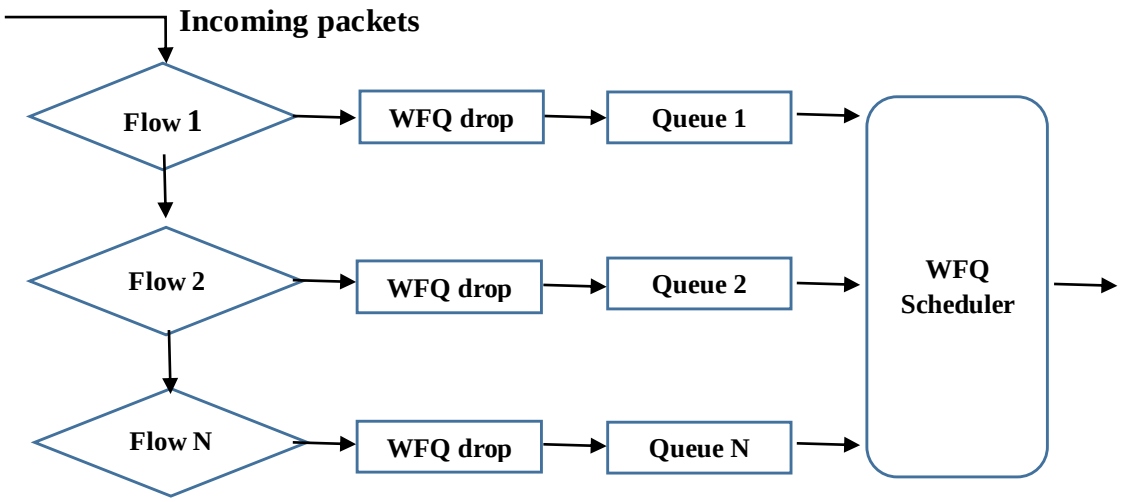


Figure 4-15

http://virtualrack.blogspot.ca/2011/11/notes-wfq.html

To explore the implementation on cisco devices and test WFQ algorithm, as below with topology and configuration:

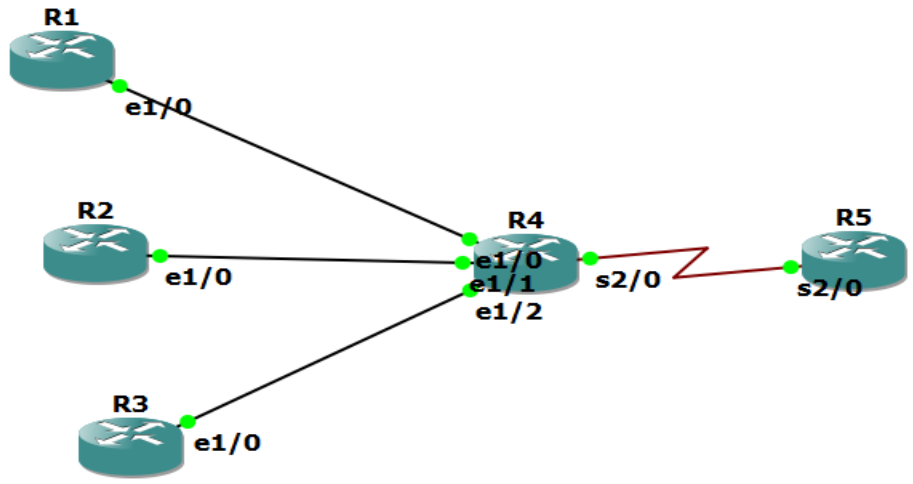


Figure 4-16

- Creating loopback0 for on R5 5.5.5.5/32
- make sure all can ping each other
- Implementing WFQ on R4 s2/0, set default IPP of the traffic from R1, R2, R3 respectively to 0, 1, 3.

```
R1#sh policy-map
  Policy Map MARK
    Class class-default
      set ip precedence 0
```

```
R2#sh policy-map
  Policy Map MARK
    Class class-default
      set ip precedence 1
```

```
R3#sh policy-map
  Policy Map MARK
    Class class-default
      set ip precedence 3
```

Figure 4-17

- Ping respectively from R2, R3, R4 to loopback R5

  Ping 5.5.5.5 timeout 0 repeat 100000000

- Show queue on R4 s2/0 to monitor WFQ:

```
R4#sh queue s2/0
This command has been deprecated. Queueing implementaion has changed - the semantics are meaningless
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 397551
  Queueing strategy: weighted fair
  Output queue: 68/1000/64/397551 (size/max total/threshold/drops)
     Conversations  3/4/256 (active/max active/max total)
     Reserved Conversations 0/0 (allocated/max allocated)
     Available Bandwidth 1158 kilobits/sec

  (depth/weight/total drops/no-buffer drops/interleaves) 20/16192/123189/0/0
  Conversation 95, linktype: ip, length: 104
  source: 24.24.24.2, destination: 5.5.5.5, id: 0x2966, ttl: 254, prot: 1

  (depth/weight/total drops/no-buffer drops/interleaves) 38/8096/108899/0/0
  Conversation 126, linktype: ip, length: 104
  source: 34.34.34.3, destination: 5.5.5.5, id: 0xF1CA, ttl: 254, prot: 1

  (depth/weight/total drops/no-buffer drops/interleaves) 10/32384/165471/0/0
  Conversation 64, linktype: ip, length: 104
  source: 14.14.14.1, destination: 5.5.5.5, id: 0xA1EA, ttl: 254, prot: 1
```

Figure 4-18

As we known, the weight=32384/(IPP+1), so for flow 1(IPP=0)=8086, flow 2(IPP=1)=16192, flow 3(IPP=3)= 8096. As above mentioned, the bandwidth allocated is inversely proportion to weight. The greater weight the less bandwidth.

Show int s2/0 is running WFQ:

```
R4#sh int s2/0
Serial2/0 is up, line protocol is up
  Hardware is M8T-X.21
  Internet address is 45.45.45.4/24
  MTU 1500 bytes, BW 1544 Kbit/sec, DLY 20000 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation HDLC, crc 16, loopback not set
  Keepalive set (10 sec)
  Restart-Delay is 0 secs
  Last input 00:00:05, output 00:00:05, output hang never
  Last clearing of "show interface" counters never
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 1169739
  Queueing strategy: weighted fair
  Output queue: 0/1000/64/1169739 (size/max total/threshold/drops)
     Conversations  0/4/256 (active/max active/max total)
     Reserved Conversations 0/0 (allocated/max allocated)
     Available Bandwidth 1158 kilobits/sec
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     84985 packets input, 8829165 bytes, 0 no buffer
     Received 298 broadcasts (0 IP multicasts)
     0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     86577 packets output, 8995052 bytes, 0 underruns
     0 output errors, 0 collisions, 2 interface resets
  --More--
```

Figure 4-19

- Create a policy-map to check the occupied bandwidth for each queue on R4

```
R5#show class-map
 Class Map match-any class-default (id 0)
   Match any

 Class Map match-all IPP1 (id 2)
   Match ip  precedence 1

 Class Map match-all IPP0 (id 1)
   Match ip  precedence 0

 Class Map match-all IPP3 (id 3)
   Match ip  precedence 3

R5#sh pol
R5#sh policy-map
  Policy Map METER
    Class IPP0
    Class IPP1
    Class IPP3
```

Figure 4-20

- Set the load-interval 30 on interface S2/0 on R5 and after 30 seconds Show policy-map int s2/0 on R5:

```
      Match: any
R5# sh policy-map interface s2/0
 Serial2/0

  Service-policy input: METER

    Class-map: IPP0 (match-all)
      14704 packets, 1529216 bytes
      30 second offered rate 21000 bps
      Match: ip precedence 0

    Class-map: IPP1 (match-all)
      18453 packets, 1919112 bytes
      30 second offered rate 42000 bps
      Match: ip precedence 1

    Class-map: IPP3 (match-all)
      35586 packets, 3700944 bytes
      30 second offered rate 81000 bps
      Match: ip precedence 3

    Class-map: class-default (match-any)
      0 packets, 0 bytes
      30 second offered rate 0 bps, drop rate 0 bps
      Match: any
R5#
```

Figure 4-21

The proportion of occupied bandwidth=(IPP+1)/(sumIPP+1), so the allocation ratio is 1:2:4 as same as the test on R5.

Therefore, this small lab tested the function of WFQ and its algorithm by result as above.

4.4 Class-base Weighted Fair Queueing

As previous we talked about WFQ, CBWFQ is an extension of WFQ which gives more customised user-defined requests and create classes for various traffic classification to satisfy and match the criteria including ACL, protocol/NBAR, and so on. The WFQ classified the traffic flows based on the characteristics of the packets, and this mechanism is working automatically. The bandwidth allocation also is done automatically depending on weight of the WFQ versus CBWFQ specifying weight manually. Since the class has been created within matched criteria, the packets can be certain class and then put into the relative queue. Therefore, it guarantees the certain traffic performance and bandwidth delivery to each class when the congestion occurs. When the packets have reached the limited queue size (queue-limit), it will start to drop the overload packets. The default congestion avoidance mechanism is tail drop, but the queue also can be configured with WRED or CBWRED. Packets in classes, get services in a round-robin fashion based on their finish time (FT). It is the same as WFQ scheduling algorithm, and the packets de-queues depend on the assigned weight. The packets with lower finish time gets high priority then other, which the packets will be de-queued first towards hardware queue. Hence, it provides more control and flexibility of the resource allocation.

The operational process illustrates in CBWFQ:

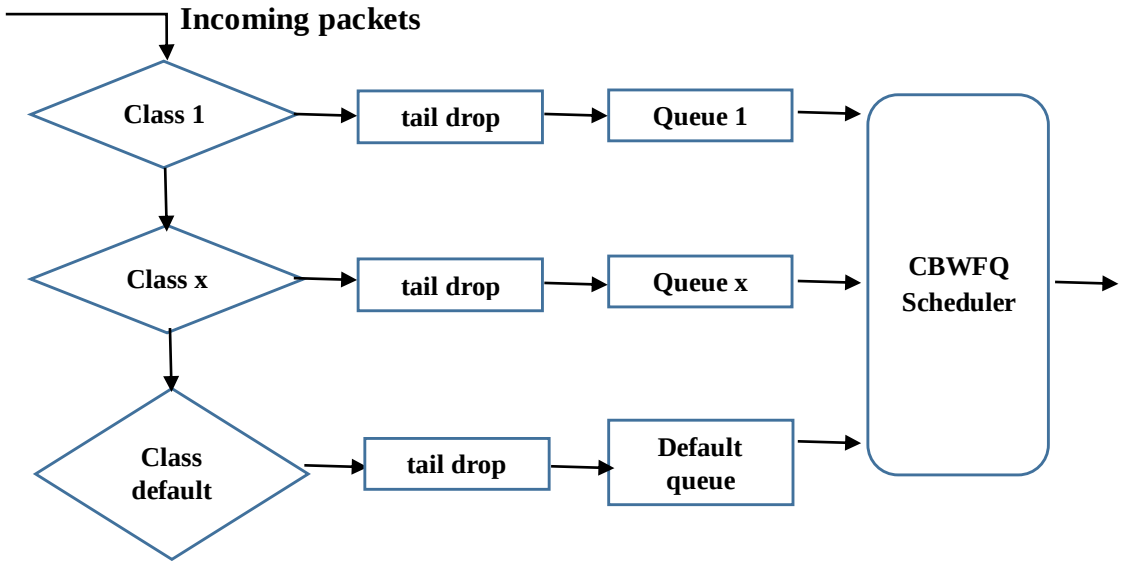


Figure 4-22

http://lh3.ggpht.com/-gOaonvekK0U/TrK2vDx47BI/AAAAAAAAANs/v0G0oU-Ha70/s1600-h/image%25255B4%25255D.png

A simple example of configuring CB-WFQ:

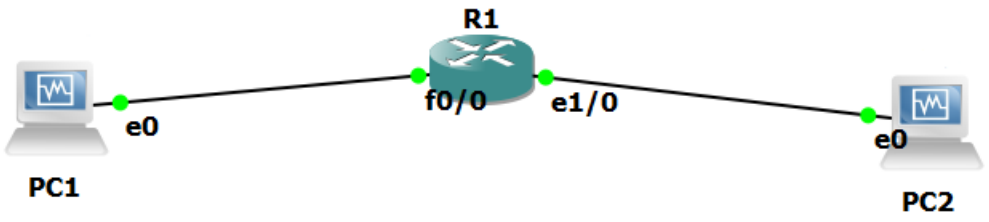


Figure 4-23

On R1, f0/0 100Mbps and e1/0 10Mbps, if too much traffic coming at the same time from PC1 to PC2, it will cause congestion on e1/0. Thus, to filter the more important traffic, we are going to classify them into different classes and give different minimum bandwidth guarantee. Sending rate of IPP1 traffic: 5Mbps, sending rate of IPP2 traffic: 5Mbps, sending rate of IPP2 traffic: 5Mbps, Sending rate of EF traffic: 20Mbps, The basic configuration as:

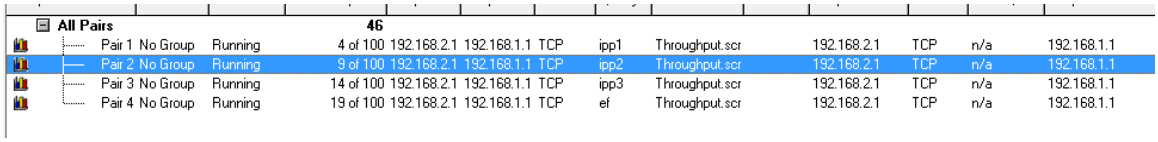
```
R1#show class-map
 Class Map match-any class-default (id 0)
   Match any

 Class Map match-all p1 (id 1)
   Match ip  precedence 1

 Class Map match-all p2 (id 2)
   Match ip  precedence 2

 Class Map match-all p3 (id 3)
   Match ip  precedence 3

R1#sh policy-map
  Policy Map CB-WFQ
    Class p1
      bandwidth 1000 (kbps)
    Class p2
      bandwidth 2000 (kbps)
    Class p3
      bandwidth 3000 (kbps)
```

Figure 4-24

```
R1#sh policy-map interface
 Ethernet1/1

  Service-policy output: CB-WFQ

    Class-map: p1 (match-all)
      17145 packets, 1187366 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: ip precedence 1
      Queueing
      queue limit 64 packets
      (queue depth/total drops/no-buffer drops) 0/0/0
      (pkts output/bytes output) 17145/1187366
      bandwidth 1000 kbps

    Class-map: p2 (match-all)
      15880 packets, 1102401 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: ip precedence 2
      Queueing
      queue limit 64 packets
      (queue depth/total drops/no-buffer drops) 0/0/0
      (pkts output/bytes output) 15880/1102401
      bandwidth 2000 kbps

    Class-map: p3 (match-all)
      18236 packets, 1256534 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: ip precedence 3
      Queueing
      queue limit 64 packets
      (queue depth/total drops/no-buffer drops) 0/0/0
      (pkts output/bytes output) 18236/1256534
      bandwidth 3000 kbps

    Class-map: class-default (match-any)
      17497 packets, 1213664 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: any
```

Figure 4-25

The policy-map ensures the minimum bandwidth for IPP1, IPP2, and IPP3.



All Pairs			46									
Pair 1	No Group	Running	4 of 100	192.168.2.1	192.168.1.1	TCP	ipp1	Throughput.scr	192.168.2.1	TCP	n/a	192.168.1.1
Pair 2	No Group	Running	9 of 100	192.168.2.1	192.168.1.1	TCP	ipp2	Throughput.scr	192.168.2.1	TCP	n/a	192.168.1.1
Pair 3	No Group	Running	14 of 100	192.168.2.1	192.168.1.1	TCP	ipp3	Throughput.scr	192.168.2.1	TCP	n/a	192.168.1.1
Pair 4	No Group	Running	19 of 100	192.168.2.1	192.168.1.1	TCP	ef	Throughput.scr	192.168.2.1	TCP	n/a	192.168.1.1

Figure 4-26-a

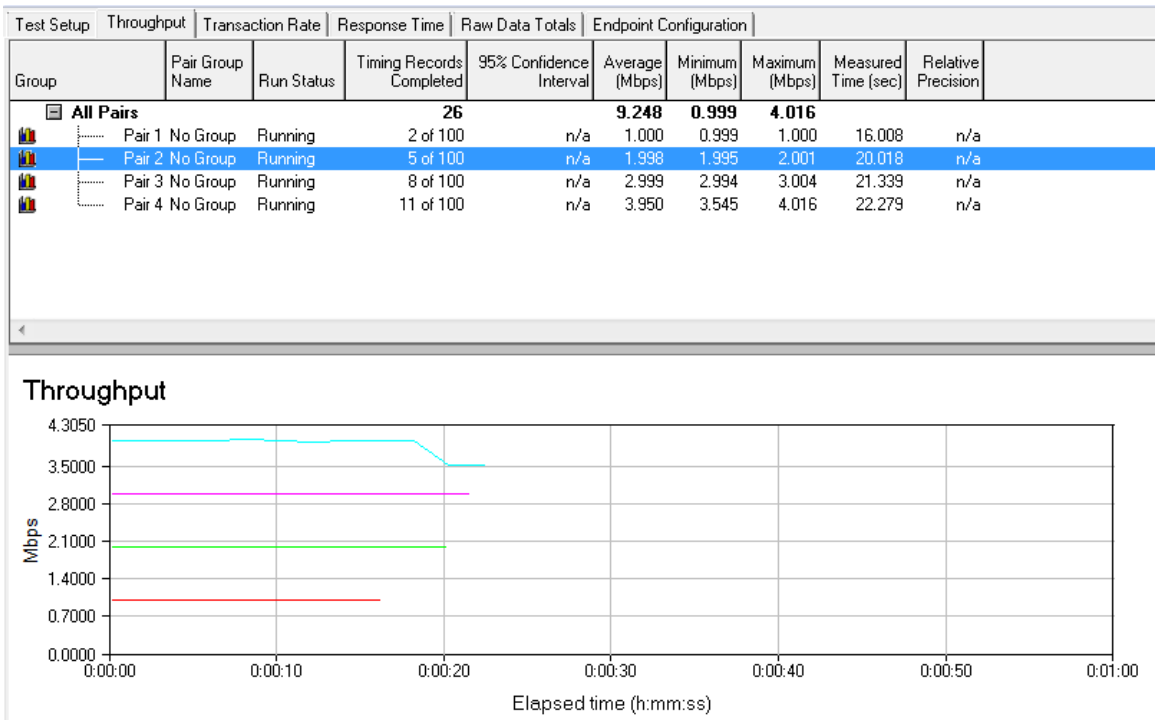

Test Setup
Throughput
Transaction Rate
Response Time
Raw Data Totals
Endpoint Configuration
Group
Pair Group Name
Run Status
Timing Records Completed
95% Confidence Interval
Average (Mbps)
Minimum (Mbps)
Maximum (Mbps)
Measured Time (sec)
Relative Precision
All Pairs 26 9.248 0.999 4.016
Pair 1 No Group Running 2 of 100 n/a 1.000 0.999 1.000 16.008 n/a
Pair 2 No Group Running 5 of 100 n/a 1.998 1.995 2.001 20.018 n/a
Pair 3 No Group Running 8 of 100 n/a 2.999 2.994 3.004 21.339 n/a
Pair 4 No Group Running 11 of 100 n/a 3.950 3.545 4.016 22.279 n/a
Throughput
4.3050
3.5000
2.8000
2.1000
1.4000
0.7000
0.0000
Mbps
0:00:00
0:00:10
0:00:20
0:00:30
0:00:40
0:00:50
0:01:00
Elapsed time (h:mm:ss)

Figure 4-26-b

4.5 Low-latency Queueing

LLQ is an upgrade of CBWFQ which based on CBWFQ within purpose providing a guaranteed priority queueing for time-sensitive application. It can prioritize the selected flow to be scheduled first and reduce delay and jitter, actually it is a combination with CBWFQ which adds a strict PQ. The strict PQ ensures a minimum bandwidth to a particular class when the congestion occurs. The packets will be dropped if the queue get stuffed. In CBWFQ, all packets are treated fairly according to weight, there is no such strict priority specified. For example, the time-sensitive traffic such as voice traffic has no tolerance of delay, and especially variation in delay. Therefore, to fix this issue, LLQ enables a single strict priority queue with guaranteed bandwidth.

The illustration shows LLQ architecture as below:

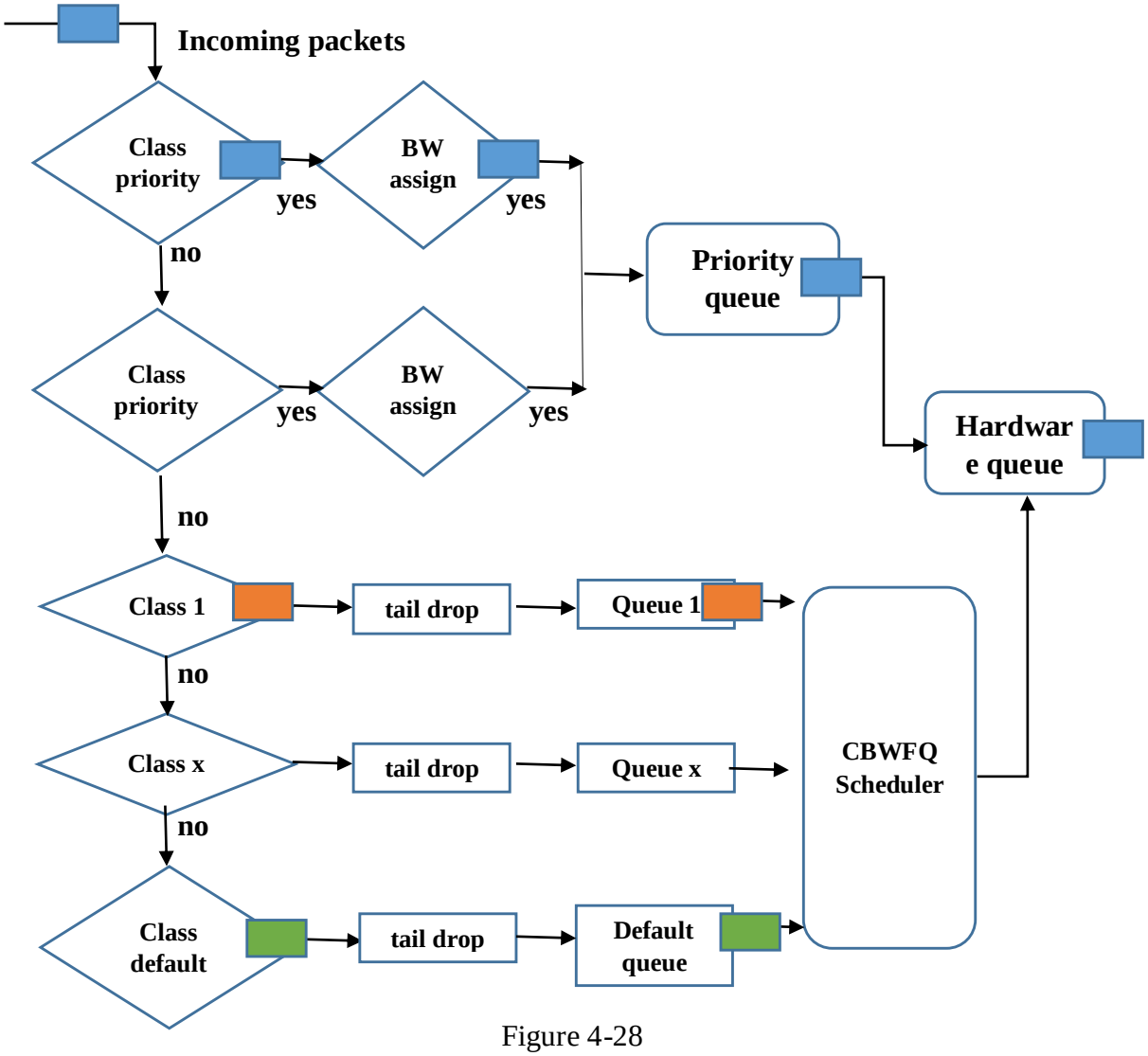


Figure 4-28

http://core0.staticworld.net/images/idge/imported/article/nww/2008/03/07fig05-100279669-orig.jpg

Voice packet

Class 1 packet

Class default packet

The packets put into their own class and Voice packets always are scheduled first.

CB-WFQ and LLQ both can guarantee minimum bandwidth, Priority queue introduced in LLQ which provides low latency and applies to real-time applications.

4.6 Queueing mechanism comparison

As above queueing mechanisms, they made a breakthrough to traditional IP network of simple FIFO congestion management policy, and empowered QoS capacity to fulfil different service requirements. Here is the comparison of these queueing methods.

type of queue	Number of queue	Advantage	Disadvantage
FIFO	**1**	**Default queueing, easy to use and configure Simple queueing method**	**All packets treated the same. The arrival sequence of packets determines the bandwidth distribution, packet delay and lost. No guarantee to time-sensitive traffic**
PQ	**4**	**Guarantee bandwidth to real-time applications, the packets can be scheduled first**	**If no limit to PQ, Lower priority queues get starving**
WFQ	**User-defined**	**Easy to configure, flow classification automatically completed Reduce delay jitter Allocate different bandwidth based on priority level Since number of flows decrease, bandwidth allocation will increase automatically for rest of flows**	**Not applicable to time-sensitive application**

CBWFQ	**User-defined**	**Classify traffic more flexible to satisfy different requirements Provide a fair queueing by weight More efficient for scheduling**	**When configured too many classes of traffic, the hardware usage will dramatically increase.**
LLQ	**User-defined**	**Ensure the important traffic can be transmitted first. Other traffic still follows WFQ scheduling mechanism**	**More complicated to configure, it requires a good understanding of PQ and WFQ. Bad configuration might get packets dropped or starving**

Table 4-29

http://h3c.com/portal/Products___Solutions/Products/Switches/H3C_S5120-EI_Series_Switches/White_Paper/200812/689004_57_0.htm#_Toc215923183

5. Congestion avoidance

Excess congestion will cause great harm to the network resource, there must be some approaches to solve this issue. Congestion avoidance is a mechanism which control and administrate traffic flows in accordance with monitoring the usage of network resource such as buffer usage. When the congestion is dramatically getting critical, the buffer will discard overload packets and adjust amount of traffic flows to relieve network overload.

Comparing with End-to-End flow control, using dropping has more significant meaning for device and manage more classes of traffic without affecting usage of resource of entire link. It can take the effect to each node and won't affect the original flow control such as TCP control flow while the device drops packets. Thus, the combination of queueing and packets dropping can optimize the throughput and maximize efficiency of the network, which also minimize the packet loss and delay.

5.1 Tail-drop

The default of dropping approach is tail-drop, it's similar concept as FIFO which strictly follow the order of the incoming and outgoing. Tail-drop is traditional dropping policy, which will drop incoming packet when the queue reach maximum threshold and cannot buffer more packets. Dropped packets may cause significant application performance degradation.

As we known, for TCP transmission, it needs three-way handshake to establish connection first. This dropping approach can cause TCP synchronization when the

TCP connection packets cannot be buffer and be discarded. The first we have to look at is TCP slow start which is the cause of TCP synchronization. So what the TCP slow start is and why TCO slow start should be apply to. In WAN, TCP packet might go through many routers and low speed link. If the sender starts to send numbers of packets at the same time, which causes increasing the buffer load dramatically among the intermediate routers, so the buffer will be overloaded and then drop incoming packets. Hypothetically, the intermediate routers can buffer 3 packets, and the sender constantly send 5 packets as 1,2,3,4,5. The router might not forward all of them at one time, so the first two packets get forwarded first and 3,4,5 packets are buffered in the queue which means the buffer has no space for incoming packets. Meanwhile, the sender starts to send new packets towards intermediate routers, and 6,7 cannot be buffer and get dropped. As a result, the sender cannot receive ACK packet during the certain time, so the retransmission will occur inevitably. Therefore, to solve this issue, TCP needs to support slow start algorithm. The slow start algorithm in the sender side includes a congestion widow which cooperate with actual window size. The receiver receives the initial packet and send out the ACK packet with its own window size. If there is no response from receiver, the sender either resends a packet or knows not to continue sending data any more. Since TCP connection has been successfully established, the sender increases congestion window gradually when it receives ACK packets. It will stop increasing congestion window size in next packet until the congestion window=actual window size or receiver's window size reaches to limit. The sending rate will increase again if the congestion is no longer there. The intermediate router starts to drop packets, which means sender congestion window is too big. The new packet sending packet rate should be equal with returning ACK packet rate. The procedure that many TCP flows concurrently enable TCP slow start called TCP synchronization. When TCP synchronization occurs, the bandwidth cannot be efficiently used, which causes a waste of bandwidth. As generalizing of network, the traffic has been dramatically increasing constantly, so the network congestion must occur. How to avoid congestion becomes very important, and tail-drop drops overloaded packets. This approach might cause TCP synchronization. Usually, the traffic is bursty. If the buffers are full, it will cause packet dropped in short time, and TCP flow has adaptiveness. The window size and sending packet rate will decrease rapidly in order to relieve the network congestion. The sending packet rate will get back to increase while the congestion no longer occurs. Back and forth, the network utilization becomes pretty low during that time, and also lower the throughput and increase jitter. Statistically, the packet loss of the network layer is 5%, which will decrease 50% of the throughput of application based on TCP in application layer. Here the utilization variation in TCP synchronization:

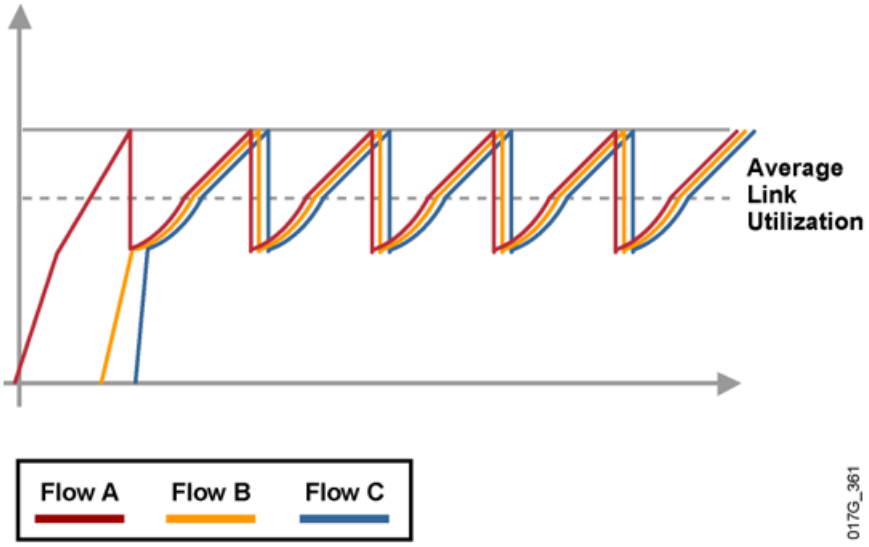


Figure 5-1

http://slideplayer.com/slide/9996714/#

In order to avoid this issue, other queue size management algorithm will reduce TCP synchronization, as well as keeping the queue efficiently being utilized. Random Early Detection (RED) and Weighted RED will introduce next.

Tail-drop also cause unfairly TCP flow bandwidth allocation. Some of aggressive flow occupy more bandwidth than others, other might starve. Especially, when UDP traffic mix with TCP. TCP slow start makes the sending rate go down to release some of bandwidth; meanwhile, UDP traffic will take over these resource rapidly. Eventually, TCP flow starves because of lack of bandwidth.

5.2 Random Early Detection

Tail-drop would cause TCP synchronization if congestion occurs. As we know from above, TCP synchronization brings a low bandwidth utilization. However, Tail-drop can be avoided prior to congestion taking place. RED is mechanism that randomly drops packets before a queue getting filled to overcome TCP synchronization. RED algorithm aims to use an average queue length/size to determine the incoming packets whether violate the congestion avoidance mechanism and drop them or not. It increases the possibility of avoiding congestion. The basic idea is to monitor the average queue size, once the RED detect the queue is approaching to be filled, it starts to randomly drop packets and decrease congestion window size in order to avoid TCP synchronization occurring. Because RED is based on FIFO queueing scheduling and only discards incoming packets, it is easy to implement.

According to two thresholds (minimum and maximum threshold) of average queue size to determine whether the incoming packets should be discarded. Minimum threshold means the number of packets prior to the queue is prepared to discard

packets. if the number of packets is less then minimum threshold, which means no congestion occurs and the packets can be buffer. When the average queue size is greater than minimum threshold, the queue runs random dropping until it exceed the maximum threshold, and then tail drops all the incoming packets. RED algorithm uses average queue size and minimum, maximum threshold to determine drop probability. The average queue size= (previous average queue size * $(1-2^{-n})$) + (current queue size * 2^{-n}). In addition, the bursty traffic has less sensitive for the change of average queue size, so it would avoid to treat bursty traffic unfairly. The comparison of RED enabled before and after:

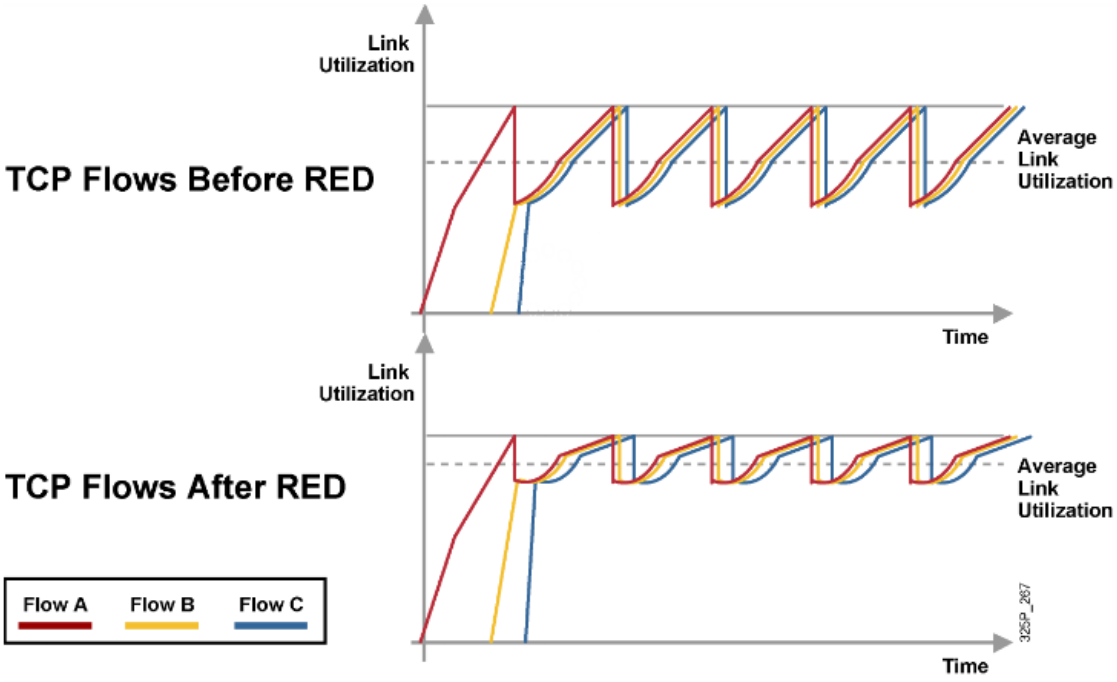


Figure 5-2

http://slideplayer.com/slide/9996714/#

Average link utilization is much closer to link bandwidth.

Although RED can avoid the congestion and boost the average link utilization, there are still some flaws:

- For the connection cannot response the congestion avoidance, RED is not able to address this kind of traffic. As the result, the bandwidth will be occupied most by this connection, it causes unfair sharing bandwidth.
- The average queue size always changes with the increase of connections; it leads to unstable network performance. RED increases drop rate as the average queue size increases.
- RED algorithm is very sensitive to parameters such as two thresholds and maximum drop probability. If one of them has changed, it will bring big effect

to network performance. For instance, the network could receive lager throughput, but might cause higher packet loss and delay.

- RED applies the random dropping, might not differentiate precedence in one class. The higher priority packets might be discarded.

5.3 Weighted Random Early Detection

WRED combines RED with IPP/DSCP to provide priority service. when the congestion occurs at interface, it not just easily randomly discards packets, but choosing lower level precedence packet to drop first. The purpose is to reduce the drop probability of higher priority packets. For example, in one queue might have serval different IP precedence packet which is associated with their own queue threshold respectively. A queue may have various thresholds, and the lower threshold is for lower priority packets, hence discarding the lower priority packets first. In this way, the higher priority packets obtain protection from other lower packets in the same queue. WRED creates its own parameter (profile) based on IPP/DSCP; each profile includes minimum threshold, maximum threshold, and maximum drop probability.

Here are figures to show the relation between IPP/DSCP and WRED profile:

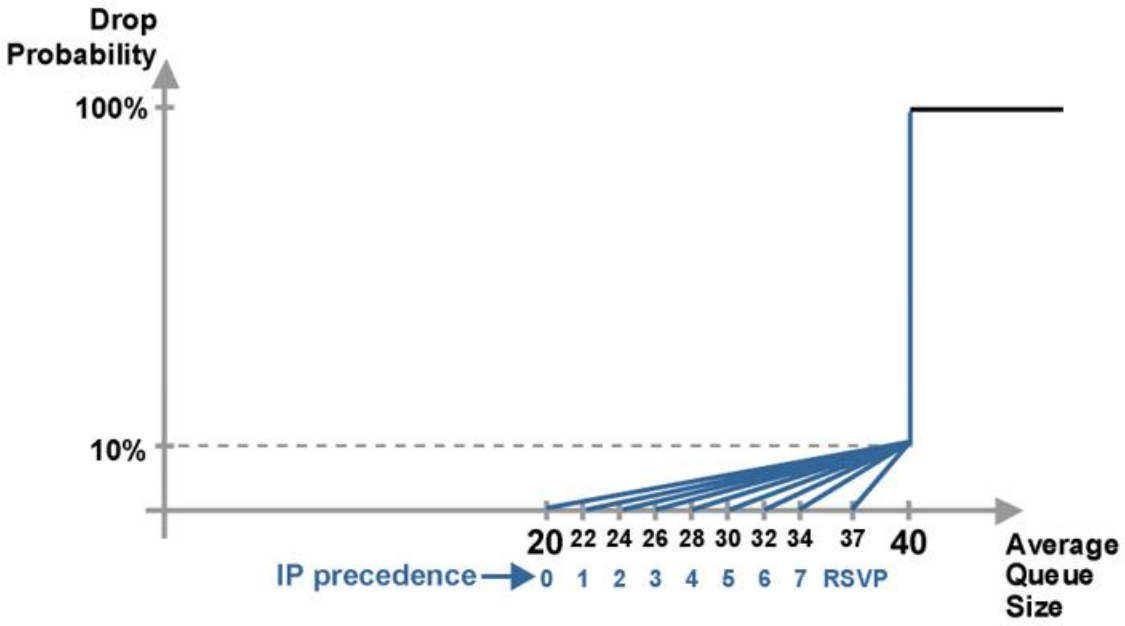


Figure 5-3

http://player.myshared.ru/9/885523/data/images/img14.jpg

From the figure we can know that the minimum threshold increasing with IPP value, the lower IPP the higher drop probability. When the IPP is the same, the lager queue

size the higher drop probability. The same thing in DSCP:

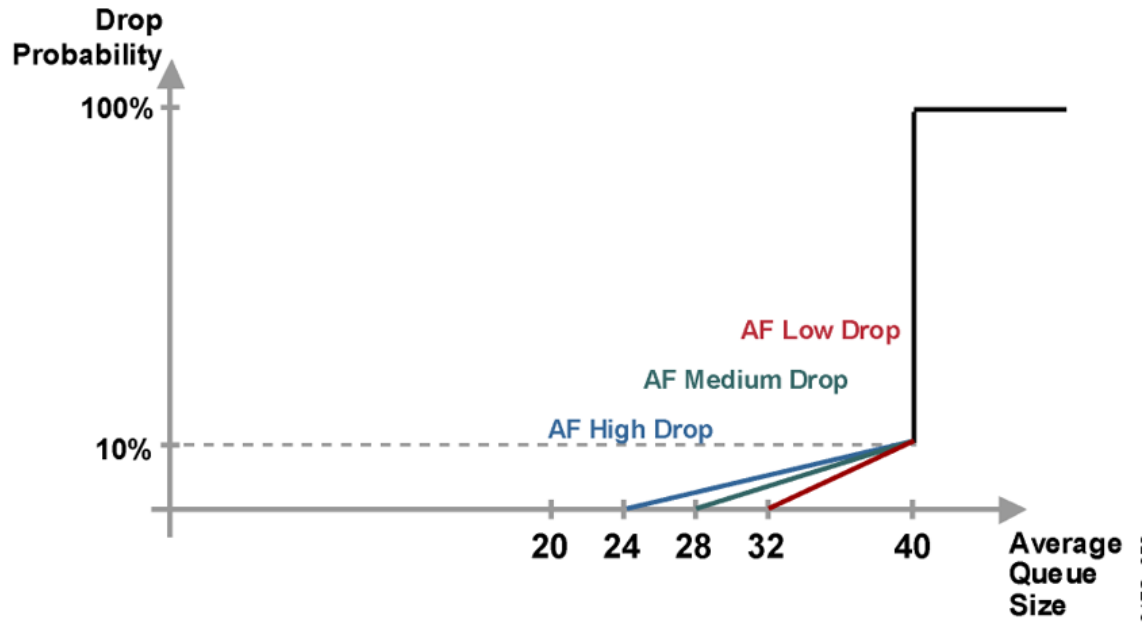


Figure 5-4

http://player.myshared.ru/9/885523/data/images/img17.jpg

WRED working procedure: once a packet arrived, first checking IPP or DSCP, according to each packet's WRED profile to determine random drop or tail drop. Illustration shown as following:

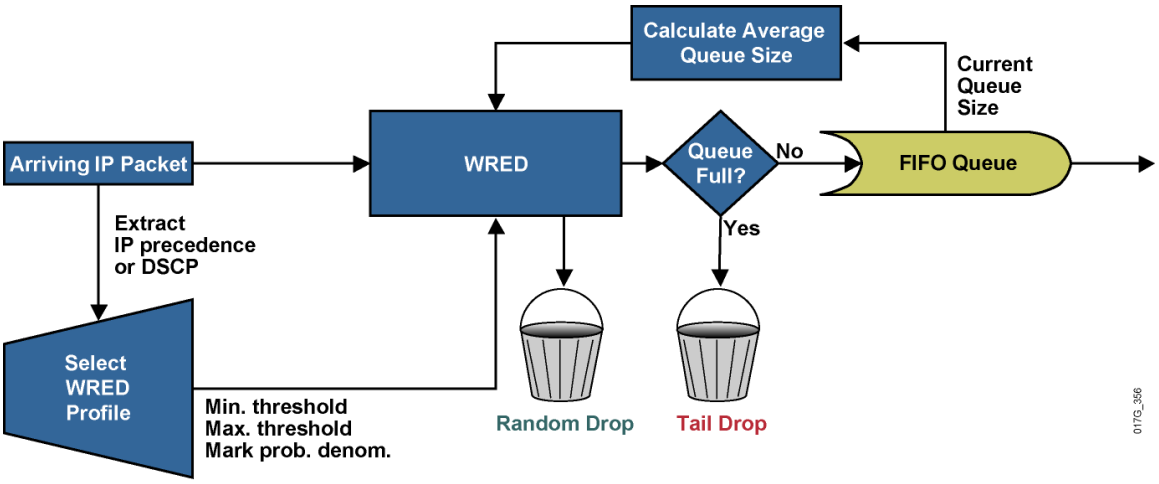


Figure 5-5

http://player.myshared.ru/9/885523/data/images/img13.jpg

In above figure, FIFO queue can be replaced by WFQ or PQ.

When queueing is WFQ:

- WRED is in accordance with different precedence to provide different drop features
- Implementing WRED based on pre-flow. Different flows have their own queues, and for small flow also has small size of queue. Therefore, the drop probability is relative low. As the result, the small flow gets protected.

When queueing is FIFO, PQ:

- WRED specifies dropping pattern based on every queue profile
- For small flows has few number of packets, so the drop probability is low, which also protect small flow traffic.

WRED usually can configured within class-based in CBWFQ or LLQ

An example of WRED discard packets depending on IPP/DSCP. There are serval traffic flows with different IPP value, and we classify them into one class. It implemented WRED with CBWFQ. The traffic transmits from PC1 to PC2, the bandwidth between PC1 and R1 is 100Mbps, and the bandwidth reduces when arriving at e1/0 with 10Mbps. Thus, the congestion would occur on interface e1/0. The topology as following:

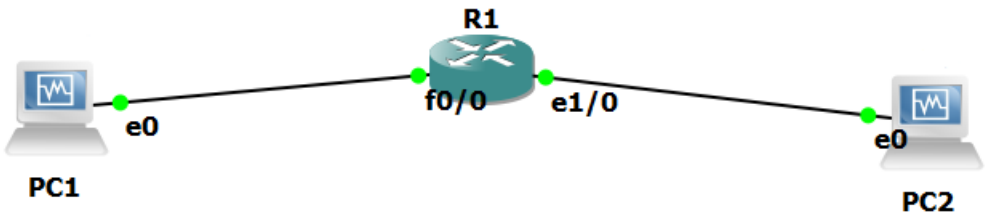


Figure 5-6

PC1: 192.168.1.1/24, PC2: 192.168.2.1/24.

Traffic: Flow1 IPP=0, bandwidth=4Mbps, PC1->PC2. Flow2 IPP=1, bandwidth=4Mbps, PC1->PC2. Flow3 IPP=2, bandwidth=4Mbps, PC1->PC2. Flow4 IPP=3, bandwidth=4Mbps, PC1->PC2. Flow5 IPP=4, bandwidth=4Mbps, PC1->PC2.

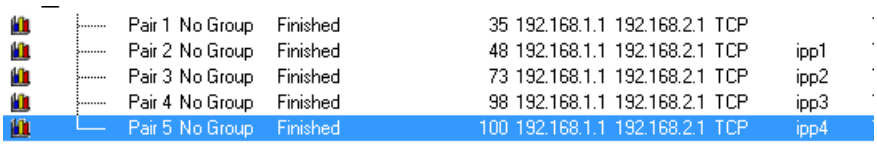


Figure 5-7

Five 4Mbps traffic send to interface e1/0 is greater than the maximum bandwidth of interface, so the congestion occurs. The queue will buffer packets and discard overloaded packets. Let us see the drop probability of each IPP flows.

1st creating access-list 101 to capture the all TCP traffic from PC1 to PC2.

```
R1(config)#access-list 101 permit tcp host 192.168.1.1 host 192.168.2.1
```

Figure 5-8

2nd enabling class-map to match this captured traffic.

```
R1#sh class-map
 Class Map match-all host (id 1)
   Match access-group  101

 Class Map match-any class-default (id 0)
   Match any
```

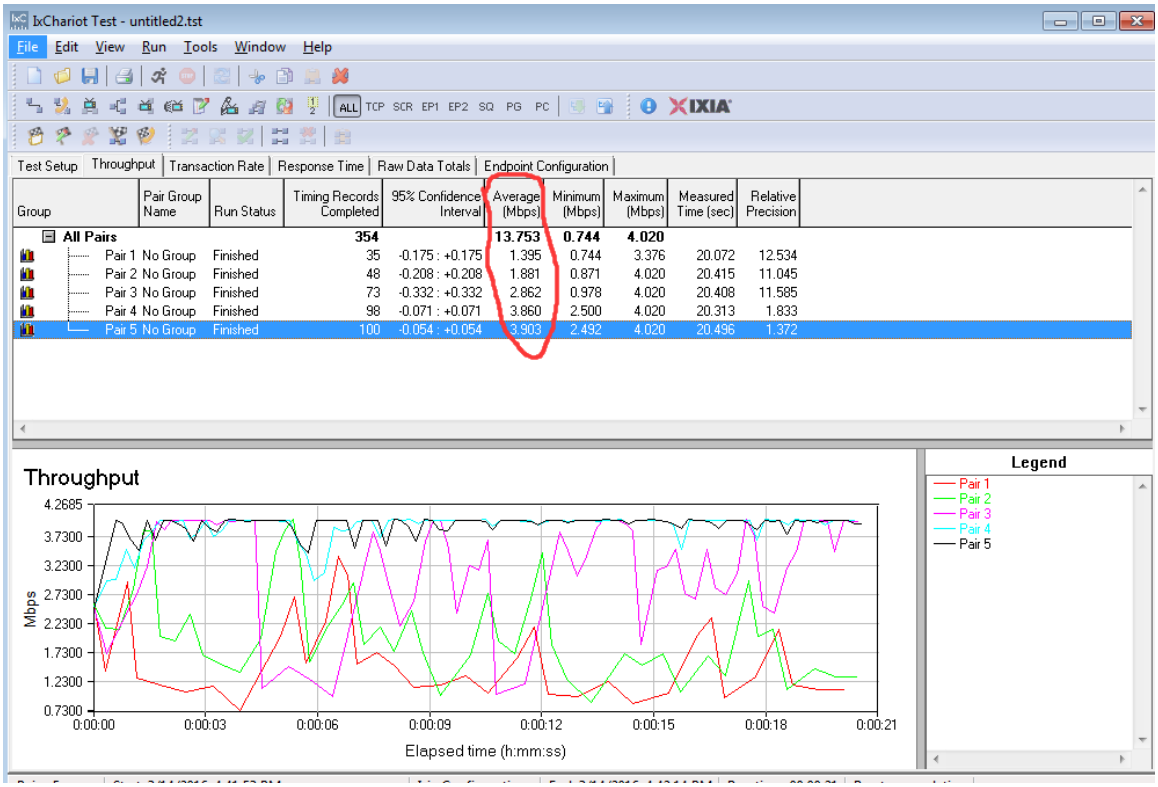
Figure 5-9

3rd invoking class-map in policy-map, enabling CBWFQ and assigning bandwidth with 5Mbps (can be any value less than 90% interface bandwidth, 10% bandwidth reserved on interface). Enabling WRED by command random-detect.

```
policy-map IPP-check
 class host
  bandwidth 5000
  random-detect
!
```

Figure 5-10

4th employing policy-map on e1/0, and starting to send traffic from PC1. (Here we are using an application called IxChariot that generates different traffic with QoS level)



Pair 1 No Group	Finished	35	192.168.1.1	192.168.2.1	TCP
Pair 2 No Group	Finished	48	192.168.1.1	192.168.2.1	TCP ipp1
Pair 3 No Group	Finished	73	192.168.1.1	192.168.2.1	TCP ipp2
Pair 4 No Group	Finished	98	192.168.1.1	192.168.2.1	TCP ipp3
Pair 5 No Group	Finished	100	192.168.1.1	192.168.2.1	TCP ipp4

Figure 5-11

According to WFQ, assign bandwidth by weight. The higher IPP, the more bandwidth usage.

Show policy interface:

```
R1#sh policy-map interface
 Ethernet1/0

  Service-policy output: IPP-check

    Class-map: host (match-all)
      25503 packets, 37549509 bytes
      5 minute offered rate 1298000 bps, drop rate 0 bps
      Match: access-group 101
      Queueing
      queue limit 64 packets
      (queue depth/total drops/no-buffer drops) 0/186/0
      (pkts output/bytes output) 25317/37271253
      bandwidth 5000 kbps
        Exp-weight-constant: 9 (1/512)
        Mean queue depth: 17 packets
        class     Transmitted       Random drop      Tail drop         Minimum       Maximum     Mark
                  pkts/bytes        pkts/bytes       pkts/bytes        thresh        thresh      prob

        0           2543/3739949      71/107494        0/0               20            40        1/10
        1           3397/5083450      61/92354         0/0               22            40        1/10
        2           5187/7681570      32/45100         0/0               24            40        1/10
        3           7089/10382818     15/22710         0/0               26            40        1/10
        4           7101/10383466      7/10598         0/0               28            40        1/10
        5              0/0               0/0           0/0               30            40        1/10
        6              0/0               0/0           0/0               32            40        1/10
        7              0/0               0/0           0/0               34            40        1/10

    Class-map: class-default (match-any)
      24 packets, 2319 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: any
```

Figure 5-12

The above statistics shows that the number of drop depends on IPP. The higher precedence, the higher drop probability.

The same thing for DSCP. Traffic: Flow1 DSCP=AF11, bandwidth=4Mbps, PC1->PC2. Flow2 DSCP=AF12, bandwidth=4Mbps, PC1->PC2. Flow3 DSCP=13, bandwidth=4Mbps, PC1->PC2. Flow4 DSCP=21, bandwidth=4Mbps, PC1->PC2. Flow5 DSCP=AF22, bandwidth=4Mbps, PC1->PC2, Flow6 DSCP=AF23, bandwidth=4Mbps, PC1->PC2.

Group	Pair Group Name	Run Status	Timing Records Completed	Endpoint 1	Endpoint 2	Network Protocol	Service Quality	Script/Stream Filename	Pair Comment	Console knows Endpoint 1	Console Protocol	Console Serv. Qual.	Endpoint 1 knows Endpoint 2
All Pairs			**374**										
Pair 1 No Group		Finished	44	192.168.1.1	192.168.2.1	TCP	af11	Throughput.scr		192.168.1.1	TCP	n/a	192.168.2.1
Pair 2 No Group		Finished	32	192.168.1.1	192.168.2.1	TCP	af12	Throughput.scr		192.168.1.1	TCP	n/a	192.168.2.1
Pair 3 No Group		Finished	28	192.168.1.1	192.168.2.1	TCP	af13	Throughput.scr		192.168.1.1	TCP	n/a	192.168.2.1
Pair 4 No Group		Finished	100	192.168.1.1	192.168.2.1	TCP	af21	Throughput.scr		192.168.1.1	TCP	n/a	192.168.2.1
Pair 5 No Group		Finished	96	192.168.1.1	192.168.2.1	TCP	af22	Throughput.scr		192.168.1.1	TCP	n/a	192.168.2.1
Pair 6 No Group		Finished	74	192.168.1.1	192.168.2.1	TCP	af23	Throughput.scr		192.168.1.1	TCP	n/a	192.168.2.1

Figure 5-13

Enabling random-detect dscp-based, and assignning mimimum and maximum threshold for each dcsp flow. And also put flows into one class by access-list.

```
R1(config)#access-list 101 permit tcp host 192.168.1.1 host 192.168.2.1
```

Figure 5-14-a

```
R1# sh class-map
 Class Map match-all af (id 3)
   Match access-group  101

 Class Map match-any class-default (id 0)
   Match any
```

Figure 5-14-b

```
policy-map 1
 class af
  bandwidth 5000
  random-detect dscp-based
  random-detect dscp 10 28 40 10
  random-detect dscp 12 26 40 10
  random-detect dscp 14 24 40 10
  random-detect dscp 18 34 40 10
  random-detect dscp 20 32 40 10
  random-detect dscp 22 30 40 10
!
!
```

Figure 5-14-c

Show policy-map:

```
R1# sh policy-map 1
  Policy Map 1
    Class af
      bandwidth 5000 (kbps)
       packet-based wred, exponential weight 9

      dscp    min-threshold    max-threshold    mark-probablity
      ------------------------------------------------------------
      af11 (10)     28               40               1/10
      af12 (12)     26               40               1/10
      af13 (14)     24               40               1/10
      af21 (18)     34               40               1/10
      af22 (20)     32               40               1/10
      af23 (22)     30               40               1/10
      default (0)   -                -                1/10
R1#
```

Figure 5-15

Starting to send traffic from PC1

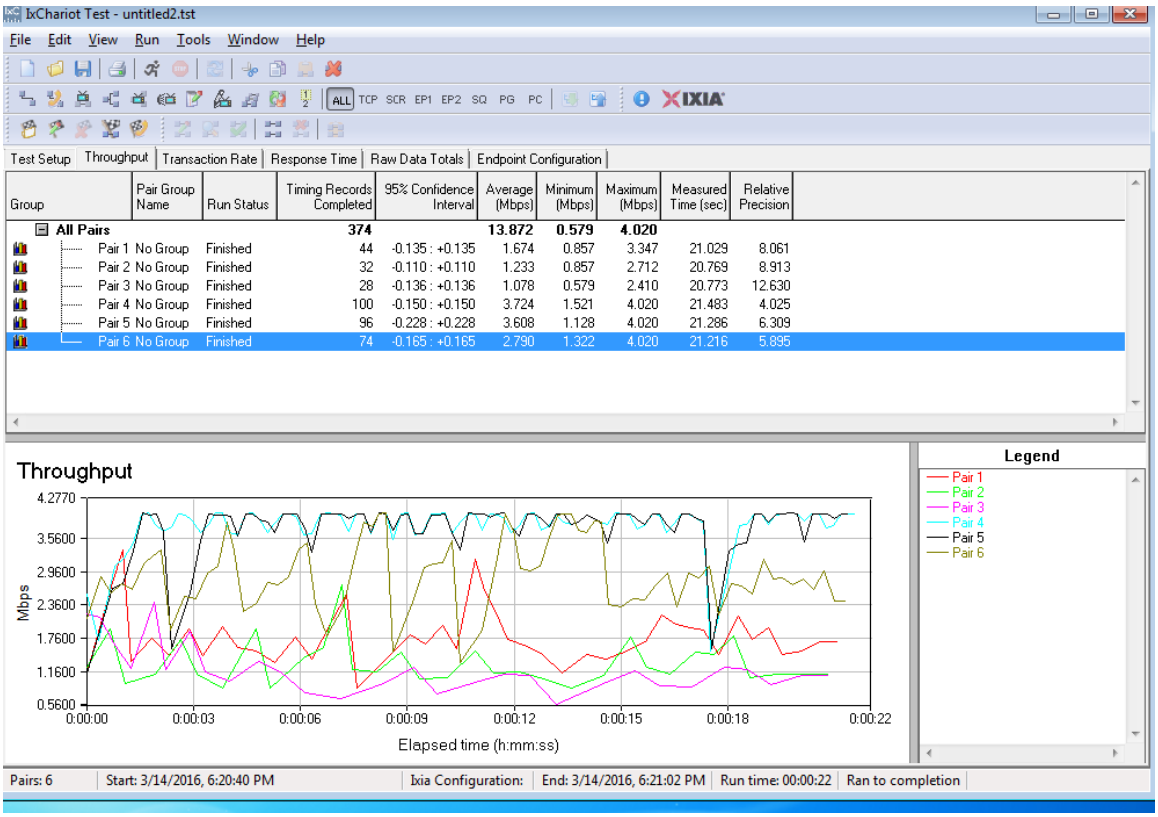


Figure 5-16

Show result of WRED:

```
R1#show policy-map interface
 Ethernet1/0

  Service-policy output: 1

    Class-map: af (match-all)
      27202 packets, 40176530 bytes
      5 minute offered rate 2456000 bps, drop rate 24000 bps
      Match: access-group 101
      Queueing
      queue limit 64 packets
      (queue depth/total drops/no-buffer drops) 0/277/0
      (pkts output/bytes output) 26925/39758632
      bandwidth 5000 kbps
        Exp-weight-constant: 9 (1/512)
        Mean queue depth: 17 packets
        dscp      Transmitted        Random drop      Tail drop        Minimum     Maximum     Mark
                  pkts/bytes         pkts/bytes       pkts/bytes       thresh      thresh      prob

        default       60/6390            0/0              0/0             20          40     1/10
        af11        3183/4771894       63/95382           0/0             28          40     1/10
        af12        2282/3423240       69/103726          0/0             26          40     1/10
        af13        2076/3112116       79/118866          0/0             24          40     1/10
        af21        7095/10383142        6/9084           0/0             34          40     1/10
        af22        6947/10175150      13/19682           0/0             32          40     1/10
        af23        5282/7886700       47/71158           0/0             30          40     1/10

    Class-map: class-default (match-any)
      3 packets, 180 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: any

      queue limit 64 packets
      (queue depth/total drops/no-buffer drops) 0/0/0
      (pkts output/bytes output) 3/180
R1#
```

Figure 5-17

Therefore, the result shows the same concept with IPP. For example, af12 is higher drop probability af11 because 2 in af12 means drop probability, the greater number is, the higher drop probability. And af21 has higher priority than af11.

Furthermore, we have noticed there is a value exp-weight-constant that is default value 9; however, this value can be selected from any number 1-16. It can be adjusted by command *random-detect exponential-weighting-constant n*, the value n is associated with average queue size to determine WRED mode (no drop, random drop, tail drop). The formula is $Q_{ave}(t+1)$ (new average queue size) = $Q_{ave}(t)$ (previous average queue size) * $(1-2^{-n})$ + Q_t (current queue size) * 2^{-n}. Usually, WRED is not reacting a bursty traffic immediately. If the congestion continues in a long term of the bursty traffic, the WRED will decide whether to discard the packets or not. Therefore, that means smaller size of average queue is more sensitive for bursty traffic because queue is easier to be reached threshold. The higher the exponential-weighting-constant is, the less sensitive to even long term bursts of traffic. By raising the value, WRED is becoming reacting slower to bursty traffic. In addition, please note that WRED does not affect UDP traffic because there is ACK mechanism and sliding windows implemented like TCP. The real-time application traffic usually applied in PQ or LLQ, the packets always are transmitted first rather than waiting in the queue.

6. Traffic shaping and policing

Traffic policing is typically control and limit the rate at which traffic is being sent from an interface or being received at an interface. When the traffic exceeds the certain rate, the mechanism will address the packets to be dropped or reset their IPP/DSCP value. Usually, using Committed Access Rate limiting or Class-based policing are to limit rate. An example, limiting HTTP traffic is not to exceed the 50% of network bandwidth. It can be implemented at interface or policy-map. For shaping, it is not like policing which just dropped the exceed packets, and the exceed packets can be buffered and sent later. The shaping is only working for outgoing traffic, unlike policing working on incoming and outgoing direction traffic. Both of them are using token bucket concept to address traffic. Traffic shaping and policing are taking place before queueing, it addresses traffic first then put them into software queue to execute other queueing mechanism.

6.1 traffic policing

In nowadays network structure, when the traffic is transmitting from a high rate link to a lower one, the bandwidth will reach to bottleneck on low rate link, which causes the data lost severely and especially for time-sensitive data such as voice. Traffic policing is typically to avoid this issue by policing entering flow criteria. Limiting rate in a reasonable range protects network resource. For ISP network, it is very necessary to control and monitor the traffic that subscriber sent into network. To enterprise network, controlling and administrating some applications traffic is also necessary. We analyse the policing based on Cisco device. The policing is implemented by token bucket algorithm that addresses the incoming traffic on the basis of pre-set actions. Based on the pre-set actions the policer will either transmit, drop, or transmit the packets with a new remarked IPP/DSCP value. First, I will introduce token bucket algorithm, which includes a token bucket and tokens that are filling during certain time as known as CIR (token arrival rate). Token bucket is like a water bucket, and the tokens are like incoming water from a pipe. In here, we defined three parameters as Bc is burst size (bucket size), Tc is the time interval, and CIR is committed information rate=Bc/Tc. Every time a packet is policed token and determine whether follow the pre-set. If the packet size is lager than tokens, the packet will be discarded or remarked. The tokens will be put back into buckets after it. In this way, the packets can be classified into three categories: conform, exceed, and violate, and it also called three colors or classifications. The CB-policing classifies the three main method based on number of buckets and number of rates. As the result, applied three colors for each of situations.

1st single rate with single token bucket, two colors: conform and exceed

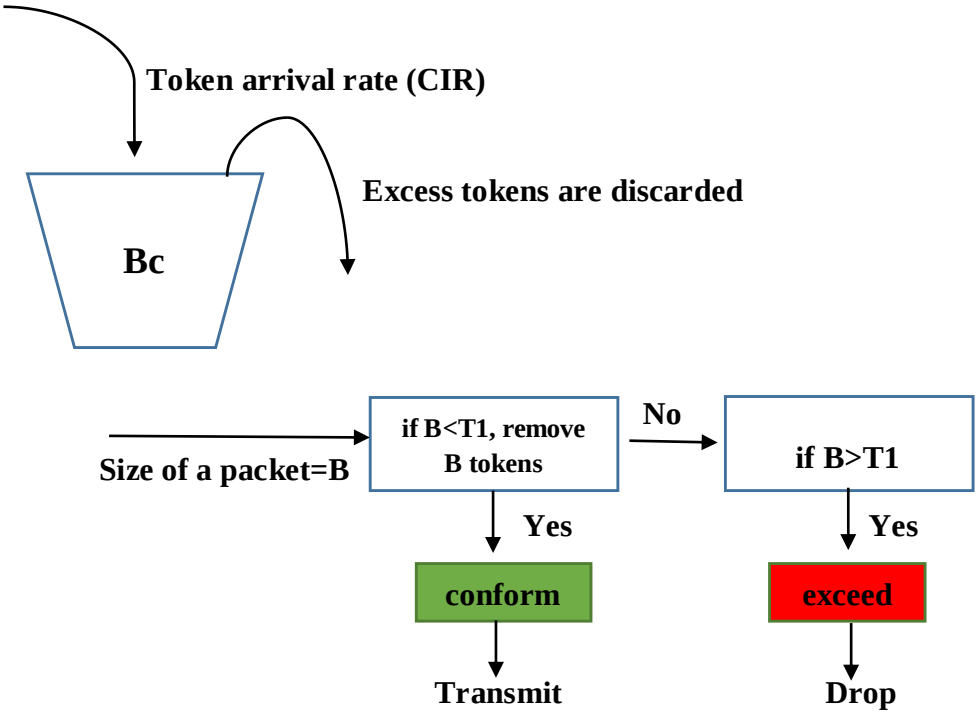


Figure 6-1

http://www.cisco.com/c/dam/en/us/products/collateral/ios-nx-os-software/quality-of-service-qos/prod_presentation0900aecd80312b59.pdf

Bc: bucket size

T1: number of tokens in bucket

B: size of a packet, number of bytes

CIR: committed information rate

The number of tokens refill into the bucket (unit in byte):

(Current_packet_arrival_time – Previous_packet_arrival_time) * Police_rate/8

Every time a packet is policed, CB policing puts some tokens back into the Bucket. If the packet colored with conform, it will take tokens from bucket. The rest of tokens=T1-B. if the packet entered exceed class, it will simply drop the packet and the number of tokens in bucket won't be changed.

2nd single rate with dual token buckets, three colors: conform, exceed, and violate

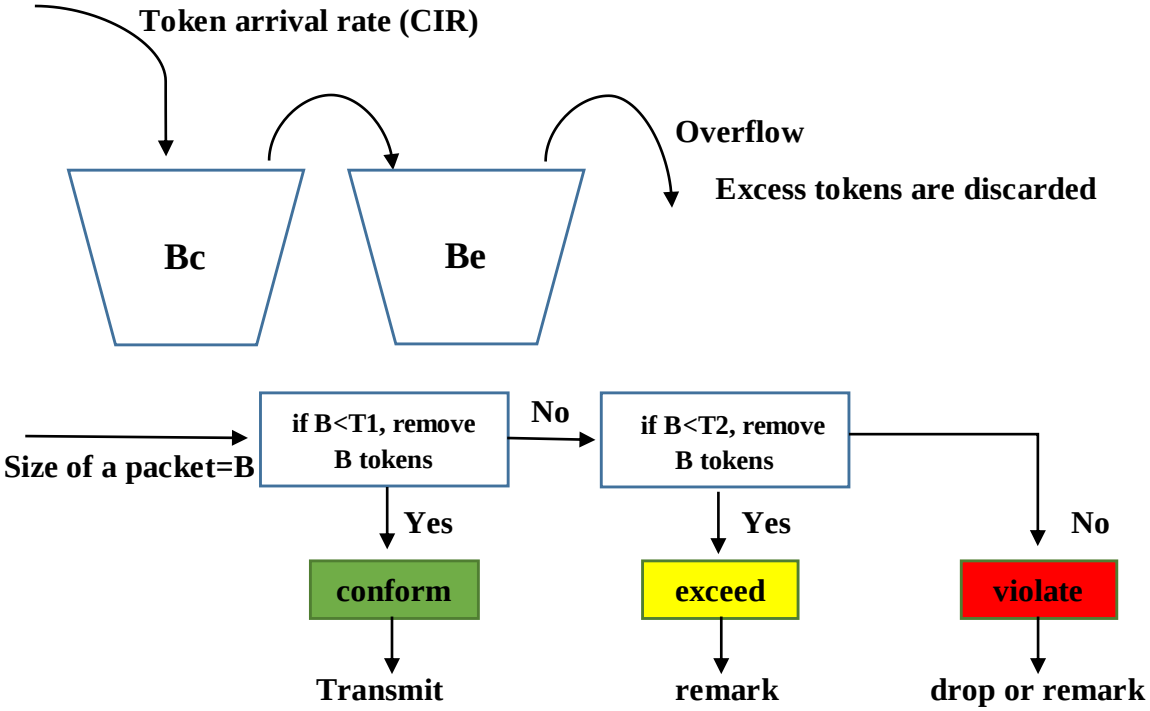


Figure 6-2

http://www.cisco.com/c/dam/en/us/products/collateral/ios-nx-os-software/quality-of-service-qos/prod_presentation0900aecd80312b59.pdf

Bc: bucket size

Be: the second bucket size

T1: number of tokens in bucket Bc

T2: number of tokens in bucket Be

B: size of a packet, number of bytes

CIR: committed information rate

Introducing another bucket into bucket algorithm. The tokens with CIR to fill into bucket, if Bc is filled up, then the excess tokens are going to Be. When Be is overflowed, just simply discards excess tokens. Assumingly, B<T1, then the packet entered conform class with transmit status in CIR rate and took B tokens. The number of tokens in Be won't change. If T1<B<T2, then the packet colored with exceed which means the packet will be remarked and took the tokens from Be, and the T1 wouldn't be changed because tokens were taken from Be. If B>T1, T2, the packet is violated and should be dropped or remarked depends on police. The number of token refill into bucket is the same formula as single bucket and single rate. The additional

bucket (Be) involved into buckets system, and it only increased the tolerance to burst traffic but the transmitting rate will be still keeping CIR.

In here, an additional explanation for remarking. The purpose of remarking can be used for later WFQ queueing to schedule the packets depending on their weight which is remarking the IPP/DSCP value in policing procedure. For example, the conformed packets also can be remarked. The packet is conformed marking with IPP1, and the excess traffic is with IPP0. Once the packets are arriving at queueing packet, WFQ will schedule them, and the higher IPP always gets preference over lower IPP packets. Also, policing remarking affects WRED to determine the probability of dropping packets.

3rd dual rate with dual token buckets, three colors: conform, exceed, and violate

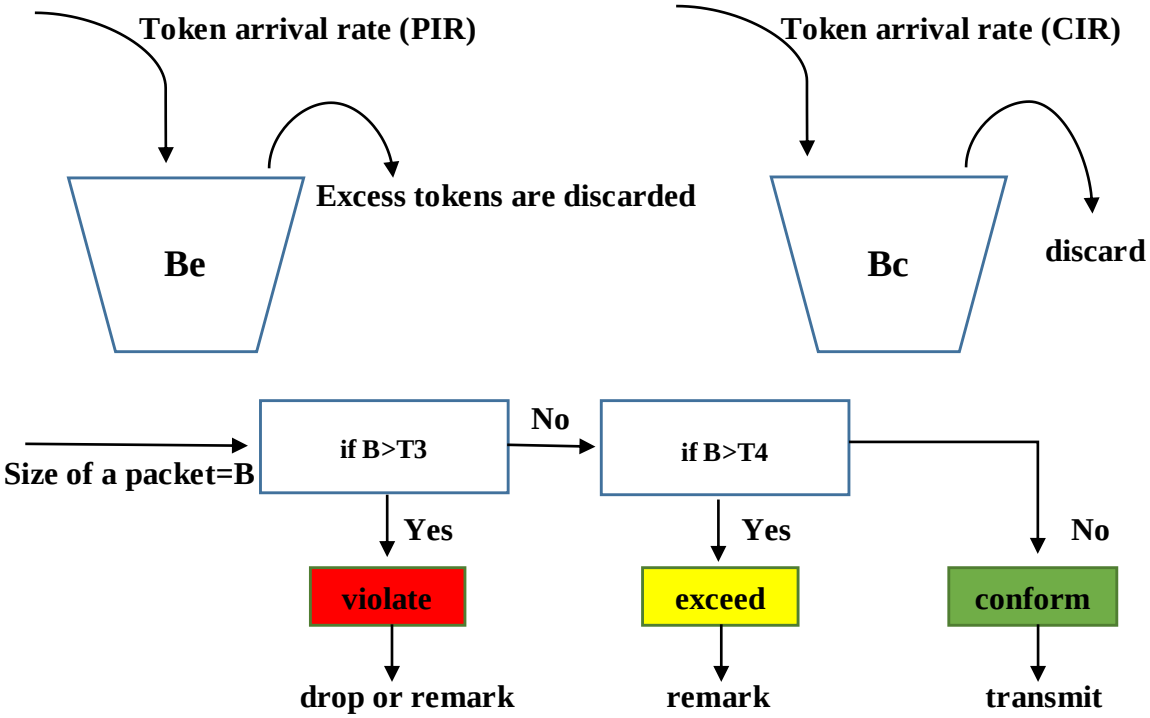


Figure 6-3

http://www.cisco.com/c/dam/en/us/products/collateral/ios-nx-os-software/quality-of-service-qos/prod_presentation0900aecd80312b59.pdf

Bc: bucket size

Be: the second bucket size

T3: number of tokens in bucket Be

T4: number of tokens in bucket Bc

B: size of a packet, number of bytes

CIR: committed information rate

PIR: peak information rate

PIR>CIR, Be>Bc

Introducing a new rate PIR to token bucket algorithm, and the PIR much be greater than CIR. Therefore, it brings the following:

Conform: if the number of byte is less than tokens in both buckets, a packet can take tokens from both of buckets is belong to conform class, and the packet will be transmitted in CIR. The packet is going to take tokens from both of them, so the rest of token is T4-B in Bc and T3-B in Be. The number of tokens refill into each bucket respectively is (Current_packet_arrival_time – Previous_packet_arrival_time) * (PIR or CIR)/8 (unit in byte).

Exceed: the packet can get the sufficient tokens from Be but not form Bc, the packet will be classified into exceed to be remarked and remove tokens from Be bucket. the number of tokens in Bc won't be changed. The transmitting rate for exceeding is PIR-CIR.

Violate: number of bytes in a packet are more than tokens in both bucket, the packet violates. CB policing does not take away any tokens in any bucket, and the packet will perform to be either dropped or remarked.

The figure shows a difference after policing and before:

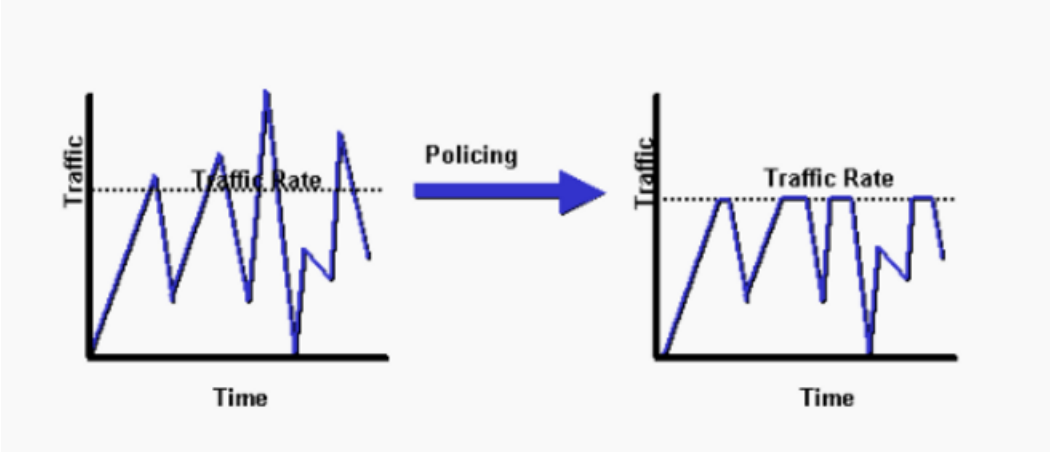


Figure 6-4

http://www.cisco.com/c/en/us/support/docs/quality-of-service-qos/qos-policing/19645-policevsshape.html

The burst traffic reaches the configured rate, and the packet will get dropped or remarked. One thing has to be noticed that the bucket size will affect the sensitivity of burst traffic. The lager bucket is; the more tolerance policing has for burst traffic.

To implement the policing, we can use two methods as CAR and CB policing. CAR is using command line interface, applying the policer to interface to address the

traffic. CB policing is using MQC to create class-map and policy-map to cope with the burst. Usually, it is applied to ingress direction.

Examples of implementing CB-policing:

1st single rate with two colors, and we will see how the bucket size take effect for the burst traffic. Note: due to flaws of the traffic generation application, the output might not be so accurate as shown in figures because is a total throughput value which the application can supply, and which is not the receiving rate at the receiver side.

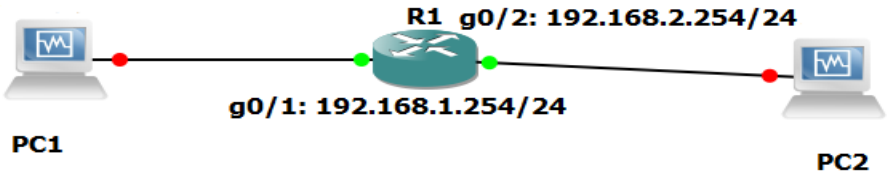


Figure 6-5

PC1:192.168.1.1/24, PC2:192.168.2.1/24, and the traffic is generating from PC1 to PC2 with 10Mbps rate and IPP1. It will be simply configured in the policy-map to limit traffic to 2Mbps and assign the policy on interface g0/2.

Creating class-map to match IPP1, and specifying CIR to the class of traffic. Shown the configuration:

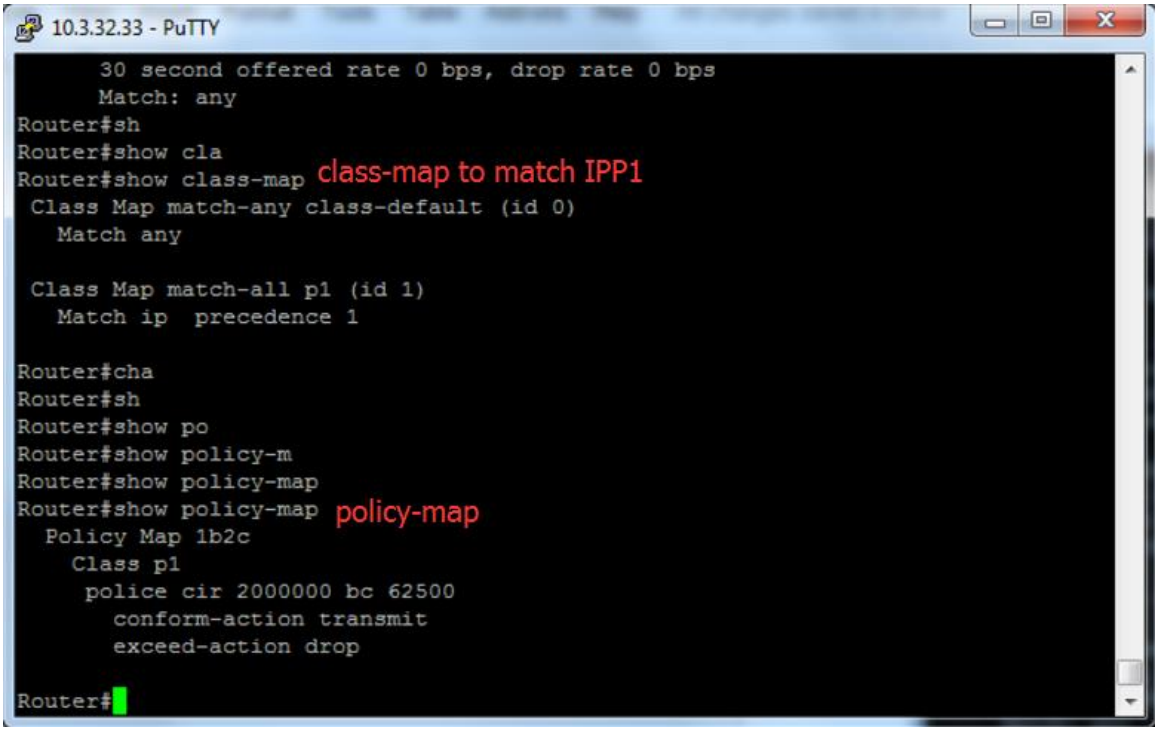


Figure 6-6

Now starting to send traffic from PC1 and show policy-map interface:

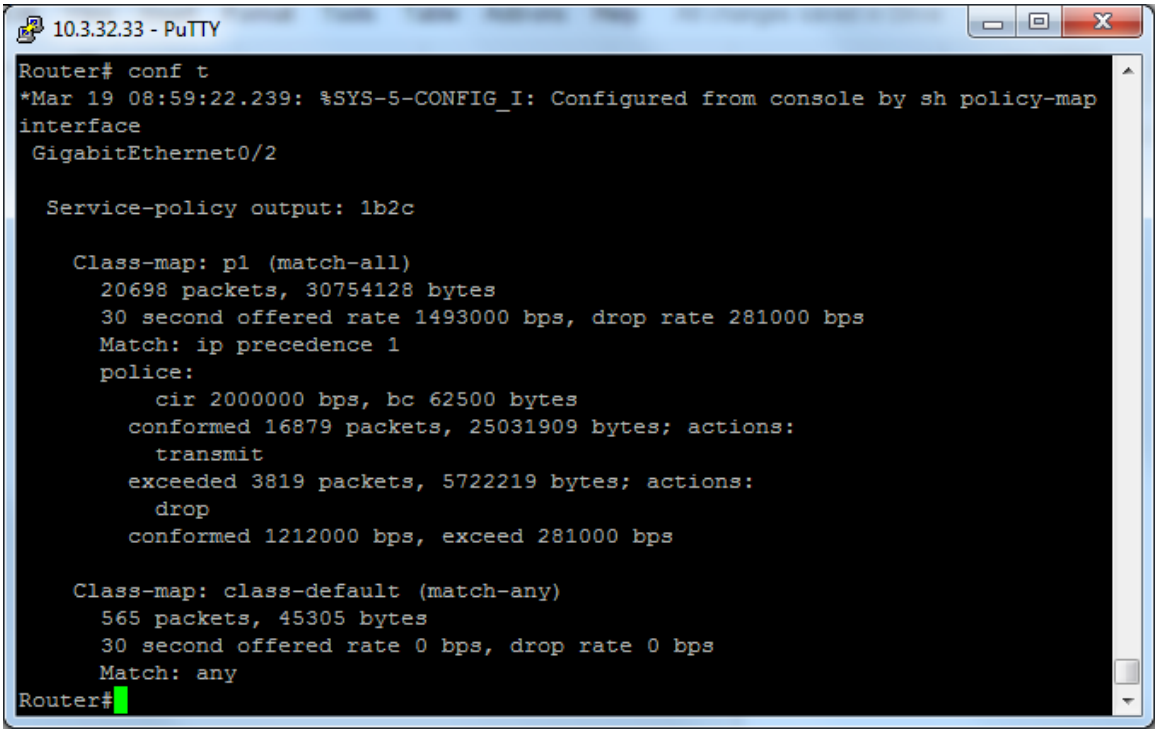


Figure 6-7

From the figure shown, the interface offered 1493000bps is about equal to conformed rate 1212000bps + exceed rate 281000.

The result of policing as following:

Test Setup | Throughput | Transaction Rate | Response Time | Raw Data Totals | Endpoint Configuration |

Group	Pair Group Name	Run Status	Timing Records Completed	95% Confidence Interval	Average (Mbps)	Minimum (Mbps)	Maximum (Mbps)	Measured Time (sec)	Relative Precision
All Pairs			**100**		**1.444**	**0.783**	**3.448**		
Pair 1	No Group	Finished	100	-0.108 : +0.108	1.445	0.783	3.448	55.362	7.444

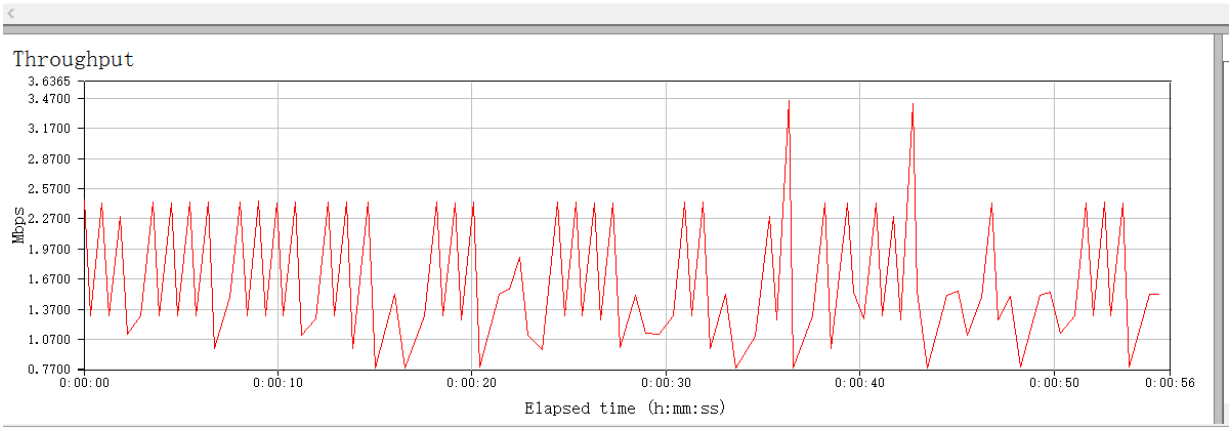


Figure 6-8

We can see the graph shows that the curve changed dramatically up and down, and you will see the shaping mechanism how the curve goes. It will definitely show the difference.

Now let change the bucket size and see the change of curve. The default size was 62500 bytes and now change to 150000 bytes twice of the previous size. The tolerance of the burst traffic has been changed dramatically, even in some moment the rate reached to 10Mbps which originally generated by PC1. The figure shows as

below:

Group	Pair Group Name	Run Status	Timing Records Completed	95% Confidence Interval	Average (Mbps)	Minimum (Mbps)	Maximum (Mbps)	Measured Time (sec)	Relative Precision
All Pairs			**100**		**1.940**	**0.964**	**10.000**		
Pair 1	No Group	Finished	100	-0.208 : +0.208	1.943	0.964	10.000	41.178	10.718

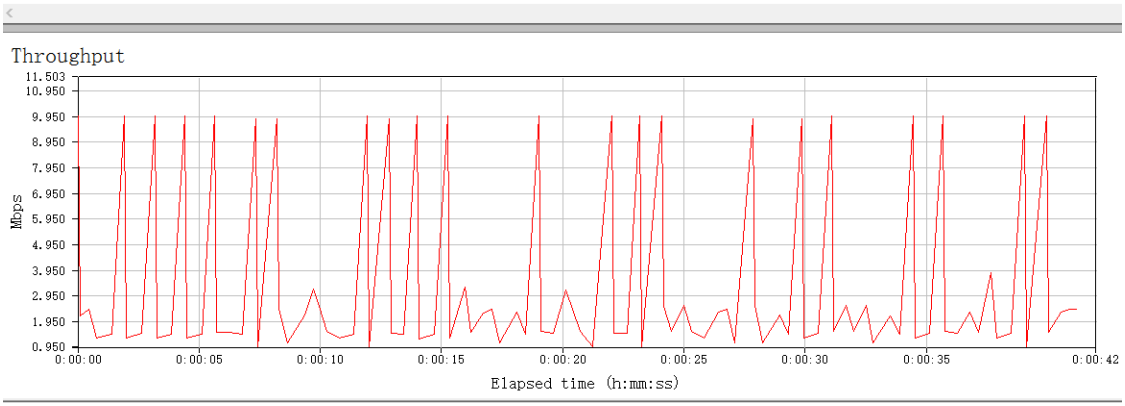


Figure 6-9

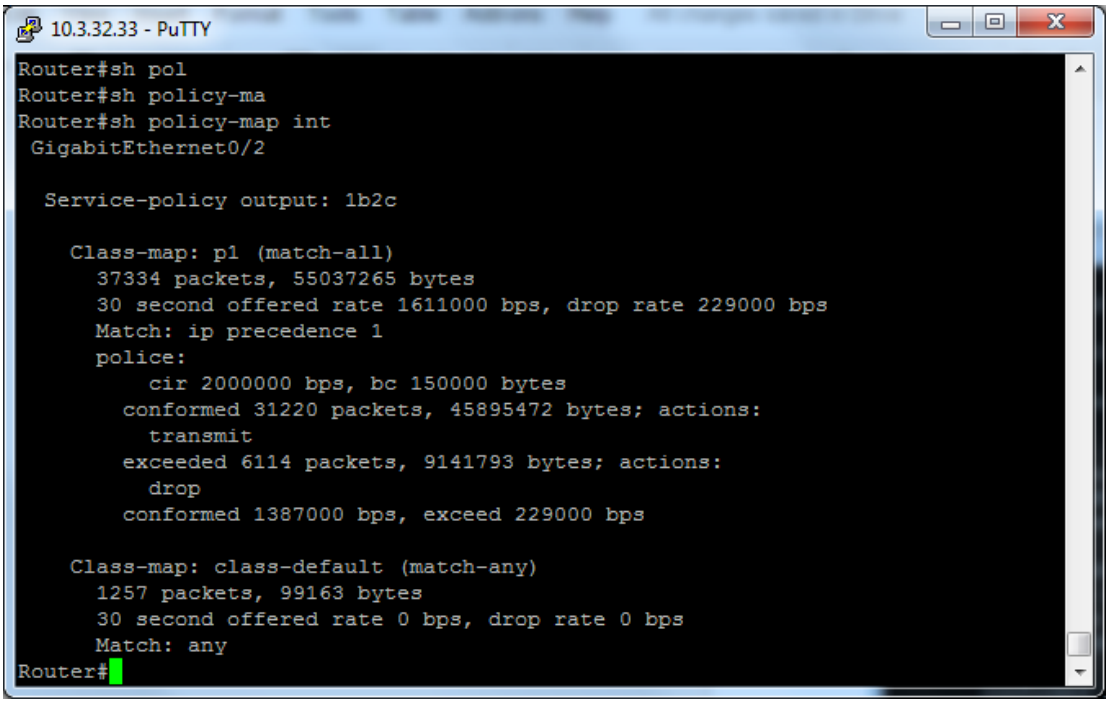


Figure 6-10

2nd an example of single rate with three colors, and using same scenario. Change the configuration in policy-map:

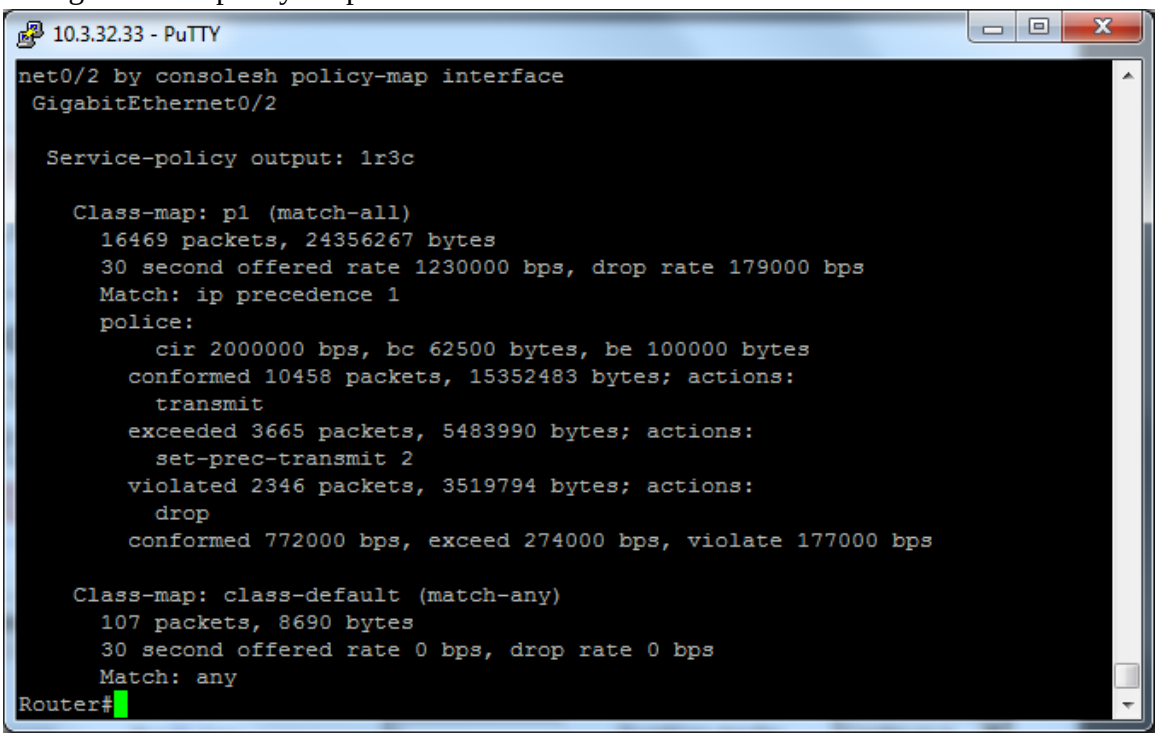


Figure 6-11

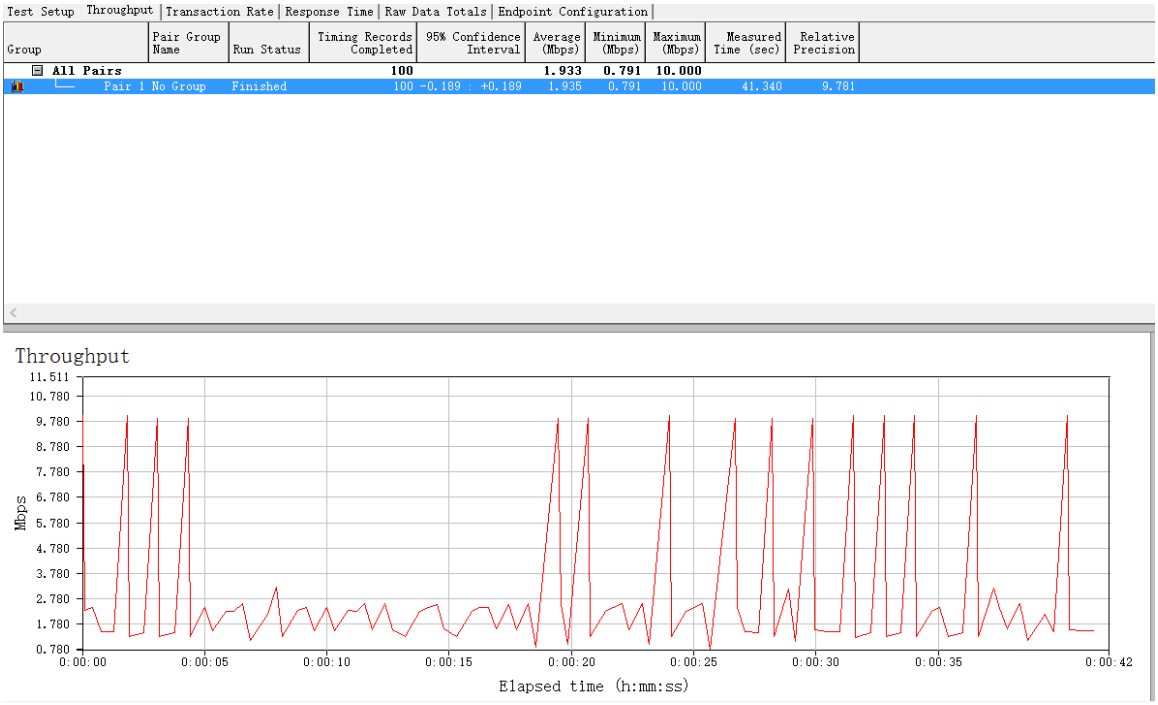


Figure 6-12

Conformed rate still keeps 2000000bps, and introduce another bucket Be, which just increased the tolerance of burst. That is the reason why the traffic can reach sending rate at some moments, and exceeded packets were remarked to IPP2 captured in Wireshark as DSCP CS2 that is equivalent to IPP2. The violated packets just simply got dropped.

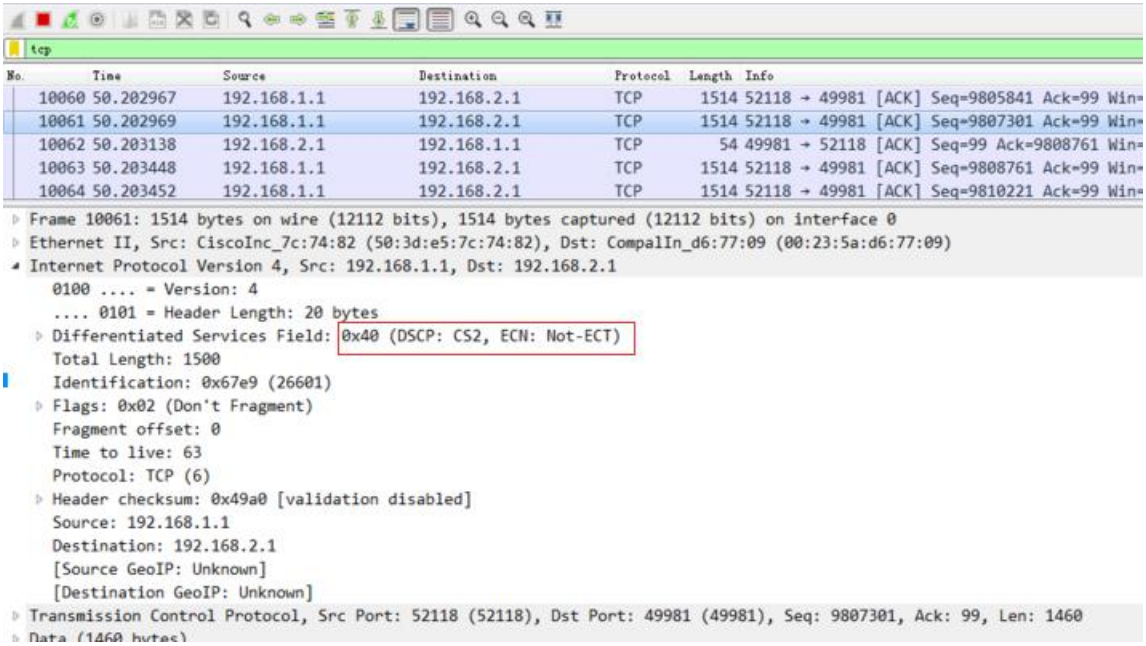


Figure 6-13

3rd two rate with three colors

```
Router#sh policy-map 2r3c
  Policy Map 2r3c
    Class p1
     police cir 2000000 bc 62500 pir 3000000 be 93750
       conform-action transmit
       exceed-action set-prec-transmit 2
       violate-action drop
```

Figure 6-14

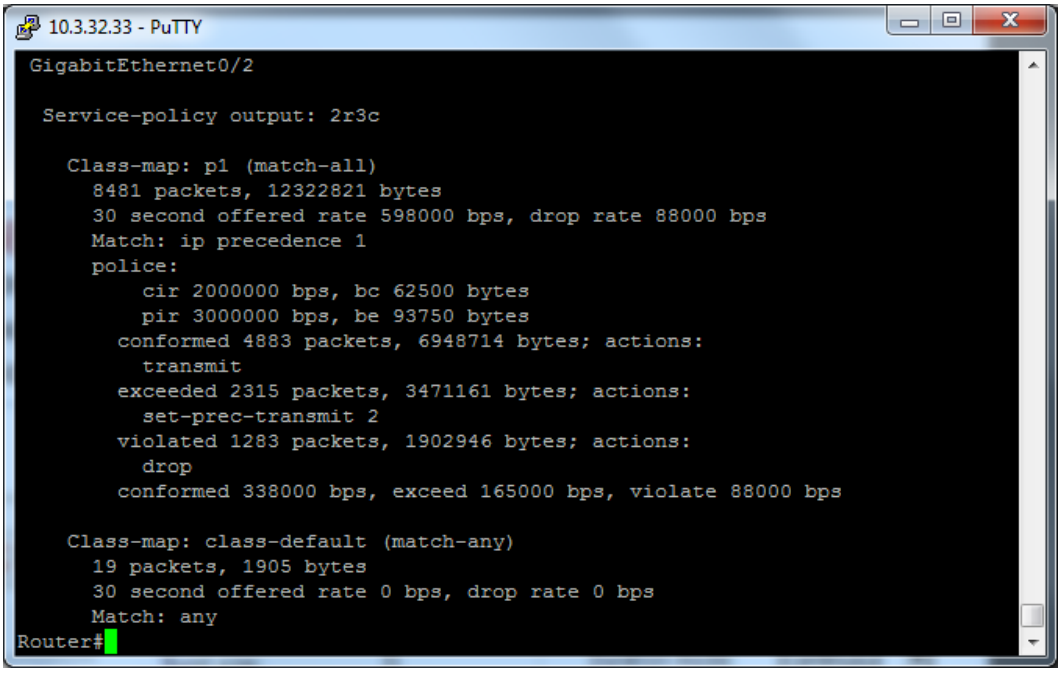
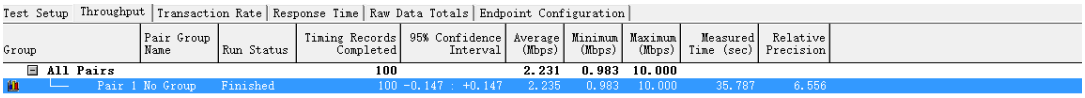


Figure 6-15



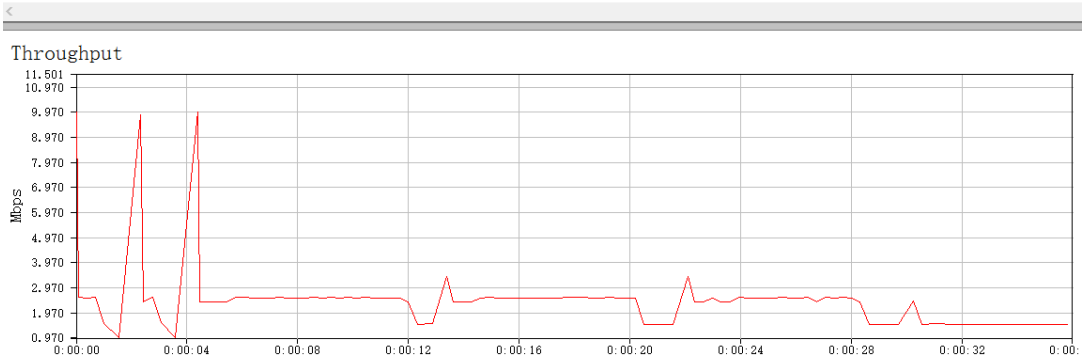


Figure 6-16

The conformed traffic is still keep same rate as CIR, but the total throughput got increased because another rate has been introduced. The exceeded traffic got rate as PIR-CIR, and the violated packets were discarded.

6.2 Traffic shaping

Comparing with traffic policing as dropping the packet without buffering them when a packet size more than tokens, so shaping introduces a buffer where temporarily store the excess rate packets and schedule them to be sent out later with the previous configured rate while the bucket has idle tokens. It will facilitate the output rate much smoother. It is not like policing can be used at incoming and outgoing direction, and shaping only can be executed at outbound interface to limit the traffic rate.

The packets might get lost when the traffic is entering into high speed interface and leaving from low speed interface. The shaping will help to buffer the packets instead discard them and use a lower rate than the capability of the egress interface. Traffic shaping is using same algorithm of token bucket which assign tokens to a packet to decide what amount of bits can be sent out at certain period of time. However, there is only one bucket introduced in this algorithm. A few parameter is included such as CIR which you paid for ISP to give you the amount of resource, Bc (committed burst) which is meant the amount of bits of traffic can be sent during Tc, Be (the second bucket, excess burst), and Tc (time interval) the duration of sending Bc bits of traffic. The maximum tokens in the bucket is Bc+Be. The relation among them as a formula Tc=Bc/CIR (in seconds). If the incoming rate of traffic is less than CIR, it will be send in Tc; otherwise, it will be buffered in the queue and wait till next Tc coming to be available to have sufficient token in the bucket to carry the buffered packets. Here is an example shows how the shape the traffic as CIR if the rate exceeds. The actual physical interface capability is 100Mbps, and you want to shape the traffic to 50Mbps. The default Tc in cisco routers is 125ms, we take 50% of time to send the traffic in rate 100Mbps and another 50% of time to pause the traffic. Hence, the shaping rate=50%*100Mbps. According to the above formula, Bc=Tc*CIR=125ms*50Mbps=6250kbits. In 125ms, the amount of bits is supposed to be sent 125ms*100Mbps, but 50% pausing exists in algorithm. The amount if bits is 62.5ms*100Mbps instead, so the traffic is shaped to 50Mbps.

The figure of shaping with time arrangement:

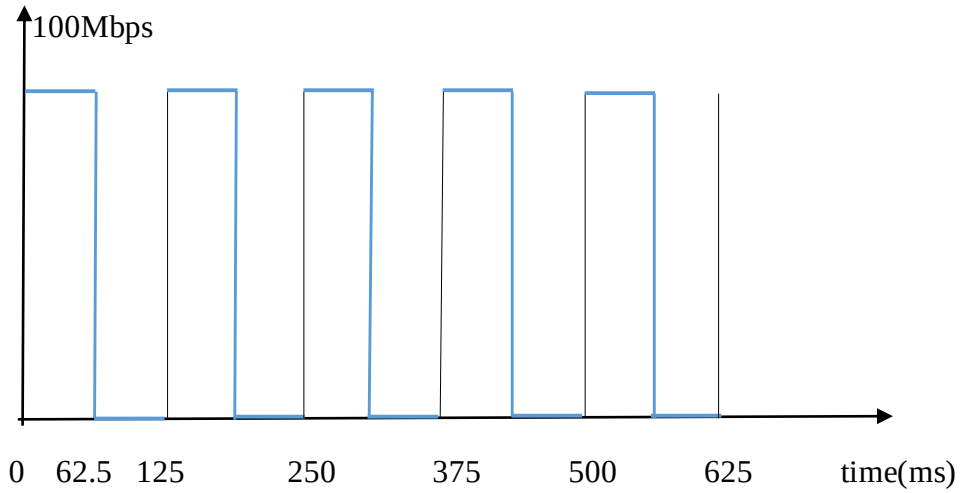


Figure 6-17

https://networklessons.com/quality-of-service/qos-traffic-shaping-explained/

The illustration of shaping algorithm:

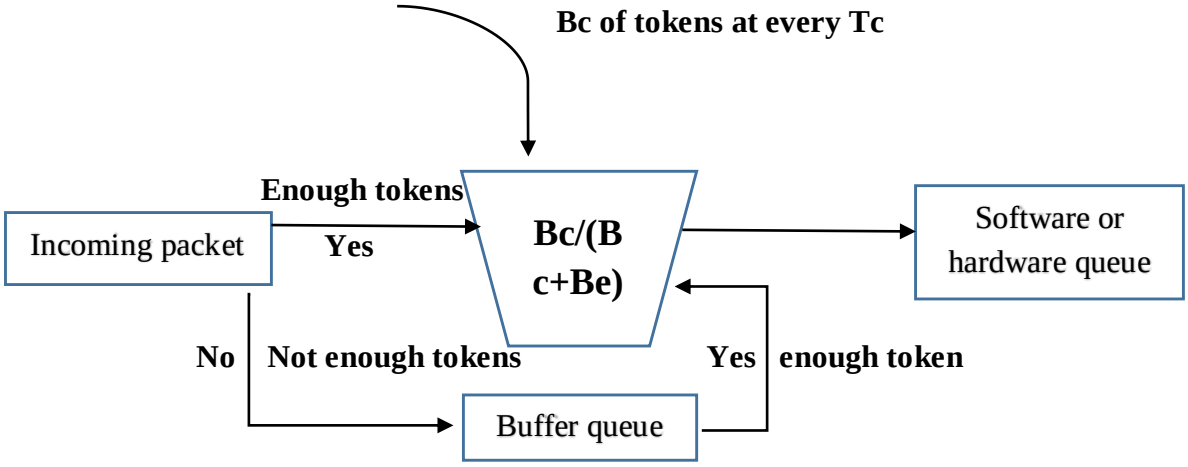


Figure 6-18

http://www.cisco.com/c/en/us/td/docs/ios/12_2/qos/configuration/guide/fqos_c/qcfpolsh.html

The maximum bucket size is Bc+Be, Be>0 allows bursting.

The figure shows a difference after shaping and before:

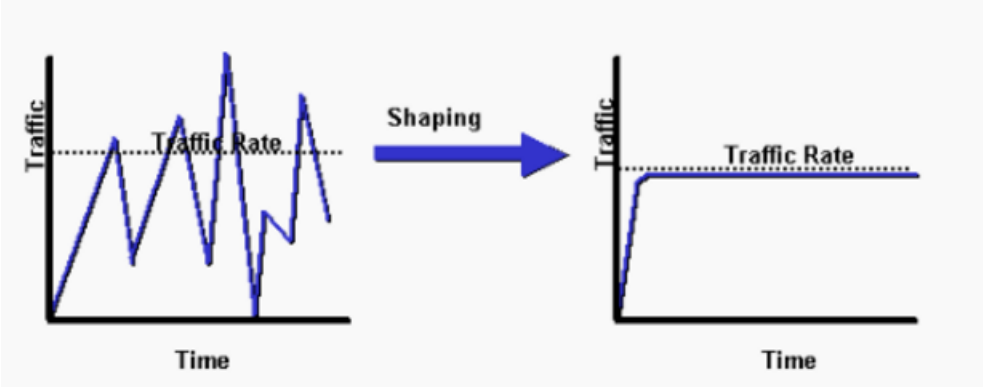


Figure 6-19

Examples of implementing shaping in cisco router:

CB-shaping test:

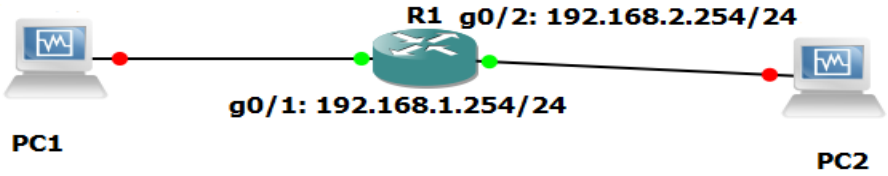


Figure 6-20

Set 4 flows: flow1 IPP1 with 10Mbps, flow2 IPP2 with 10Mbps, flow3 IPP3 with 10Mbps, flow4 IPP6 with 10Mbps.

1st test average rate:

Assign the policy-map on interface g0/2 egress direction:

```
interface GigabitEthernet0/2
 ip address 192.168.2.254 255.255.255.0
 load-interval 30
 duplex auto
 speed auto
 service-policy output shape_av
```

Figure 6-21-a

```
Router#sh class-map p1
 Class Map match-all p1 (id 1)
   Match ip  precedence 1
```

Figure 6-21-b

```
Router#sh policy-map shape_av
  Policy Map shape_av
    Class p1
      Average Rate Traffic Shaping
      cir 2000000 (bps)
Router#
```

Figure 6-21-c

Here shows policy-map interface:

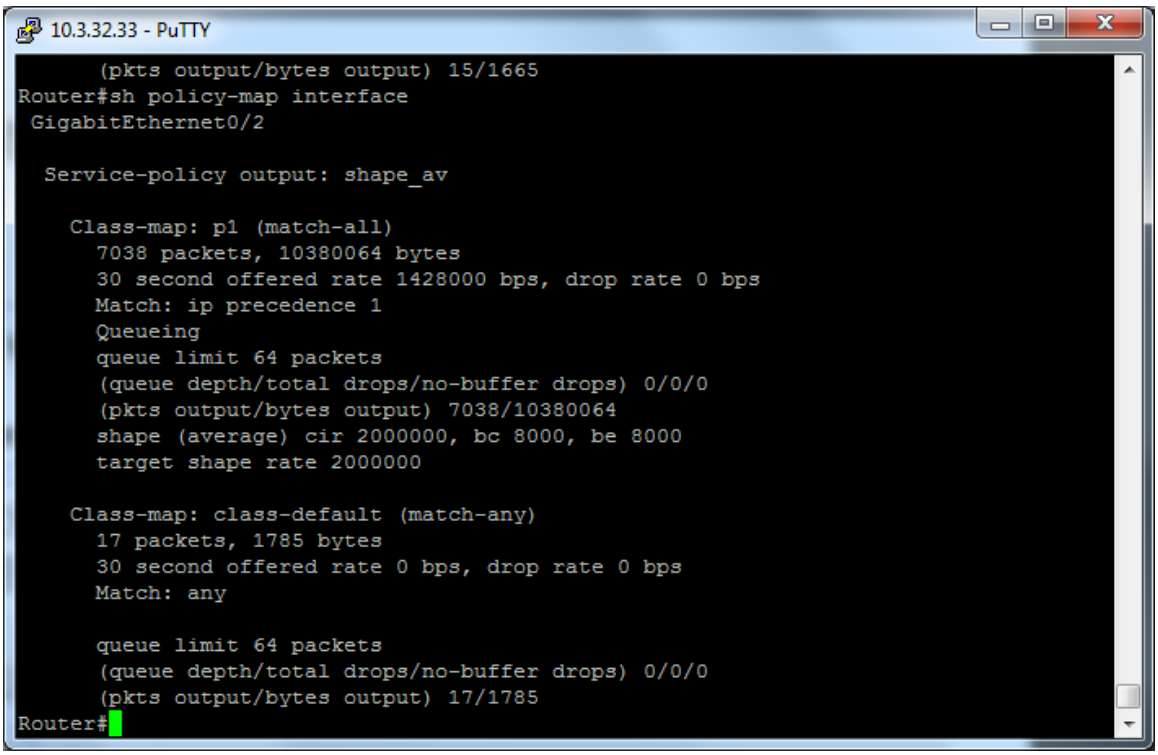


Figure 6-22

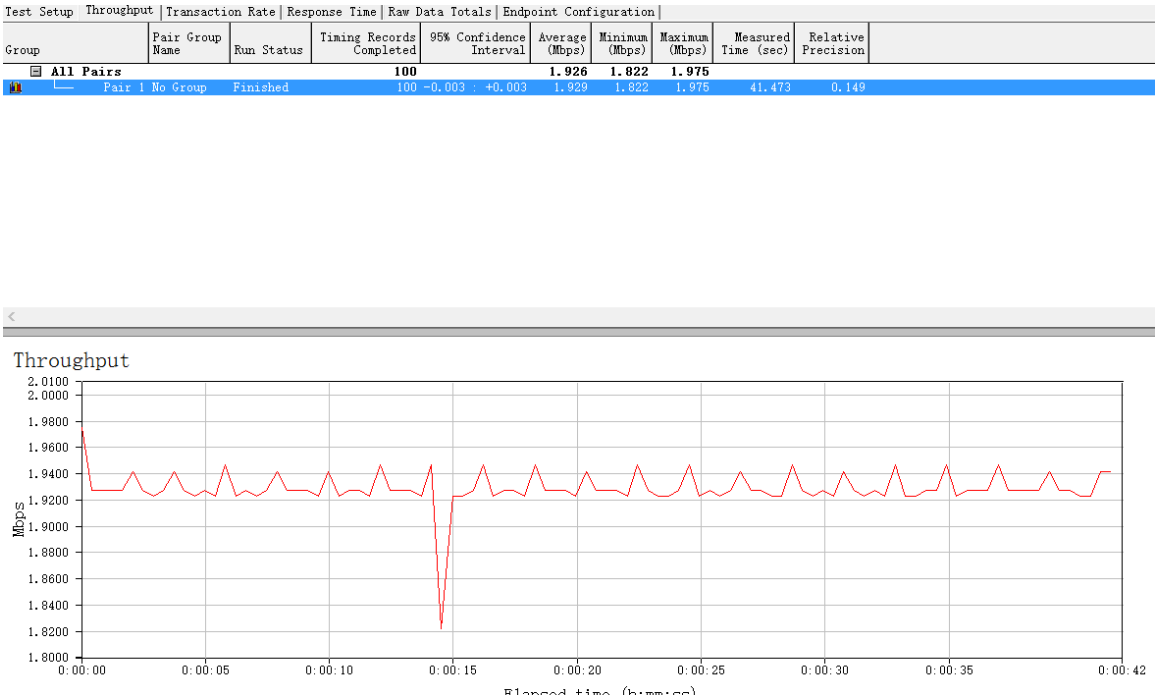


Figure 6-23

It obviously shows that the curve becomes smoother comparing with policing, because buffer introduced in the shaping mechanism and the packets don't get discarded when exceeding the limited rate. Actually, configuring the Be size does not really affect the average rate, but it can increase the tolerance to burst.

Adding another flow to transmission with 10Mbps and IPP6, and change the speed of interface to 10Mbps. Thus, if two flows transmit simultaneously, congestion will occur. Let see the result of overflowed traffic.

Test Setup | Throughput | Transaction Rate | Response Time | Raw Data Totals | Endpoint Configuration |

Group	Pair Group Name	Run Status	Timing Records Completed	95% Confidence Interval	Average (Mbps)	Minimum (Mbps)	Maximum (Mbps)	Measured Time (sec)	Relative Precision
All Pairs			**124**		**9.403**	**1.887**	**7.843**		
Pair 1	No Group	Finished	24	-0.002 : +0.002	1.898	1.887	1.905	10.114	0.106
Pair 2	No Group	Finished	100	-0.016 : +0.016	7.621	7.477	7.843	10.497	0.216

Figure 6-24

The shaping maintains target rate and the rest of bandwidth given to flow6, so which means shaping can guarantee the minimum bandwidth when congestion occurs.

And then, changing the shaping scheme to peak.

```
Policy Map shape_pe
  Class p1
    Peak Rate Traffic Shaping
    cir 2000000 (bps) bc 8000 (bits) be 4000 (bits)
```

Figure 6-25

And show policy-map interface:

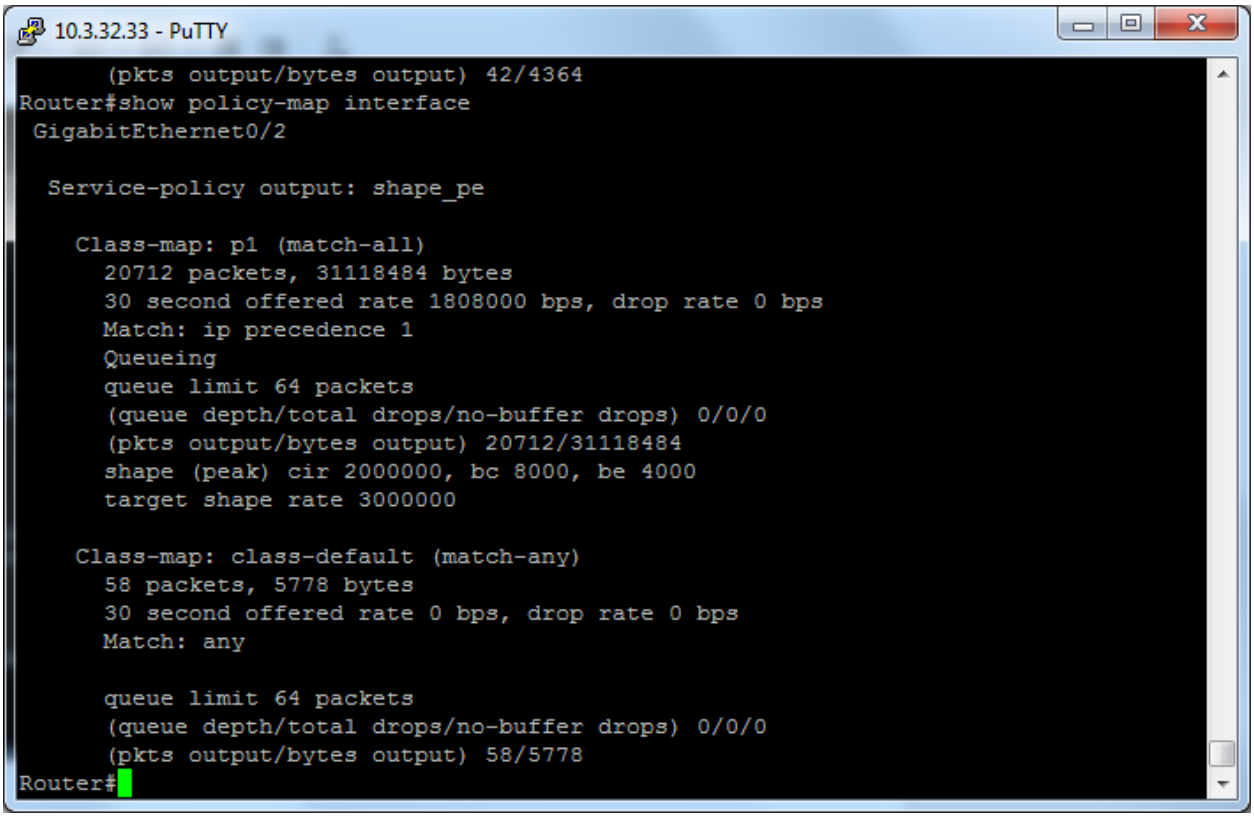


Figure 6-26

Within peak rate get invovled, the formula of traget rate after shaped: target rate=(1+Be/Bc)*CIR. In which we set Be=4000bits and Bc=8000bits, so target rate=3/2*2000000=3000000bps as same as below figure here:

Test Setup | Throughput | Transaction Rate | Response Time | Raw Data Totals | Endpoint Configuration

Group	Pair Group Name	Run Status	Timing Records Completed	95% Confidence Interval	Average (Mbps)	Minimum (Mbps)	Maximum (Mbps)	Measured Time (sec)	Relative Precision
All Pairs			**100**		**2.888**	**2.623**	**2.909**		
Pair 1	No Group	Finished	100	-0.007 : +0.007	2.894	2.623	2.909	27.642	0.235

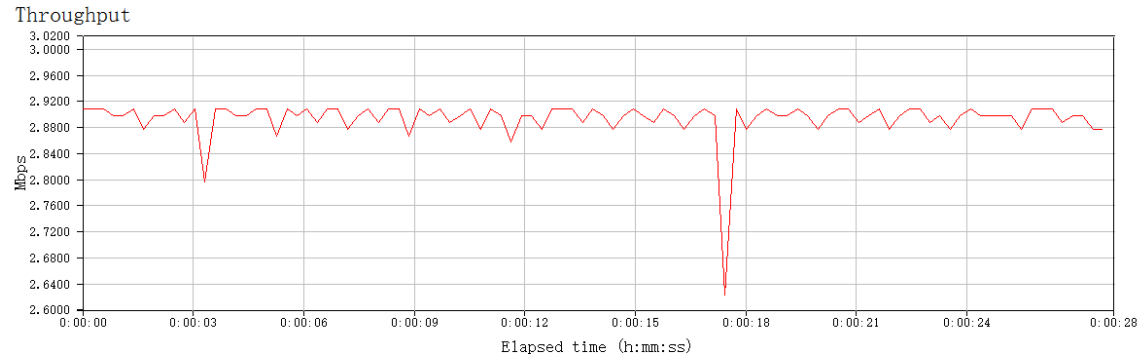


Figure 6-27

Now we add another flow6 and run the traffic from PC1, here shows the result of two flows process:

Test Setup | Throughput | Transaction Rate | Response Time | Raw Data Totals | Endpoint Configuration

Group	Pair Group Name	Run Status	Timing Records Completed	95% Confidence Interval	Average (Mbps)	Minimum (Mbps)	Maximum (Mbps)	Measured Time (sec)	Relative Precision
All Pairs			**142**		**9.485**	**2.759**	**6.957**		
Pair 1	No Group	Finished	42	-0.009 : +0.009	2.813	2.759	2.888	11.945	0.305
Pair 2	No Group	Finished	100	-0.025 : +0.025	6.705	6.557	6.957	11.931	0.379

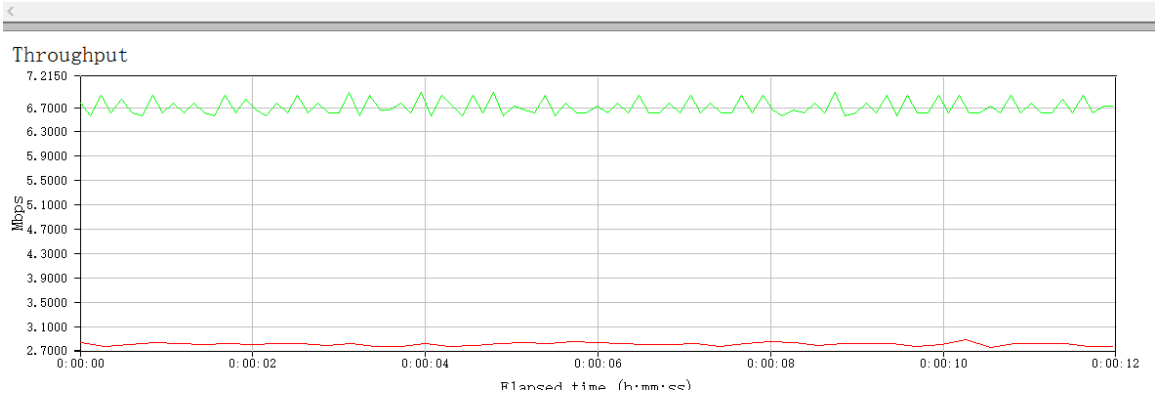


Figure 6-28

The target rate gets guaranteed, and rest of available bandwidth assigns to flow6.

There are serval ways to utilize shaping combining with other mechanism such as priority queueing. Shaping also can configure with deep concept as policy in policy. Here is an example:

The configuration on R1 is:

Creating a access-list to match all traffic from PC1 to PC2, and assign a average rate in 2000000bps executed in policy-map parent. Builting up other two class-map respectively to match IPP1 and IPP2, and then specify arverage rate in policy-map child which will be invoked in policy-map parent. Therefore, the average rate from child will obtain the bandwidth from the parent that defined the maximum average rate. It implemented the CB-policing from different layers to gain more controls over policies.

The configuration for class-map and policy-map showing as following: deploying the policy-map parent on interface g0/2 outbound direction.

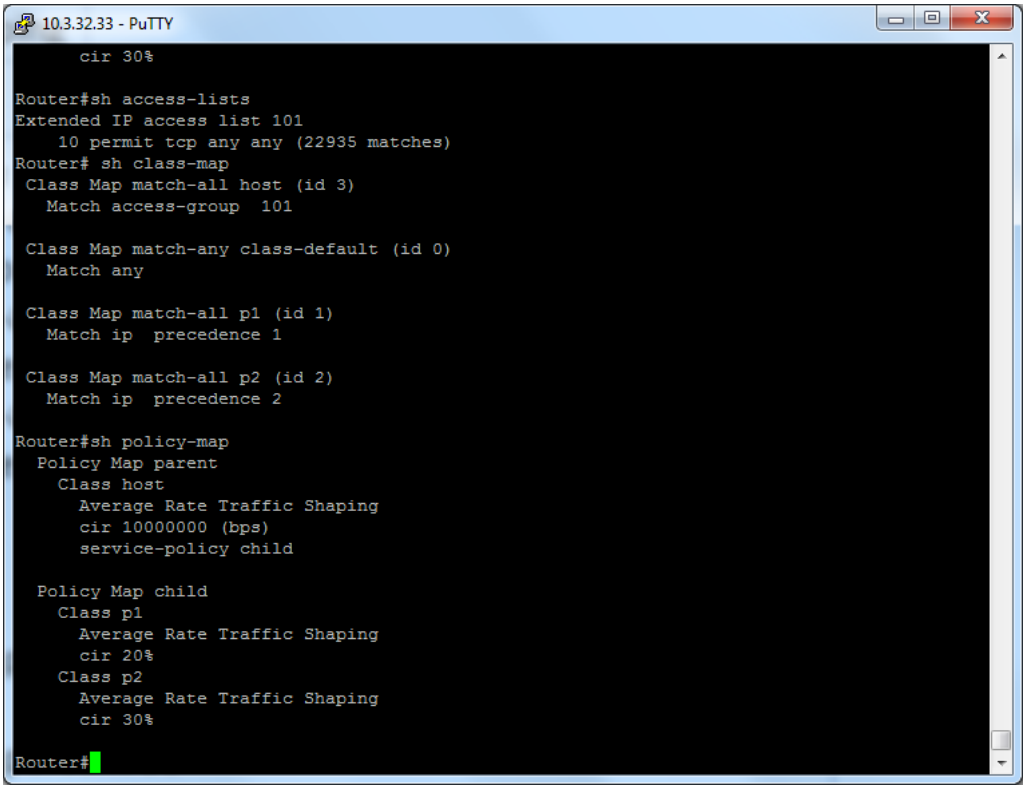


Figure 6-29

The result of demonstrating the configuration:

Test Setup | Throughput | Transaction Rate | Response Time | Raw Data Totals | Endpoint Configuration |

Group	Pair Group Name	Run Status	Timing Records Completed	95% Confidence Interval	Average (Mbps)	Minimum (Mbps)	Maximum (Mbps)	Measured Time (sec)	Relative Precision
All Pairs			**326**		**9.430**	**0.820**	**4.878**		
Pair 1	No Group	Finished	66	-0.002 : +0.002	1.931	1.914	1.966	27.342	0.127
Pair 3	No Group	Finished	100	-0.004 : +0.004	2.898	2.837	2.952	27.607	0.149
Pair 4	No Group	Finished	90	-0.223 : +0.223	2.632	1.039	4.878	27.355	8.481
Pair 5	No Group	Finished	70	-0.240 : +0.240	2.043	0.820	4.734	27.406	11.755

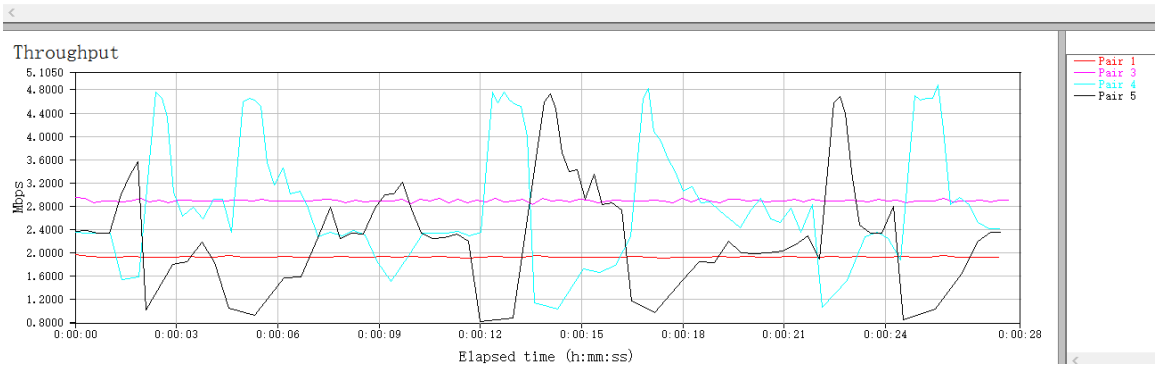


Figure 6-30

Pair1- flow1 IPP1, Pair3- flow2 IPP2, Pair4- flow3 IPP3, Pair5- flow6 IPP6, and all give bandwidth 10Mbps. It is obvious to see flow1 and flow2 is ensured to be granted the expected rate as configured. In addition, the curves of them keep smooth during the period of time, but other two flows without shaping was getting up and down frequently. The propagating rate they have is trading off and taking turns, which means one is increasing, another instead decreasing.

6.3 Over-subscription

Introduction

After we introduced the functionalities of policing and shaping which practically applied to over-subscription, and ISPs mostly apply these techniques to achieve that more efficiently utilize network resource without huge cost of upgrading their hardware specifications. First, let us take a look the definition of over-subscription; literally, over-subscription descripts that something has been over used and exceed its maximum capability to arrange things such as a car renting company having 10 cars and wanting to rent 15 customers concurrently. It exceeds the availability of number of cars they can arrange. In the case of ISP business, over-subscription is often related to represent in percentage of how much resource being subscripted which the value can be either less or more than 100%. For instance, customer subscribes (purchases) two 50Mbps virtual connections to a 100Mbps UNI (user network interface, physical port), so there is no oversubscription appeared; both links can reach up to their maximum capability and give guaranteed accessing speed for customers. However, adding two more connections with 50Mbps to the same 100Mbps UNI, and the ratio

of subscription will be increased to 200%, which might cause the congestion on the UNI ingress interface and break the SLA of QoS that is not guaranteed within congestion. In the SLA, the service provider should ensure low delay, jitter, and packet loss for customer, specially for time-sensitive applications. How we implement best performance for customers when customers over-subscribe bandwidth and also guarantee the minimum rate, becomes a considerable topic being responsible for ISPs to solve. The traditional technique is TDM-based (time division multiplexing), which is sharing the network resource. As the increase of customers, it urgently requires ISPs to upgrade their equipment. Therefore, the best solution for ISP to allocate bandwidth is using policing or shaping technique to be assigned to ingress PE router or egress CE router to control the traffic.

Bandwidth profiles

A new definition has been introduced by service provider is bandwidth profile to allow service provider to sell the bandwidth to ensure the performance of data transmission in in-profile service, and the bandwidth subscriber needs should be less than UNI physical maximum speed. It is often related to SLA to assign resource to subscribers at ingress and egress routers. Here the in-profile means the rate which allow the traffic toward PE router, and the out-of-profile means excess rate can be allowed go inside of ISP core but no service performance insurance. The non-committed rate is the excess rate traffic will get discarded when entering ISP network.

Also, the bandwidth profile can be categorized into two classes, which is referred to the direction egress and ingress. The two classes of bandwidth profiles will be introduced with policing and shaping. The policing can be applied to both directions, unlike the shaping only can be used to egress interface. Both techniques apply token bucket algorithm to divide traffic rates.

Policing for over-subscription

Policing to facilitate implementing and is extensive application to over-subscription, and divide the traffic into three classes or colors:

- green: in-profile - committed rate with service performance insurance.
- yellow: out-of-profile - excess rate transmitting traffic without service performance insurance.
- red: non-committed - violate rate, drop packet immediately.

In the concept of cisco, there are a few parameters including as:

CIR (committed information rate), PIR (peak information rate), Bc (committed burst size), Be (excess burst size), PIR>CIR.

The policing is using three combinations of token bucket and the number of rates. It can be using single bucket and single rate, which if a packet size is less than Bc, the packet will take the tokens and transmitted; otherwise, it gets discarded immediately. There are only two color involved. You can understand that if the incoming traffic

rate is higher than CIR, the policing will be working on it and cut the excess rate to meet the same rate as CIR, and other rate just simply discarded. The result of implementing policing of single rate and single bucket has been shown above content at policing chapter.

Another is single rate and two token buckets, which introduces two burst size. An incoming packet either choose Bc or Be, and it provides more tolerance for the burst traffic.

The third one is dual rates and dual buckets; it is mostly often used policing approach for implementing over-subscription. It uses all the parameters above introduced. The CIR is the committed rate which service provider provides guaranteed service performance, and if the incoming traffic rate is less than CIR; it gets forwarded as same rate immediately and belong to in-profile. In case of excess traffic rate, if the rate is higher than CIR and less than PIR, the traffic will be divide into two classes, which one class (green) will be transmitted as CIR as in-profile committed rate, and another class (yellow) will be transmitted and remarked as incoming rate-CIR in which the traffic might not get same service performance as green class. If the incoming traffic rate is higher than PIR, the policing will classify this flow into three colors which green is transmitting with CIR in-profile, yellow should be transmitted with PIR-CIR out-of-profile, and red class just immediately gets discarded. The illustration of token bucket algorithm as previous mentioned:

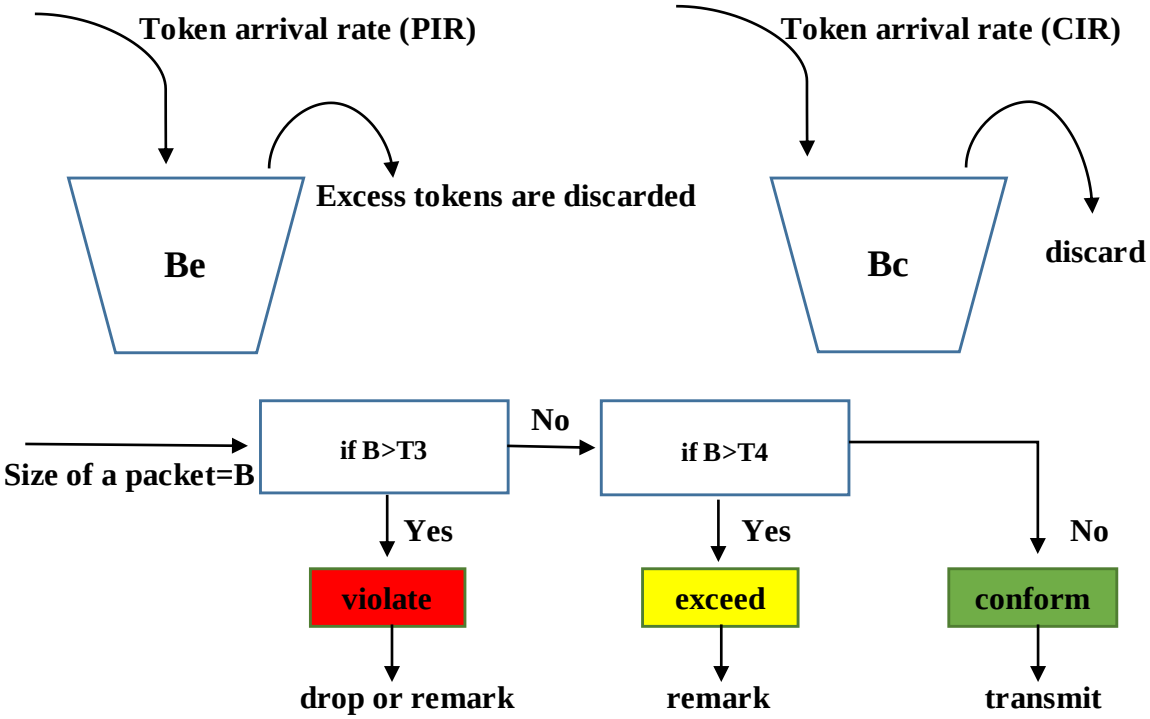


Figure 6-31

http://www.cisco.com/c/dam/en/us/products/collateral/ios-nx-os-software/quality-of-service-qos/prod_presentation0900aecd80312b59.pdf

The policing approaches can be applied in ingress or egress interface, and the result of three policing approaches have been shown in above content.

Shaping for over-subscription

The shaping can help to deal with over-subscription as well, and difference between shaping and policing is shaping won't drop the packets. Instead, it takes the excess traffic which did not get sufficient tokens will be put into a buffer waiting till the tokens become available to be used for excess traffic. Here is similar as policing parameters including CIR (committed information rate), PIR (peak information rate), Bc (committed burst size), Be (excess burst size), and Tc (time interval), but the algorithm only utilizes one bucket and one rate. The bucket size can be either Bc or Bc+Be which increase the capability to tolerant burst traffic. In cisco devices, the configuration can be two choices which are average and peak. The average rate is actually same as CIR when the incoming traffic rate higher and CIR, and the excess packets will be buffered inside of a queue waiting for next Tc coming. If we configure shaping with peak, the actual rate will be (1+Be/Bc)*CIR. Thus, using shaping approach in egress interface can limit rate and bring a flexibity of adjusting maximum rate.

Ingress and egress bandwidth profiles

Ingress – policing only	Egress – policing or shaping
Per-UNI ingress bandwidth profile	Per-UNI egress bandwidth profile
Per-EVC ingress bandwidth profile	Per-EVC egress bandwidth profile
Per-DSCP ingress bandwidth profile	Per-DSCP egress bandwidth profile

Table 6-32

https://www.mef.net/Assets/White_Papers/Understanding_MEF_6.2_Bandwidth_Profiles_FINAL.pdf

Per-UNI ingress bandwidth profile is based on capability of physical port, and all service will be treated as one pipe transmission. if the incoming traffic <= CIR<=UNI bandwidth, the traffic will be green class and enter network with assurance of service performance. If UNI>PIR>incoming traffic rate>CIR, one part is belonged to green class, and another is in yellow which also allowed to enter network. If incoming traffic even is higher than PIR, the traffic will be classified two part which green with CIR, yellow with PIR-CIR, and red simply drops.

Per-UNI Ingress bandwidth profile:

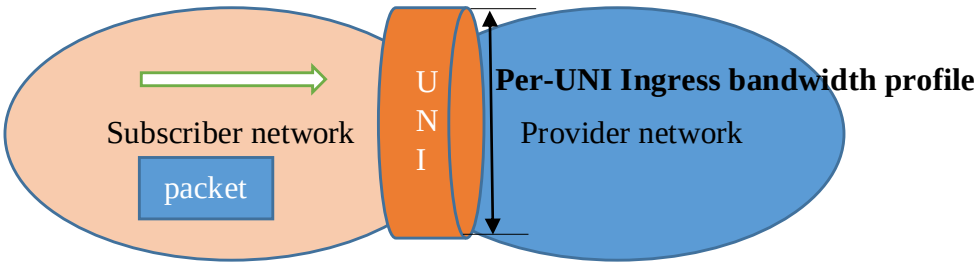


Figure 6-33

https://www.mef.net/Assets/White_Papers/Bandwidth-Profiles-for-Ethernet-Services.pdf

Per-EVC Ingress bandwidth profile is classifying flows to different EVCs. For instance, we create two access-list for each of EVC by TCP/UDP or IP addresses. The UNI is 100Mbps and includes two EVCs. The EVC1 is CIR=10Mbps, and PIR=100Mbps which is equal to UNI bandwidth. The EVC1 has 10Mbps guaranteed resource for packets to maintain service performance as in-profile, and excess rate will be colored yellow. The EVC2 has a CIR=30Mbps and PIR=100Mbps. The sum of all EVCs must be <= the actual UNI bandwidth. Two ways to implement this are rate-limit and CB-policing. An example of configuring CB-policing and assign it to ingress interface:

```
R1#sh access-lists
Extended IP access list 101
    10 permit tcp any any
Extended IP access list 102
    10 permit udp any any
R1#sh class-map
 Class Map match-any class-default (id 0)
   Match any

 Class Map match-all EVC1 (id 1)
   Match access-group  101

 Class Map match-all EVC2 (id 2)
   Match access-group  102

R1#sh policy-map
  Policy Map EVC1
    Class EVC1
     police cir 10000000 bc 312500 pir 100000000 be 3125000
       conform-action transmit
       exceed-action set-dscp-transmit default
       violate-action drop

  Policy Map EVC2
    Class EVC2
     police cir 30000000 bc 937500 pir 100000000 be 3125000
       conform-action transmit
       exceed-action set-dscp-transmit default
       violate-action drop
```

Figure 6-34

Per-EVC Ingress bandwidth profile:

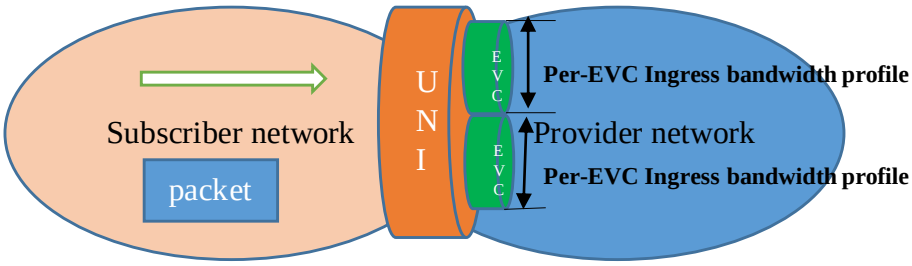


Figure 6-35

https://www.mef.net/Assets/White_Papers/Bandwidth-Profiles-for-Ethernet-Services.pdf

Per-DSCP Ingress bandwidth profile assign policing scheduler for each DSCP class, and the percentage of CIR allocation depends on the value of DSCP. The higher DSCP is, the more CIR will be allocated. An example of configuration is given as the

following: the UNI is 100Mbps speed.

```
R1#SH CLAss-map
 Class Map match-any EF (id 5)
   Match   dscp ef (46)

 Class Map match-any class-default (id 0)
   Match any

 Class Map match-any AF4X (id 4)
   Match   dscp af41 (34)
   Match   dscp af42 (36)
   Match   dscp af43 (38)

 Class Map match-any AF3X (id 3)
   Match   dscp af31 (26)
   Match   dscp af32 (28)
   Match   dscp af33 (30)

 Class Map match-any AF2X (id 2)
   Match   dscp af21 (18)
   Match   dscp af22 (20)
   Match   dscp af23 (22)

 Class Map match-any AF1X (id 1)
   Match   dscp af11 (10)
   Match   dscp af12 (12)
   Match   dscp af13 (14)

R1#sh policy-map
  Policy Map DSCP
    Class AF1X
     police cir 5000000 bc 156250 pir 100000000 be 3125000
       conform-action transmit
       exceed-action drop
       violate-action drop
    Class AF2X
     police cir 15000000 bc 468750 pir 100000000 be 3125000
       conform-action transmit
       exceed-action drop
       violate-action drop
    Class AF3X
     police cir 20000000 bc 625000 pir 100000000 be 3125000
       conform-action transmit
       exceed-action drop
       violate-action drop
    Class AF4X
     police cir 25000000 bc 781250 pir 100000000 be 3125000
       conform-action transmit
       exceed-action drop
       violate-action drop
```

```
    Class EF
     police cir 35000000 bc 1093750 pir 100000000 be 3125000
       conform-action transmit
       exceed-action drop
       violate-action drop
```

Figure 6-36

Per-DSCP Ingress bandwidth profile:

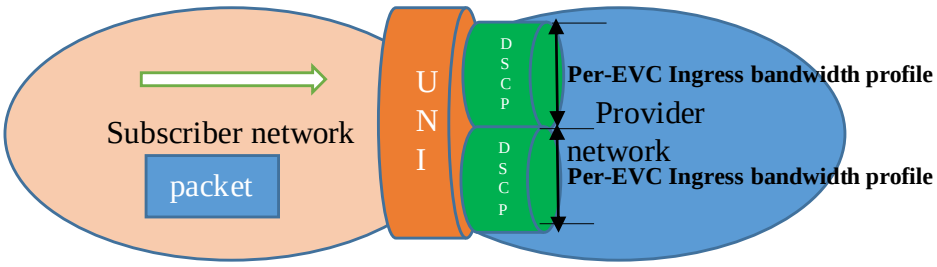


Figure 6-37

https://www.mef.net/Assets/White_Papers/Bandwidth-Profiles-for-Ethernet-Services.pdf

Egress policing is using same concept as ingress, which classifies bandwidth profile towards egress interface to limit traffic. Let us take a look at shaping specified on egress interface. Shaping is unlike policing which drop the non-conformant traffic, it will buffer the traffic into a queue temporarily. However, if the burst traffic remains a long term, if might cause increasing latency and packets loss. It also can be subdivided into three bandwidth profile by different type of pipes. Per-UNI Ingress bandwidth profile treat all service equally, Per-EVC Ingress bandwidth profile has a few EVCs to run various type of traffic, and finally Per-DSCP Ingress bandwidth profile is looking at DSCP value to determine how much rate need to be shaped by the importance of traffic. The shaping approach not only guarantees the minimum bandwidth for certain class but also provides an increment for the rate by peak command. Note that the shaping method can only be assign to egress interface. For example, two customers subscribe 50Mbps bandwidth profile connecting to a 100Mbps UNI by shaping method to limit the rate. By the formula (1+Be/Bc)*CIR, we set Be=Bc, so the shaping rate will be 2CIR. When one customer is not using the link, another cutomter can take over another 50Mbps and ISP can price the bandwidth incrementally. Whereas, they both occupy the link at the same time, the 50Mbps speed can be guaranteed, and also is flexible to take over another link's resource when the link is idle.

An example of configuring shaping:

```
R1#sh policy-map
  Policy Map shaping
    Class AF1X
      Peak Rate Traffic Shaping
      cir 5000000 (bps)
    Class AF2X
      Peak Rate Traffic Shaping
      cir 15000000 (bps)
    Class AF3X
      Peak Rate Traffic Shaping
      cir 20000000 (bps)
    Class AF4X
      Peak Rate Traffic Shaping
      cir 25000000 (bps)
    Class EF
      Peak Rate Traffic Shaping
      cir 35000000 (bps)
```

Figure 6-38

```
R1#sh policy-map interface
 FastEthernet0/0
  Service-policy output: shaping
    Class-map: AF1X (match-any)
      0 packets, 0 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match:  dscp af11 (10)
        0 packets, 0 bytes
        5 minute rate 0 bps
      Match:  dscp af12 (12)
        0 packets, 0 bytes
        5 minute rate 0 bps
      Match:  dscp af13 (14)
        0 packets, 0 bytes
        5 minute rate 0 bps
      Queueing
      queue limit 64 packets
      (queue depth/total drops/no-buffer drops) 0/0/0
      (pkts output/bytes output) 0/0
      shape (peak) cir 5000000, bc 20000, be 20000
      target shape rate 10000000
```

Class-map: AF2X (match-any)
0 packets, 0 bytes
5 minute offered rate 0 bps, drop rate 0 bps
Match: dscp af21 (18)
0 packets, 0 bytes
5 minute rate 0 bps
Match: dscp af22 (20)
0 packets, 0 bytes
5 minute rate 0 bps
Match: dscp af23 (22)
0 packets, 0 bytes
5 minute rate 0 bps
Queueing
queue limit 64 packets
(queue depth/total drops/no-buffer drops) 0/0/0
(pkts output/bytes output) 0/0
shape (peak) cir 15000000, bc 60000, be 60000
target shape rate 30000000

Class-map: AF3X (match-any)
0 packets, 0 bytes
5 minute offered rate 0 bps, drop rate 0 bps
Match: dscp af31 (26)
0 packets, 0 bytes
5 minute rate 0 bps
Match: dscp af32 (28)
0 packets, 0 bytes
5 minute rate 0 bps
Match: dscp af33 (30)
0 packets, 0 bytes
5 minute rate 0 bps
Queueing
queue limit 64 packets
(queue depth/total drops/no-buffer drops) 0/0/0
(pkts output/bytes output) 0/0
shape (peak) cir 20000000, bc 80000, be 80000
target shape rate 40000000

Class-map: AF4X (match-any)
0 packets, 0 bytes
5 minute offered rate 0 bps, drop rate 0 bps
Match: dscp af41 (34)
0 packets, 0 bytes
5 minute rate 0 bps
Match: dscp af42 (36)
0 packets, 0 bytes
5 minute rate 0 bps
Match: dscp af43 (38)
0 packets, 0 bytes
5 minute rate 0 bps
Queueing
queue limit 64 packets
(queue depth/total drops/no-buffer drops) 0/0/0
(pkts output/bytes output) 0/0
shape (peak) cir 25000000, bc 100000, be 100000
target shape rate 50000000

Class-map: EF (match-any)
0 packets, 0 bytes
5 minute offered rate 0 bps, drop rate 0 bps
Match: dscp ef (46)
0 packets, 0 bytes
5 minute rate 0 bps
Queueing
queue limit 64 packets
(queue depth/total drops/no-buffer drops) 0/0/0
(pkts output/bytes output) 0/0
shape (peak) cir 35000000, bc 140000, be 140000
target shape rate 70000000

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

queue limit 64 packets

(queue depth/total drops/no-buffer drops) 0/0/0

(pkts output/bytes output) 0/0

Now we take a look at combination of shaping and policing as the following scenario show the link between PE1 and PE2 is Ethernet interface 10Mbps, which means the maximum capability is 10Mbps. Hypothetically, ISP sells 10Mbps to each of customers and only 5Mbps can be guaranteed, and the time interval assumes to be 100ms. Where the 10Mbps link is causes the over-subscription. CE1 and CE2 implement shaping of peak on their egress interface, and policing will be implemented on ingress interface with CIR 5Mbps and PIR 10Mbps. In addition, CB-WFQ will be introduced for queueing between different class of flows.

The scenario:

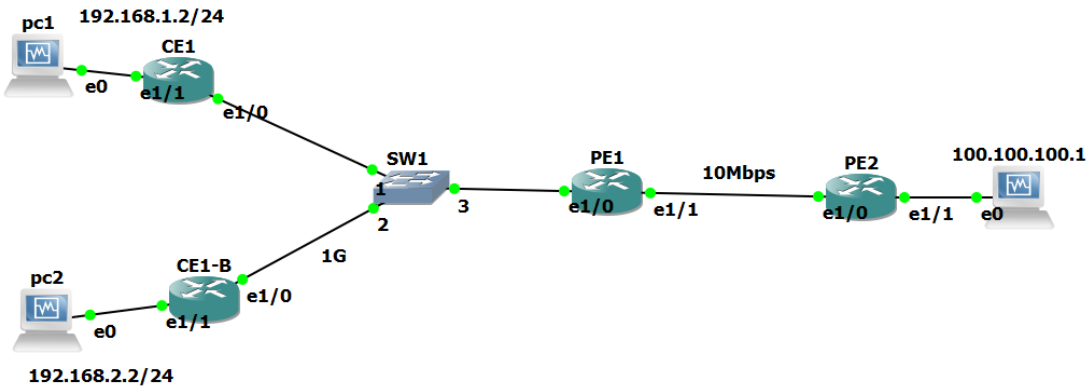


Figure 6-39

The configuration on each router:

CE1#sh run

class-map match-all TCP

match access-group 101

policy-map shaping

class TCP

shape peak 5000000 800000 800000

interface Ethernet1/0

service-policy output shaping

CE1-B#sh run

hostname CE1-B

class-map match-all TCP

match access-group 101

```
policy-map shaping
 class TCP
  shape peak 5000000 800000 800000
interface Ethernet1/0
service-policy output shaping
PE1(config-if)#do sh run
class-map match-all p0
 match ip precedence 0
class-map match-all p1
 match ip precedence 0
class-map match-all fromR1
 match access-group name fromr1
class-map match-all fromR2
 match access-group name fromr2
policy-map policing
 class fromR1
  police cir 5000000 pir 10000000
   conform-action set-prec-transmit 1
   exceed-action set-prec-transmit 0
   violate-action drop
 class fromR2
  police cir 5000000 pir 10000000
   conform-action set-prec-transmit 1
   exceed-action set-prec-transmit 0
   violate-action drop
policy-map CB-WFQ
 class p0
  bandwidth percent 5
 class p1
  bandwidth percent 20
interface Ethernet1/0
 load-interval 30
service-policy input policing
 service-policy output CB-WFQ
ip access-list extended fromr1
 permit tcp host 192.168.1.2 any
ip access-list extended fromr2
```

permit tcp host 192.168.2.2 any

set the conformed traffic as IPP1 and excess as IPP0, it is for later CB-WFQ to schedule packet depending on their weight.

1st Now we run 10Mbps traffic from pc1 to remote, and show the policy-map on CE1 and PE1:

```
CE1#sh policy-map interface
 Ethernet1/0

  Service-policy output: shaping

    Class-map: TCP (match-all)
      28265 packets, 42264036 bytes
      5 minute offered rate 986000 bps, drop rate 0 bps
      Match: access-group 101
      Queueing
      queue limit 64 packets
      (queue depth/total drops/no-buffer drops) 0/124/0
      (pkts output/bytes output) 28164/42112145
      shape (peak) cir 5000000, bc 800000, be 800000
      target shape rate 10000000

    Class-map: class-default (match-any)
      5 packets, 300 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: any

      queue limit 64 packets
      (queue depth/total drops/no-buffer drops) 0/0/0
      (pkts output/bytes output) 5/300
```

Figure 6-40

The traffic is going through egress interface and getting shaping with 10Mbps outgoing traffic rate.

```
PE1#sh policy-map interface
 Ethernet1/0

  Service-policy input: policing

    Class-map: fromR1 (match-all)
      139461 packets, 209213485 bytes
      30 second offered rate 7318000 bps, drop rate 0 bps
      Match: access-group name fromr1
      police:
          cir 5000000 bps, bc 156250 bytes
          pir 10000000 bps, be 312500 bytes
        conformed 84153 packets, 126199198 bytes; actions:
          set-prec-transmit 1
        exceeded 55308 packets, 83014287 bytes; actions:
          set-prec-transmit 0
        violated 0 packets, 0 bytes; actions:
          drop
        conformed 4859000 bps, exceed 2466000 bps, violate 0 bps
```

Figure 6-41

In 30s e1/0 conform almost 5Mbps traffic when the rate exceeds the configured CIR (green), and rest of them belong to excess (yellow).

The throughput is displayed as below:

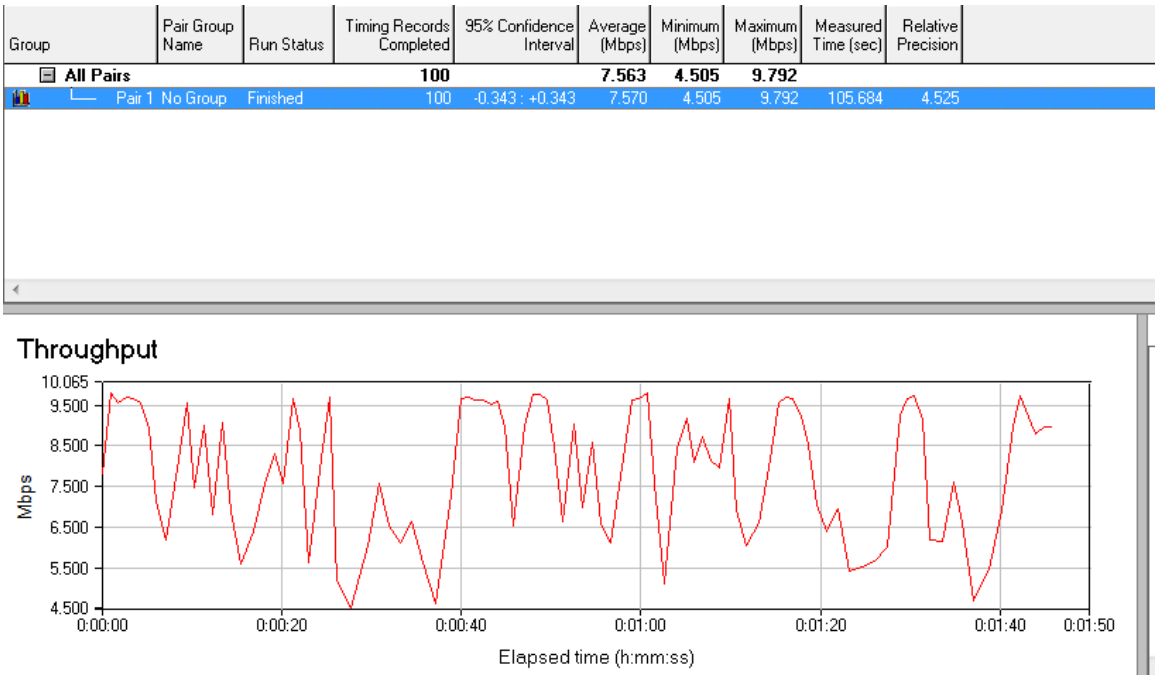


Figure 6-42

2nd generating traffic from pc1 and pc2 simultaneously with both 10Mbps, and see how the shaping and policing varying.

PE1:

```
PE1#sh policy-map interface
 Ethernet1/0

  Service-policy input: policing

    Class-map: fromR1 (match-all)
      245540 packets, 370767032 bytes
      30 second offered rate 5237000 bps, drop rate 0 bps
      Match: access-group name fromr1
      police:
          cir 5000000 bps, bc 156250 bytes
          pir 10000000 bps, be 312500 bytes
        conformed 150412 packets, 226916542 bytes; actions:
          set-prec-transmit 1
        exceeded 95128 packets, 143850490 bytes; actions:
          set-prec-transmit 0
        violated 0 packets, 0 bytes; actions:
          drop
        conformed 4171000 bps, exceed 1064000 bps, violate 0 bps

    Class-map: fromR2 (match-all)
      240759 packets, 363303449 bytes
      30 second offered rate 5658000 bps, drop rate 0 bps
      Match: access-group name fromr2
      police:
          cir 5000000 bps, bc 156250 bytes
          pir 10000000 bps, be 312500 bytes
        conformed 152798 packets, 230267990 bytes; actions:
          set-prec-transmit 1
        exceeded 87961 packets, 133035459 bytes; actions:
          set-prec-transmit 0
        violated 0 packets, 0 bytes; actions:
          drop
        conformed 4648000 bps, exceed 1012000 bps, violate 0 bps

    Class-map: class-default (match-any)
      0 packets, 0 bytes
      30 second offered rate 0 bps, drop rate 0 bps
      Match: any
```

Figure 6-43

When over-subscription is taking place, the policy helps to ensure the CIR as conformed

CE1:

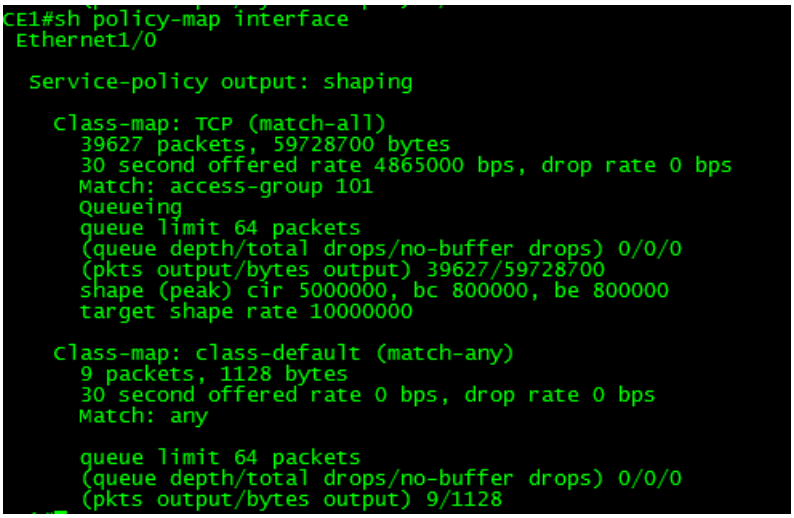


Figure 6-44

CE1-B:

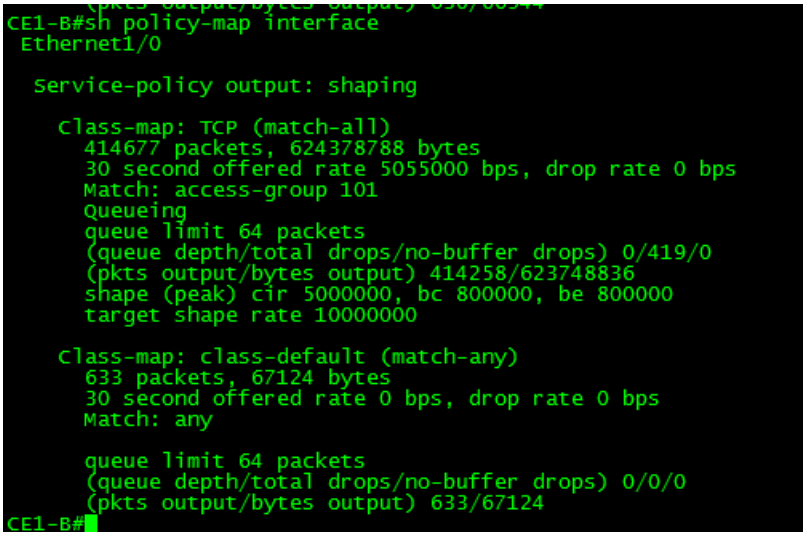


Figure 6-45

When the over-subscription is happening on the link, the shaping facilitates to limit the traffic to SLA rate to guarantee the service performance for each of customers.

The CB-WFQ:

```
      Match: any
Ethernet1/1

 Service-policy output: CB-WFQ

   Class-map: p0 (match-all)
     490 packets, 738847 bytes
     5 minute offered rate 0 bps, drop rate 0 bps
     Match: ip precedence 0
     Queueing
     queue limit 64 packets
     (queue depth/total drops/no-buffer drops) 0/0/0
     (pkts output/bytes output) 493/743389
     bandwidth 5% (500 kbps)

   Class-map: p1 (match-all)
     5041 packets, 7613109 bytes
     5 minute offered rate 0 bps, drop rate 0 bps
     Match: ip precedence 1
     Queueing
     queue limit 64 packets
     (queue depth/total drops/no-buffer drops) 13/0/0
     (pkts output/bytes output) 5057/7637333
     bandwidth 20% (2000 kbps)

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

Figure 6-46

PC1 throughput:

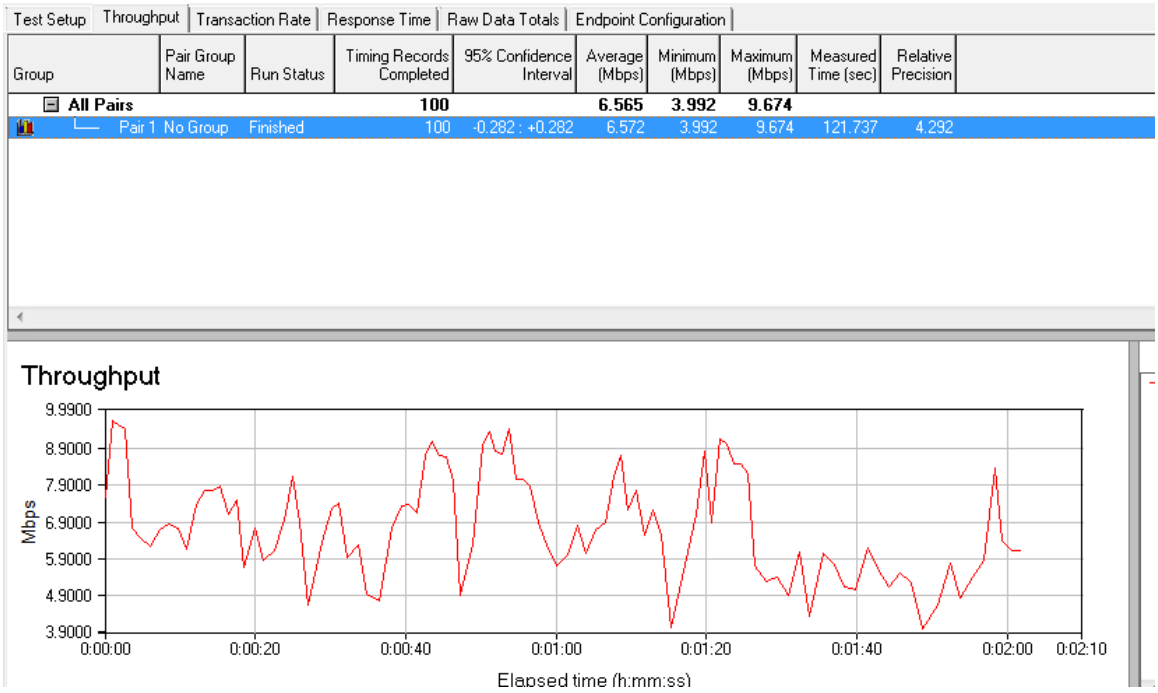


Figure 6-47

PC2 throughput:

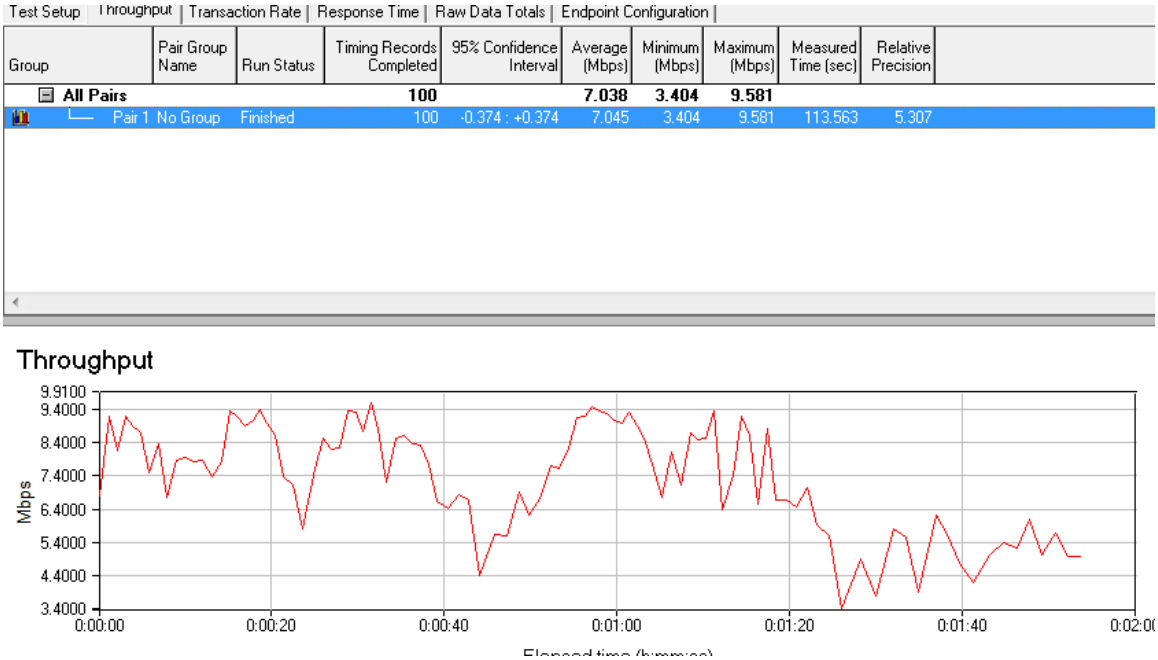


Figure 6-48

Their relation is trading off and taking turns.

The small testament shows us how the policing and shaping cooperate with each other to make sure the green class service performance when over-subscription takes place. Also, it helps to take over idle resource to be used.

7. MPLS QoS

MPLS (Multiprotocol Label Switching) is using label switching to take place of the conventional routing to forward packets. It has powerful routing function, flexible, fast processing, and satisfied to various of new network requirement, and scalable to some network protocols such as IPv6, IPX, and so on. MPLS saves a lot of resource and CPU consuming in routers. Nowadays, this technique is widely applied to ISP's core network and a huge network infrastructure, so QoS applying to MPLS must be considered.

For MPLS network, QoS is the same as IP network where has to consider serval parameter to maintain certain service quality including propagation rate, delay, jitter, packet loss, and so on. According to network controlling diverse application traffic differently, QoS service also can classify three categories: BE, DiffServ, and IntServ. Different MPLS service is in accordance with EXP value in MPLS header to determine PHB implementation. The IntServ of MPLS network is utilizing MPLS-TE technique to be performed.

7.1 MPLS DiffServ

The mechanism of DiffServ for MPLS is same as IP DiffServ, which the traffic enter a configured area starts to be classified by its regulation. In MPLS network, it forwards packets depending on label in MPLS header and doesn't check IP header, so it is not able to use IPP/DSCP value to classify traffic. The services are mapped to certain class according to the service characteristics and requirements at MPLS network edge, and then each of node in core network checks and treats traffic via recognizing the 3-bit EXP which is configured by service policy and to guarantee their quality of service. 3-bit EXP carries DiffServ PHB and label switching router takes a decision by considering MPLS EXP. However, DiffServ PHB is maximum up to 64 code values (2^6), so how 3-bit EXP that is maximum 8 classes (2^3) can handle that many values. There are two solutions for determine PHB:

The first method E-LSP, which EXP decide PHB and LSP (label switching path): It applies to less than 8 PHBs network, and certain DSCPs directly map to certain EXP to specify PHB. On the way of propagating packets, LSP decide the path of forwarding; meanwhile, EXP will determine queueing and scheduling on each of LSR (label switching router). Therefore, one LSP can carry 8 different PHBs with using EXP to differentiate. EXP can either copy DSCP from IP header or be configured by ISP.

The second approach L-LSP supports more than 8 PHBs network. In here, EXP is not sufficient to carry various of PHB information, so it requests label and EXP working together to decide PHB and LSP. In the procedure of forwarding, labels not only work for determining forwarding path but also assign scheduling behaviours, and EXP is carrying dropping precedence. Thus, they both cooperate with each other to conclude PHB. Due to existing relation between label and PHB, so when establishing

LSP, it need to transmit this information. In consequence, the LSP that needs label to propagate PHB to establish called L-LSP.

After using these approaches to classify traffic, QoS mechanisms will be the same as IP network such as shaping, policing, and congestion management and avoidance.

The table shows the default mapping between DiffServ PHB and EXP:

DSCP PHB	EXP
CS7 (111000)	111 7
CS6 (110000)	110 6
EF (101110)	101 5
AF4X (100XX0)	100 4
AF3X (011XX0)	011 3
AF2X(010XX0)	010 2
AF(001XX0)	001 1
Best Effort	000 0

Table 7-1

The illustration explains how DiffServ and EXP exchange between each other:

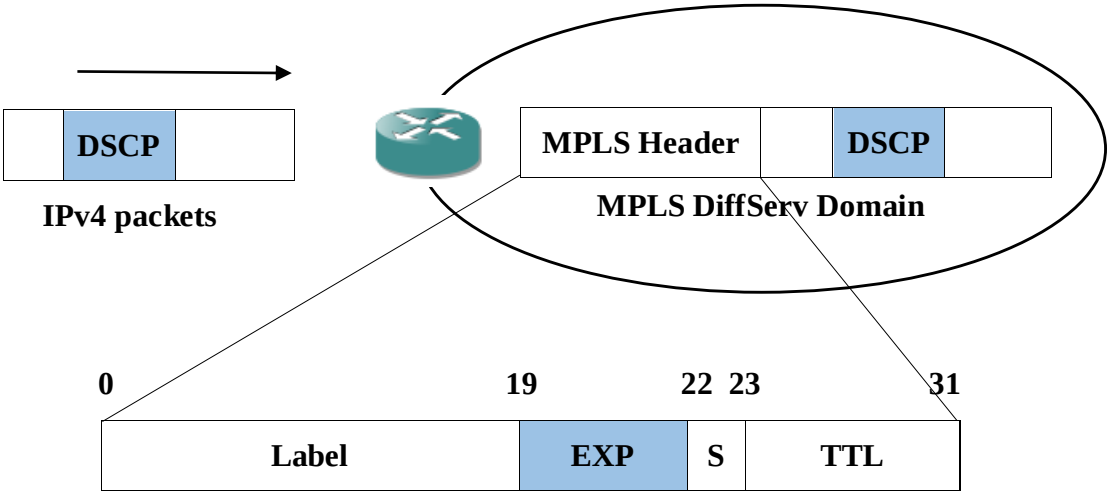


Figure 7-2

http://h3c.com/portal/Products___Solutions/Products/Switches/H3C_S5120-EI_Series_Switches/White_Paper/200812/689004_57_0.htm#_Toc215923183

As shown above illustration, at the edge of MPLS, the IP packets directly copy DSCP to MPLS header of EXP by default. However, if ISP doesn't trust any incoming data or might request to define different DiffServ, so EXP or DSCP in IP header can be changed according to scheduling policy along the path.

7.2 MPLS DiffServ tunneling

As MPLS VPNs are rapidly getting popular over private WAN to provide ISP core network routing and switching with their customers, and supply scalable VPNs and faster packets forwarding with end-to-end quality of service. MPLS VPN mainly consists of CE, PE, and P. Customer Edge Router: directly connected ISP edge router which is not perceptive the existence of VPN. Provider Edge Router: responsible for VPN entrance and cope with VPN-IPv4 routing. Provider Router: expedited forwarding data. The basic architecture of MPLS VPN network is displayed as below:

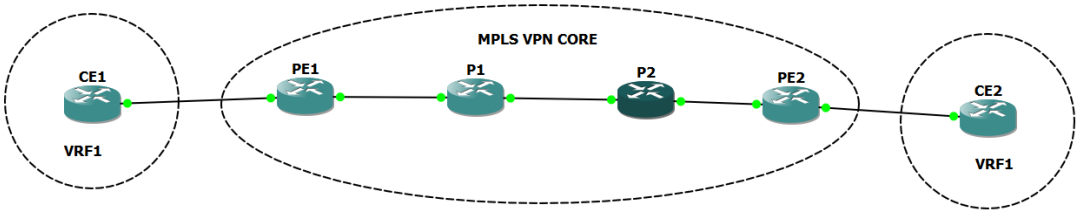


Figure 7-3

http://www.cisco.com/c/en/us/support/docs/multiprotocol-label-switching-mpls/mpls/47815-diffserv-tunnel.html

Note: 1. Enabling ip cef

2. creating IGP like OSPF across all ISP routers and IBGP between PE1 and PE2

3. creating VRF (virtual routing and forward) and assign it to ingress interface

4. enabling mpls ip on the interface which connects to ISP routers

5. CE1 and CE2 belong to their own Autonomous system, and redistribute their routing entries to IBGP and IBGP routing entries to the pointed AS.

Now the network behind CE1 and CE2 is able to ping each other. We will continue to implement three different tunneling models based on this scenario, and analyze how the EXP and DSCP swaps while traffic traveling through MPLS core network from outside of it.

Actually, MPLS network essentially provide a tunnel to carry these DiffServ, and there are three MPLS DiffServ tunneling modes which defined in RFC3270. They are including Uniform, Short-Pipe, and Pipe mode. By default, a packet enters the MPLS network copying and mapping the DSCP value from IP header to EXP bits of all pushed labels, and then carry this value all the way along the path. The packet leaves

the MPLS domain with the same DSCP as it incoming to MPLS domain. Different tunneling modes are introduced to allow differentiate QoS in MPLS network.

7.2.1 Uniform mode

Uniform mode is keeping the marking information uniform and having one layer of QoS. The ingress PE router copies the DSCP value from incoming IP packet to MPLS EXP bits of imposed label. When the packet is crossing the MPLS core network through P routers, the EXP might be change or not. If the EXP of topmost label get changed, it will copy to the newly exposed label after label popped. Also, the egress PE router will copy the EXP value to exposed DSCP in IP packet.

The illustration of EXP and DSCP swapping shows:

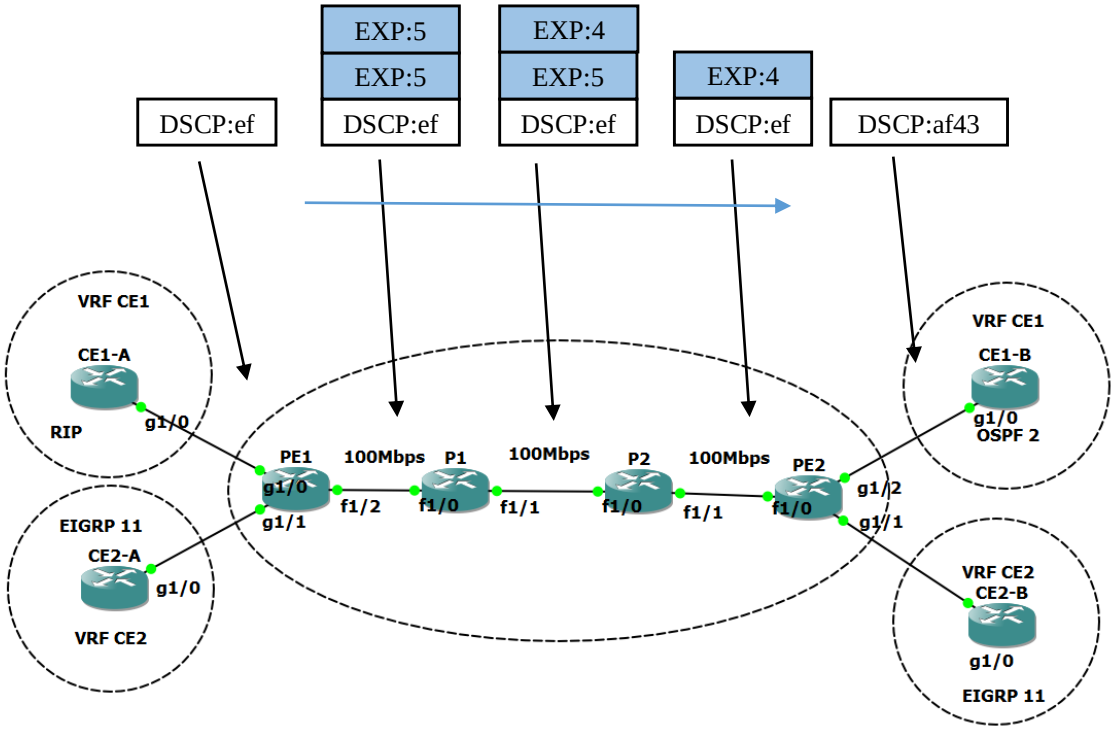


Figure 7-4

http://www.cisco.com/c/en/us/support/docs/multiprotocol-label-switching-mpls/mpls/47815-diffserv-tunnel.html

For example, the traffic is generating behind CE routers towards PE1, and we will see how to complete the EXP and DSCP swapping as the illustration.

In addition, the bandwidth between PE1 and P1 is 1Gbps, and because of interface on, which means the capability of transmission is 100Mbps in total. If the ISP sell 100Mbps to each of customers behind each CE routers, it will cause oversubscription and occurrence of congestion. For example, if ISP sells 100Mbps respectively to two customers behind CE1-A router, it might not get the bandwidth which the customer wants originally when they generate traffic simultaneously. The purchased bandwidth

might loss its quality when congestion occurs. Therefore, it requires policing and shaping cooperating with CE router and PE router to implement oversubscription and provide minimum bandwidth insurance. Furthermore, we implement CB-WFQ and WRED on each provider routers in case the congestion occurs. The detail configuration will be displayed later.

Behind CE1-A network: 5.1.1.1/32, 6.1.1.1/32, 7.1.1.1/32

Behind CE2-A network: 11.1.1.1/32, 12.1.1.1/32, 13.1.1.1/32

Behind CE1-B network: 8.1.1.1/32, 9.1.1.1/32, 10.1.1.1/32

Behind CE2-B network: 14.1.1.1/32, 15.1.1.1/32, 16.1.1.1/32

1st uniform mode configurations of each router:

```
CE1-A-WEI#sh run
hostname CE1-A-WEI
ip cef
class-map match-all VOICE
 match  dscp ef
class-map match-all VIDEOCON
 match  dscp af41
class-map match-all CRITICALDATA
 match  dscp af31
class-map match-all CALLSIGNAL
 match  dscp cs3
class-map match-all VIDEOSTR
 match  dscp cs4
class-map match-all NETMAN
 match  dscp cs2
class-map match-all BULKDATA
 match  dscp af11
class-map match-all SCAVEN
 match  dscp cs1
class-map match-all BESTEFF
 match  dscp 0
policy-map shaping
 class VOICE
  shape peak 50000000 5000000 5000000
 class VIDEOCON
  shape peak 50000000 5000000 5000000
 class CRITICALDATA
  shape peak 50000000 5000000 5000000
 class CALLSIGNAL
  shape peak 50000000 5000000 5000000
 class VIDEOSTR
  shape peak 50000000 5000000 5000000
class NETMAN
  shape peak 50000000 5000000 5000000
class BULKDATA
  shape peak 50000000 5000000 5000000
class BESTEFF
  shape peak 50000000 5000000 5000000
interface Loopback1
 ip address 5.1.1.1 255.255.255.255
!
interface Loopback2
 ip address 6.1.1.1 255.255.255.255
interface Loopback3
 ip address 7.1.1.1 255.255.255.255
interface GigabitEthernet1/0
 ip address 192.168.1.1 255.255.255.192
 duplex full
 service-policy output shaping
router rip
```

```
 version 2
 network 5.0.0.0
 network 6.0.0.0
 network 7.0.0.0
 network 192.168.1.0
 no auto-summary
CE2-A-WEI#sh run
hostname CE2-A-WEI
ip cef
class-map match-all VOICE
 match  dscp ef
class-map match-all VIDEOCON
 match  dscp af41
class-map match-all CRITICALDATA
 match  dscp af31
class-map match-all CALLSIGNAL
 match  dscp cs3
class-map match-all VIDEOSTR
 match  dscp cs4
class-map match-all NETMAN
 match  dscp cs2
class-map match-all BULKDATA
 match  dscp af11
class-map match-all SCAVEN
 match  dscp cs1
class-map match-all BESTEFF
 match  dscp 0
policy-map shaping
 class VOICE
  shape peak 50000000 5000000 5000000
 class VIDEOCON
  shape peak 50000000 5000000 5000000
 class CRITICALDATA
  shape peak 50000000 5000000 5000000
 class CALLSIGNAL
  shape peak 50000000 5000000 5000000
 class VIDEOSTR
  shape peak 50000000 5000000 5000000
class NETMAN
  shape peak 50000000 5000000 5000000
class BULKDATA
  shape peak 50000000 5000000 5000000
class BESTEFF
  shape peak 50000000 5000000 5000000
interface Loopback1
 ip address 11.1.1.1 255.255.255.255
interface Loopback2
 ip address 12.1.1.1 255.255.255.255
interface Loopback3
 ip address 13.1.1.1 255.255.255.255
interface GigabitEthernet1/0
 ip address 192.168.3.1 255.255.255.0
 duplex full
 service-policy output shaping
router eigrp 11
 network 11.1.1.1 0.0.0.0
 network 12.1.1.1 0.0.0.0
 network 13.1.1.1 0.0.0.0
 network 192.168.3.0
PE1-WEI#sh run
hostname PE1-WEI
ip cef
ip vrf CE1
 rd 1:1
 route-target export 123:1
 route-target import 234:1
 route-target import 567:1
 route-target import 789:1
ip vrf CE2
 rd 1:2
```

```
 route-target export 789:1
 route-target import 567:1
 route-target import 123:1
 route-target import 234:1
mpls label range 100 200
mpls label protocol ldp
class-map match-all MPLS-5
 match mpls experimental topmost 5
class-map match-all MPLS-0
 match mpls experimental topmost 0
class-map match-all MPLS-1
 match mpls experimental topmost 1
class-map match-all MPLS-2
 match mpls experimental topmost 2
class-map match-all MPLS-3
 match mpls experimental topmost 3
class-map match-all MPLS-4
 match mpls experimental topmost 4
class-map match-any VOICE
 match  dscp ef
class-map match-any VIDEODA
 match  dscp af41
class-map match-any CD&CS
 match  dscp af31
 match  dscp cs3
class-map match-any NM
 match  dscp cs2
class-map match-any BD&SCA
 match  dscp af11
 match  dscp cs1
class-map match-any BESTEFF
 match  dscp 0
policy-map outbound
 class MPLS-5
  priority percent 25
class MPLS-0
  bandwidth percent 3
  random-detect
 class MPLS-1
  bandwidth percent 5
  random-detect
 class MPLS-2
  bandwidth percent 20
  random-detect
 class MPLS-3
  bandwidth percent 15
  random-detect
 class MPLS-4
  priority percent 20
policy-map set-MPLS-EXP
 class BESTEFF
  police cir 2000000 pir 50000000
   conform-action set-mpls-exp-imposition-
transmit 0
   exceed-action set-mpls-exp-imposition-
transmit 0
   violate-action drop
class BD&SCA
  police cir 4000000 pir 50000000
   conform-action set-mpls-exp-imposition-
transmit 1
   exceed-action set-mpls-exp-imposition-
transmit 0
   violate-action drop
 class VIDEODA
  police cir 15000000 pir 50000000
   conform-action set-mpls-exp-imposition-
transmit 4
   exceed-action set-mpls-exp-imposition-
transmit 3
   violate-action drop
 class CD&CS
```

```
 police cir 5000000 pir 50000000
  conform-action set-mpls-exp-imposition-transmit 3
  exceed-action set-mpls-exp-imposition-transmit 2
  violate-action drop
 class NM
  police cir 9000000 pir 50000000
  conform-action set-mpls-exp-imposition-transmit 2
  exceed-action set-mpls-exp-imposition-transmit 1
  violate-action drop
 class VOICE
  police cir 15000000 pir 50000000
  conform-action set-mpls-exp-imposition-transmit 5
  exceed-action set-mpls-exp-imposition-transmit 4
  violate-action drop
interface Loopback1
 ip address 1.1.1.1 255.255.255.255
interface GigabitEthernet1/0
 ip vrf forwarding CE1
 ip address 192.168.1.2 255.255.255.0
 duplex full
 service-policy input set-MPLS-EXP
interface GigabitEthernet1/1
 ip vrf forwarding CE2
 ip address 192.168.3.2 255.255.255.0
 duplex full
interface FastEthernet1/2
 ip address 10.1.11.1 255.255.255.252
 duplex full
 mpls ldp discovery transport-address interface
 mpls ip
 service-policy output outbound
router eigrp 11
address-family ipv4 vrf CE2 autonomous-system 11
  redistribute bgp 11 metric 15000 1 255 255 65535
  network 192.168.3.0
 exit-address-family
router ospf 1
 router-id 1.1.1.1
 network 1.1.1.1 0.0.0.0 area 1
 network 10.1.11.0 0.0.0.3 area 1
router rip
address-family ipv4 vrf CE1
  redistribute bgp 11 metric 5
  network 192.168.1.0
  no auto-summary
  version 2
 exit-address-family
router bgp 11
 bgp log-neighbor-changes
 neighbor 2.2.2.2 remote-as 11
 neighbor 2.2.2.2 update-source Loopback1
 !
 address-family vpnv4
  neighbor 2.2.2.2 activate
  neighbor 2.2.2.2 send-community extended
 exit-address-family
 !
 address-family ipv4 vrf CE1
  redistribute rip
 exit-address-family
 !
 address-family ipv4 vrf CE2
  redistribute eigrp 11
 exit-address-family
```

```
mpls ldp router-id Loopback1
P1-WEI#sh run
hostname P1-WEI
ip cef
mpls label range 200 300
mpls label protocol ldp
class-map match-all MPLS-IN
 match mpls experimental topmost 5
policy-map MPLS-IN
 class MPLS-IN
  set mpls experimental topmost 4
interface Loopback1
 ip address 3.3.3.3 255.255.255.255
interface FastEthernet1/0
 ip address 10.1.11.2 255.255.255.252
 duplex full
 mpls ldp discovery transport-address
interface
 mpls ip
 service-policy input MPLS-IN
interface FastEthernet1/1
 ip address 10.1.11.5 255.255.255.252
 duplex full
 mpls ldp discovery transport-address
interface
 mpls ip
router ospf 1
 router-id 3.3.3.3
 network 10.1.11.0 0.0.0.3 area 1
 network 10.1.11.4 0.0.0.3 area 1
mpls ldp router-id Loopback1
P2-WEI#sh run
hostname P2-WEI
ip cef
mpls label range 300 400
mpls label protocol ldp
class-map match-all qos-group-5
 match qos-group 5
class-map match-all MPLS-5
 match mpls experimental topmost 5
class-map match-all MPLS-0
 match mpls experimental topmost 0
class-map match-all MPLS-1
 match mpls experimental topmost 1
class-map match-all MPLS-2
 match mpls experimental topmost 2
class-map match-all MPLS-3
 match mpls experimental topmost 3
class-map match-all MPLS-4
 match mpls experimental topmost 4
class-map match-all qos-group-4
 match qos-group 4
class-map match-all qos-group-1
 match qos-group 1
class-map match-all qos-group-3
 match qos-group 3
class-map match-all qos-group-2
 match qos-group 2
class-map match-all qos-group-0
 match qos-group 0
policy-map qos-group-in
 class MPLS-5
  set qos-group mpls experimental topmost
 class MPLS-4
  set qos-group mpls experimental topmost
 class MPLS-3
  set qos-group mpls experimental topmost
 class MPLS-2
  set qos-group mpls experimental topmost
 class MPLS-1
  set qos-group mpls experimental topmost
```

```
class MPLS-0
 set qos-group mpls experimental topmost
policy-map qos-group-out
 class qos-group-0
  bandwidth percent 3
  random-detect
 class qos-group-1
  bandwidth percent 5
  random-detect
  set mpls experimental topmost qos-group
 class qos-group-2
  bandwidth percent 20
  random-detect
  set mpls experimental topmost qos-group
 class qos-group-3
  bandwidth percent 15
  random-detect
  set mpls experimental topmost qos-group
 class qos-group-4
  priority percent 20
  set mpls experimental topmost qos-group
 class qos-group-5
  priority percent 25
  set mpls experimental topmost qos-group
interface Loopback1
 ip address 4.4.4.4 255.255.255.255
interface FastEthernet1/0
 ip address 10.1.11.6 255.255.255.252
 duplex full
 mpls ldp discovery transport-address
interface
 mpls ip
 service-policy input qos-group-in
interface FastEthernet1/1
 ip address 10.1.11.9 255.255.255.252
 duplex full
 mpls ldp discovery transport-address
interface
 mpls ip
 service-policy output qos-group-out
router ospf 1
 router-id 4.4.4.4
 network 10.1.11.4 0.0.0.3 area 1
 network 10.1.11.8 0.0.0.3 area 1
mpls ldp router-id Loopback1
PE2-WEI#sh run
hostname PE2-WEI
ip cef
ip vrf CE1
 rd 1:1
 route-target export 234:1
 route-target import 123:1
 route-target import 567:1
 route-target import 789:1
ip vrf CE2
 rd 1:2
 route-target export 567:1
 route-target import 789:1
 route-target import 123:1
 route-target import 234:1
mpls label range 400 500
mpls label protocol ldp
class-map match-all qos-group-5
 match qos-group 5
class-map match-all MPLS-5
 match mpls experimental topmost 5
class-map match-all MPLS-0
 match mpls experimental topmost 0
class-map match-all MPLS-1
 match mpls experimental topmost 1
```

```
class-map match-all MPLS-2
 match mpls experimental topmost 2
class-map match-all MPLS-3
 match mpls experimental topmost 3
class-map match-all MPLS-4
 match mpls experimental topmost 4
class-map match-all qos-group-4
 match qos-group 4
class-map match-all qos-group-1
 match qos-group 1
class-map match-all qos-group-3
 match qos-group 3
class-map match-all qos-group-2
 match qos-group 2
class-map match-all qos-group-0
 match qos-group 0
policy-map qos-group-in
 class MPLS-5
  set qos-group mpls experimental topmost
 class MPLS-4
  set qos-group mpls experimental topmost
 class MPLS-3
  set qos-group mpls experimental topmost
 class MPLS-2
  set qos-group mpls experimental topmost
 class MPLS-1
  set qos-group mpls experimental topmost
class MPLS-0
  set qos-group mpls experimental topmost
policy-map qos-group-out
class qos-group-0
  set precedence qos-group
  bandwidth percent 3
  random-detect
class qos-group-1
  set precedence qos-group
  bandwidth percent 5
  random-detect
class qos-group-2
  set precedence qos-group
  bandwidth percent 20
  random-detect
class qos-group-3
  set precedence qos-group
  bandwidth percent 15
  random-detect
class qos-group-4
  set precedence qos-group
  priority percent 20
class qos-group-5
  set precedence qos-group
  priority percent 25
interface Loopback1
 ip address 2.2.2.2 255.255.255.255
interface FastEthernet1/0
 ip address 10.1.11.10 255.255.255.252
 duplex full
 mpls ldp discovery transport-address
interface
 mpls ip
 service-policy input qos-group-in
interface GigabitEthernet1/1
 bandwidth 100000
 ip vrf forwarding CE2
 ip address 192.168.4.1 255.255.255.0
 duplex full
 service-policy output qos-group-out
interface GigabitEthernet1/2
 ip vrf forwarding CE1
 ip address 192.168.2.1 255.255.255.0
```

```
 duplex full
router eigrp 11
address-family ipv4 vrf CE2 autonomous-system 11
  redistribute bgp 11 metric 15000 1 255 255 65535
  network 192.168.4.0
 exit-address-family
router ospf 2 vrf CE1
 redistribute bgp 11 subnets
 network 192.168.2.0 0.0.0.255 area 2
router ospf 1
 router-id 2.2.2.2
 network 2.2.2.2 0.0.0.0 area 1
 network 10.1.11.8 0.0.0.3 area 1
router bgp 11
 bgp log-neighbor-changes
 neighbor 1.1.1.1 remote-as 11
 neighbor 1.1.1.1 update-source Loopback1
address-family vpnv4
  neighbor 1.1.1.1 activate
  neighbor 1.1.1.1 send-community extended
 exit-address-family
address-family ipv4 vrf CE1
  redistribute ospf 2
 exit-address-family
address-family ipv4 vrf CE2
  redistribute eigrp 11
 exit-address-family
mpls ldp router-id Loopback1
CE2-B-WEI#sh run
hostname CE2-B-WEI
ip cef
interface Loopback1
 ip address 14.1.1.1 255.255.255.255
interface Loopback2
 ip address 15.1.1.1 255.255.255.255
interface Loopback3
 ip address 16.1.1.1 255.255.255.255
interface GigabitEthernet1/0
 ip address 192.168.4.2 255.255.255.0
 duplex full
router eigrp 11
 network 14.1.1.1 0.0.0.0
 network 15.1.1.1 0.0.0.0
 network 16.1.1.1 0.0.0.0
 network 192.168.4.0
CE1-B-WEI# sh run
hostname CE1-B-WEI
ip cef
interface Loopback1
 ip address 8.1.1.1 255.255.255.255
!
interface Loopback2
 ip address 9.1.1.1 255.255.255.255
!
interface Loopback3
 ip address 10.1.1.1 255.255.255.255
interface GigabitEthernet1/0
 ip address 192.168.2.2 255.255.255.0
 duplex full
router ospf 2
 network 8.1.1.1 0.0.0.0 area 2
 network 9.1.1.1 0.0.0.0 area 2
 network 10.1.1.1 0.0.0.0 area 2
 network 192.168.2.0 0.0.0.255 area 2
```

2nd test the connectivity from CE1 routers to CE2 routers

```
CE1-A-WEI#ping 8.1.1.1 so
CE1-A-WEI#ping 8.1.1.1 source 5.1.1.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 8.1.1.1, timeout is 2 seconds:
Packet sent with a source address of 5.1.1.1
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 116/151/184 ms
CE1-A-WEI#ping 16.1.1.1 so
CE1-A-WEI#ping 16.1.1.1 source 5.1.1.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 16.1.1.1, timeout is 2 seconds:
Packet sent with a source address of 5.1.1.1
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 128/149/192 ms
CE1-A-WEI#
```

Figure 7-5

Show ip route on CE1-A:

```
CE1-A-WEI#sh ip rou
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
       + - replicated route, % - next hop override

Gateway of last resort is not set

      5.0.0.0/32 is subnetted, 1 subnets
C        5.1.1.1 is directly connected, Loopback1
      6.0.0.0/32 is subnetted, 1 subnets
C        6.1.1.1 is directly connected, Loopback2
      7.0.0.0/32 is subnetted, 1 subnets
C        7.1.1.1 is directly connected, Loopback3
      8.0.0.0/32 is subnetted, 1 subnets
R        8.1.1.1 [120/5] via 192.168.1.2, 00:00:16, Ethernet1/0
      9.0.0.0/32 is subnetted, 1 subnets
R        9.1.1.1 [120/5] via 192.168.1.2, 00:00:16, Ethernet1/0
      10.0.0.0/32 is subnetted, 1 subnets
R        10.1.1.1 [120/5] via 192.168.1.2, 00:00:16, Ethernet1/0
      11.0.0.0/32 is subnetted, 1 subnets
R        11.1.1.1 [120/5] via 192.168.1.2, 00:00:16, Ethernet1/0
      12.0.0.0/32 is subnetted, 1 subnets
R        12.1.1.1 [120/5] via 192.168.1.2, 00:00:16, Ethernet1/0
      13.0.0.0/32 is subnetted, 1 subnets
R        13.1.1.1 [120/5] via 192.168.1.2, 00:00:16, Ethernet1/0
      14.0.0.0/32 is subnetted, 1 subnets
R        14.1.1.1 [120/5] via 192.168.1.2, 00:00:16, Ethernet1/0
      15.0.0.0/32 is subnetted, 1 subnets
R        15.1.1.1 [120/5] via 192.168.1.2, 00:00:16, Ethernet1/0
      16.0.0.0/32 is subnetted, 1 subnets
R        16.1.1.1 [120/5] via 192.168.1.2, 00:00:16, Ethernet1/0
      192.168.1.0/24 is variably subnetted, 2 subnets, 2 masks
C        192.168.1.0/26 is directly connected, Ethernet1/0
L        192.168.1.1/32 is directly connected, Ethernet1/0
R     192.168.2.0/24 [120/5] via 192.168.1.2, 00:00:16, Ethernet1/0
R     192.168.3.0/24 [120/5] via 192.168.1.2, 00:00:16, Ethernet1/0
R     192.168.4.0/24 [120/5] via 192.168.1.2, 00:00:16, Ethernet1/0
```

Figure 7-6

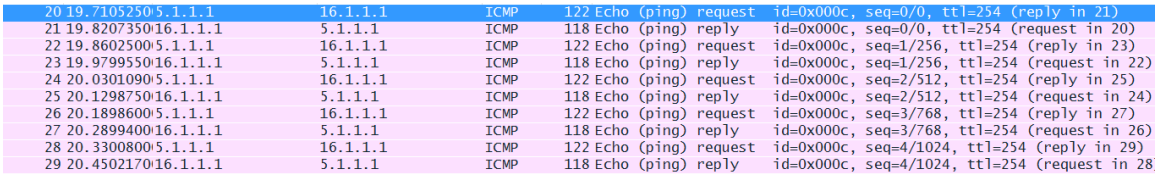
3rd enable debug MPLS packet and look at how the EXP changed in label:

Ping 16.1.1.1 from 5.1.1.1 with ToS 184 (DSCP EF)

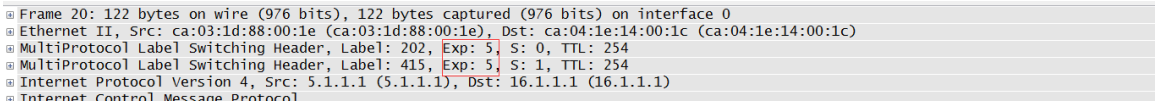
On PE1 debug mpls packet and wireshark to see the label between PE1 and P1.

```
PE1-WEI#end
*Mar 21 19:10:50.604: MPLS turbo: Et1/2: rx: Len 118 Stack {103 4 252} - ipv4 data
*Mar 21 19:10:50.744: MPLS turbo: Et1/2: rx: Len 118 Stack {103 4 252} - ipv4 data
*Mar 21 19:10:50.948: MPLS turbo: Et1/2: rx: Len 118 Stack {103 4 252} - ipv4 data
*Mar 21 19:10:51.080: MPLS turbo: Et1/2: rx: Len 118 Stack {103 4 252} - ipv4 data
*Mar 21 19:10:51.248: MPLS turbo: Et1/2: rx: Len 118 Stack {103 4 252} - ipv4 data
```

Figure 7-7-a



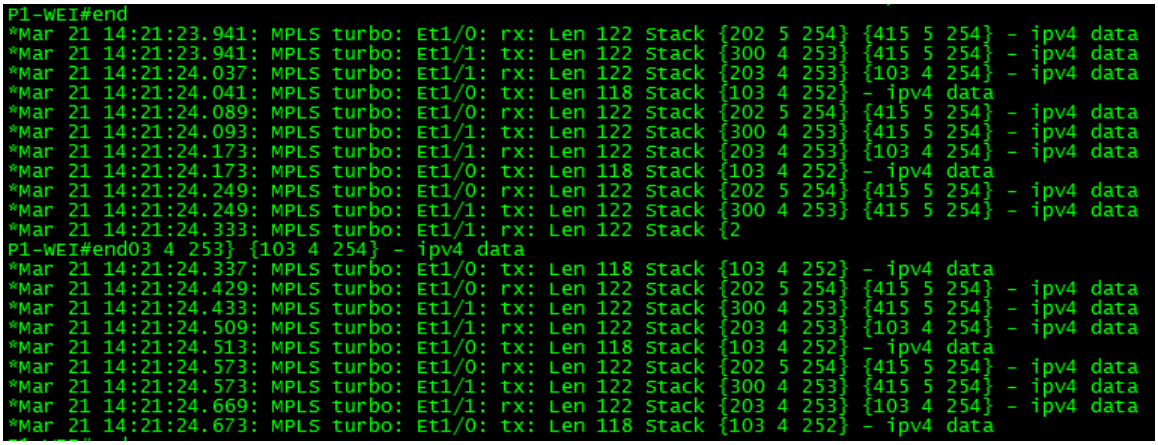
No.	Time	Source	Destination	Protocol	Length	Info
20	19.7105250	5.1.1.1	16.1.1.1	ICMP	122	Echo (ping) request id=0x000c, seq=0/0, ttl=254 (reply in 21)
21	19.8207350	16.1.1.1	5.1.1.1	ICMP	118	Echo (ping) reply id=0x000c, seq=0/0, ttl=254 (request in 20)
22	19.8602500	5.1.1.1	16.1.1.1	ICMP	122	Echo (ping) request id=0x000c, seq=1/256, ttl=254 (reply in 23)
23	19.9799550	16.1.1.1	5.1.1.1	ICMP	118	Echo (ping) reply id=0x000c, seq=1/256, ttl=254 (request in 22)
24	20.0301090	5.1.1.1	16.1.1.1	ICMP	122	Echo (ping) request id=0x000c, seq=2/512, ttl=254 (reply in 25)
25	20.1298750	16.1.1.1	5.1.1.1	ICMP	118	Echo (ping) reply id=0x000c, seq=2/512, ttl=254 (request in 24)
26	20.1898600	5.1.1.1	16.1.1.1	ICMP	122	Echo (ping) request id=0x000c, seq=3/768, ttl=254 (reply in 27)
27	20.2899400	16.1.1.1	5.1.1.1	ICMP	118	Echo (ping) reply id=0x000c, seq=3/768, ttl=254 (request in 26)
28	20.3300800	5.1.1.1	16.1.1.1	ICMP	122	Echo (ping) request id=0x000c, seq=4/1024, ttl=254 (reply in 29)
29	20.4502170	16.1.1.1	5.1.1.1	ICMP	118	Echo (ping) reply id=0x000c, seq=4/1024, ttl=254 (request in 28



```
⊞ Frame 20: 122 bytes on wire (976 bits), 122 bytes captured (976 bits) on interface 0
⊞ Ethernet II, Src: ca:03:1d:88:00:1e (ca:03:1d:88:00:1e), Dst: ca:04:1e:14:00:1c (ca:04:1e:14:00:1c)
⊞ MultiProtocol Label Switching Header, Label: 202, Exp: 5, S: 0, TTL: 254
⊞ MultiProtocol Label Switching Header, Label: 415, Exp: 5, S: 1, TTL: 254
⊞ Internet Protocol Version 4, Src: 5.1.1.1 (5.1.1.1), Dst: 16.1.1.1 (16.1.1.1)
⊞ Internet Control Message Protocol
```

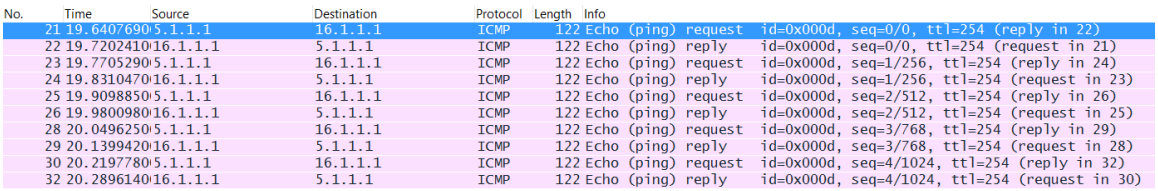
Figure 7-7-b

On P1: the topmost label has changed, wireshark between P1 and P2

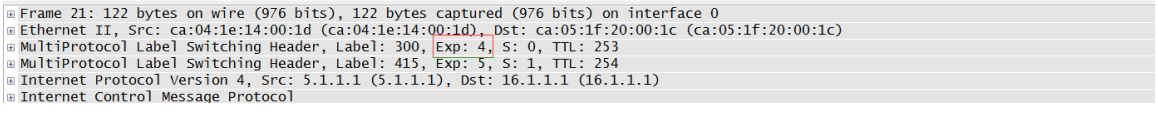


```
P1-WEI#end
*Mar 21 14:21:23.941: MPLS turbo: Et1/0: rx: Len 122 Stack {202 5 254} {415 5 254} - ipv4 data
*Mar 21 14:21:23.941: MPLS turbo: Et1/1: tx: Len 122 Stack {300 4 253} {415 5 254} - ipv4 data
*Mar 21 14:21:24.037: MPLS turbo: Et1/1: rx: Len 122 Stack {203 4 253} {103 4 254} - ipv4 data
*Mar 21 14:21:24.041: MPLS turbo: Et1/0: tx: Len 118 Stack {103 4 252} - ipv4 data
*Mar 21 14:21:24.089: MPLS turbo: Et1/0: rx: Len 122 Stack {202 5 254} {415 5 254} - ipv4 data
*Mar 21 14:21:24.093: MPLS turbo: Et1/1: tx: Len 122 Stack {300 4 253} {415 5 254} - ipv4 data
*Mar 21 14:21:24.173: MPLS turbo: Et1/1: rx: Len 122 Stack {203 4 253} {103 4 254} - ipv4 data
*Mar 21 14:21:24.173: MPLS turbo: Et1/0: tx: Len 118 Stack {103 4 252} - ipv4 data
*Mar 21 14:21:24.249: MPLS turbo: Et1/0: rx: Len 122 Stack {202 5 254} {415 5 254} - ipv4 data
*Mar 21 14:21:24.249: MPLS turbo: Et1/1: tx: Len 122 Stack {300 4 253} {415 5 254} - ipv4 data
*Mar 21 14:21:24.333: MPLS turbo: Et1/1: rx: Len 122 Stack {2
P1-WEI#end03 4 253} {103 4 254} - ipv4 data
*Mar 21 14:21:24.337: MPLS turbo: Et1/0: tx: Len 118 Stack {103 4 252} - ipv4 data
*Mar 21 14:21:24.429: MPLS turbo: Et1/0: rx: Len 122 Stack {202 5 254} {415 5 254} - ipv4 data
*Mar 21 14:21:24.433: MPLS turbo: Et1/1: tx: Len 122 Stack {300 4 253} {415 5 254} - ipv4 data
*Mar 21 14:21:24.509: MPLS turbo: Et1/1: rx: Len 122 Stack {203 4 253} {103 4 254} - ipv4 data
*Mar 21 14:21:24.513: MPLS turbo: Et1/0: tx: Len 118 Stack {103 4 252} - ipv4 data
*Mar 21 14:21:24.573: MPLS turbo: Et1/0: rx: Len 122 Stack {202 5 254} {415 5 254} - ipv4 data
*Mar 21 14:21:24.573: MPLS turbo: Et1/1: tx: Len 122 Stack {300 4 253} {415 5 254} - ipv4 data
*Mar 21 14:21:24.669: MPLS turbo: Et1/1: rx: Len 122 Stack {203 4 253} {103 4 254} - ipv4 data
*Mar 21 14:21:24.673: MPLS turbo: Et1/0: tx: Len 118 Stack {103 4 252} - ipv4 data
```

Figure 7-8-a



No.	Time	Source	Destination	Protocol	Length	Info
21	19.6407690	5.1.1.1	16.1.1.1	ICMP	122	Echo (ping) request id=0x000d, seq=0/0, ttl=254 (reply in 22)
22	19.7202410	16.1.1.1	5.1.1.1	ICMP	122	Echo (ping) reply id=0x000d, seq=0/0, ttl=254 (request in 21)
23	19.7705290	5.1.1.1	16.1.1.1	ICMP	122	Echo (ping) request id=0x000d, seq=1/256, ttl=254 (reply in 24)
24	19.8310470	16.1.1.1	5.1.1.1	ICMP	122	Echo (ping) reply id=0x000d, seq=1/256, ttl=254 (request in 23)
25	19.9098850	5.1.1.1	16.1.1.1	ICMP	122	Echo (ping) request id=0x000d, seq=2/512, ttl=254 (reply in 26)
26	19.9800980	16.1.1.1	5.1.1.1	ICMP	122	Echo (ping) reply id=0x000d, seq=2/512, ttl=254 (request in 25)
28	20.0496250	5.1.1.1	16.1.1.1	ICMP	122	Echo (ping) request id=0x000d, seq=3/768, ttl=254 (reply in 29)
29	20.1399420	16.1.1.1	5.1.1.1	ICMP	122	Echo (ping) reply id=0x000d, seq=3/768, ttl=254 (request in 28)
30	20.2197780	5.1.1.1	16.1.1.1	ICMP	122	Echo (ping) request id=0x000d, seq=4/1024, ttl=254 (reply in 32)
32	20.2896140	16.1.1.1	5.1.1.1	ICMP	122	Echo (ping) reply id=0x000d, seq=4/1024, ttl=254 (request in 30)



```
⊞ Frame 21: 122 bytes on wire (976 bits), 122 bytes captured (976 bits) on interface 0
⊞ Ethernet II, Src: ca:04:1e:14:00:1d (ca:04:1e:14:00:1d), Dst: ca:05:1f:20:00:1c (ca:05:1f:20:00:1c)
⊞ MultiProtocol Label Switching Header, Label: 300, Exp: 4, S: 0, TTL: 253
⊞ MultiProtocol Label Switching Header, Label: 415, Exp: 5, S: 1, TTL: 254
⊞ Internet Protocol Version 4, Src: 5.1.1.1 (5.1.1.1), Dst: 16.1.1.1 (16.1.1.1)
⊞ Internet Control Message Protocol
```

Figure 7-8-b

On P2:

```
P2-WEI#end
*Mar 21 14:19:03.762: MPLS turbo: Et1/0: rx: Len 122 Stack {300 4 253} {415 5 254} - ipv4 data
*Mar 21 14:19:03.762: MPLS turbo: Et1/1: tx: Len 118 Stack {415 4 252} - ipv4 data
*Mar 21 14:19:03.846: MPLS turbo: Et1/1: rx: Len 122 Stack {302 4 254} {103 4 254} - ipv4 data
*Mar 21 14:19:03.850: MPLS turbo: Et1/0: tx: Len 122 Stack {203 4 253} {103 4 254} - ipv4 data
*Mar 21 14:19:03.910: MPLS turbo: Et1/0: rx: Len 122 Stack {300 4 253} {415 5 254} - ipv4 data
*Mar 21 14:19:03.910: MPLS turbo: Et1/1: tx: Len 118 Stack {415 4 252} - ipv4 data
*Mar 21 14:19:03.950: MPLS turbo: Et1/1: rx: Len 122 Stack {302 4 254} {103 4 254} - ipv4 data
*Mar 21 14:19:03.950: MPLS turbo: Et1/0: tx: Len 122 Stack {203 4 253} {103 4 254} - ipv4 data
*Mar 21 14:19:04.086: MPLS turbo: Et1/0: rx: Len 122 Stack {300 4 253} {415 5 254} - ipv4 data
*Mar 21 14:19:04.086: MPLS turbo: Et1/1: tx: Len 118 Stack {415 4 252} - ipv4 data
*Mar 21 14:19:04.162: MPLS turbo: Et1/1: rx: Len 122 Stack {302 4 254} {1
P2-WEI#end03 4 254} - ipv4 data
*Mar 21 14:19:04.162: MPLS turbo: Et1/0: tx: Len 122 Stack {203 4 253} {103 4 254} - ipv4 data
*Mar 21 14:19:04.258: MPLS turbo: Et1/0: rx: Len 122 Stack {300 4 253} {415 5 254} - ipv4 data
*Mar 21 14:19:04.262: MPLS turbo: Et1/1: tx: Len 118 Stack {415 4 252} - ipv4 data
*Mar 21 14:19:04.286: MPLS turbo: Et1/1: rx: Len 122 Stack {302 4 254} {103 4 254} - ipv4 data
*Mar 21 14:19:04.290: MPLS turbo: Et1/0: tx: Len 122 Stack {203 4 253} {103 4 254} - ipv4 data
*Mar 21 14:19:04.394: MPLS turbo: Et1/0: rx: Len 122 Stack {300 4 253} {415 5 254} - ipv4 data
*Mar 21 14:19:04.394: MPLS turbo: Et1/1: tx: Len 118 Stack {415 4 252} - ipv4 data
*Mar 21 14:19:04.454: MPLS turbo: Et1/1: rx: Len 122 Stack {302 4 254} {103 4 254} - ipv4 data
*Mar 21 14:19:04.458: MPLS turbo: Et1/0: tx: Len 122 Stack {203 4 253} {103 4 254} - ipv4 data
P2-WEI#end
```

Figure 7-9

On PE2: the EXP bits has been swapped to 4 according to configuration. Wireshark between P2 and PE2

```
PE2-WEI#end
*Mar 21 14:21:53.621: MPLS turbo: Et1/0: rx: Len 118 Stack {415 4 252} - ipv4 data
*Mar 21 14:21:53.749: MPLS turbo: Et1/0: rx: Len 118 Stack {415 4 252} - ipv4 data
*Mar 21 14:21:53.937: MPLS turbo: Et1/0: rx: Len 118 Stack {415 4 252} - ipv4 data
*Mar 21 14:21:54.097: MPLS turbo: Et1/0: rx: Len 118 Stack {415 4 252} - ipv4 data
*Mar 21 14:21:54.257: MPLS turbo: Et1/0: rx: Len 118 Stack {415 4 252} - ipv4 data
```

Figure 7-10-a

No.	Time	Source	Destination	Protocol	Length	Info
47	42.8791490	5.1.1.1	16.1.1.1	ICMP	118	Echo (ping) request id=0x000e, seq=0/0, ttl=254 (reply in 48)
48	42.9098080	16.1.1.1	5.1.1.1	ICMP	122	Echo (ping) reply id=0x000e, seq=0/0, ttl=254 (request in 47)
49	43.0288380	5.1.1.1	16.1.1.1	ICMP	118	Echo (ping) request id=0x000e, seq=1/256, ttl=254 (reply in 50)
50	43.0792310	16.1.1.1	5.1.1.1	ICMP	122	Echo (ping) reply id=0x000e, seq=1/256, ttl=254 (request in 49)
51	43.1893580	5.1.1.1	16.1.1.1	ICMP	118	Echo (ping) request id=0x000e, seq=2/512, ttl=254 (reply in 52)
52	43.2298590	16.1.1.1	5.1.1.1	ICMP	122	Echo (ping) reply id=0x000e, seq=2/512, ttl=254 (request in 51)
53	43.3403730	5.1.1.1	16.1.1.1	ICMP	118	Echo (ping) request id=0x000e, seq=3/768, ttl=254 (reply in 54)
54	43.3893240	16.1.1.1	5.1.1.1	ICMP	122	Echo (ping) reply id=0x000e, seq=3/768, ttl=254 (request in 53)
55	43.4893760	5.1.1.1	16.1.1.1	ICMP	118	Echo (ping) request id=0x000e, seq=4/1024, ttl=254 (reply in 56)
56	43.5393440	16.1.1.1	5.1.1.1	ICMP	122	Echo (ping) reply id=0x000e, seq=4/1024, ttl=254 (request in 55)

```
⊞ Frame 47: 118 bytes on wire (944 bits), 118 bytes captured (944 bits) on interface 0
⊞ Ethernet II, Src: ca:05:1f:20:00:1d (ca:05:1f:20:00:1d), Dst: ca:06:1f:ac:00:1c (ca:06:1f:ac:00:1c)
⊞ MultiProtocol Label Switching Header, Label: 415, Exp: 4, S: 1, TTL: 252
⊞ Internet Protocol Version 4, Src: 5.1.1.1 (5.1.1.1), Dst: 16.1.1.1 (16.1.1.1)
⊞ Internet Control Message Protocol
```

Figure 7-10-b

And the packet left PE2 and DSCP has been changed to AF43

No.	Time	Source	Destination	Protocol	Length	Info
19	24.9300480	5.1.1.1	16.1.1.1	ICMP	114	Echo (ping) request id=0x000e, seq=0/0, ttl=251 (reply in 20)
20	24.9402390	16.1.1.1	5.1.1.1	ICMP	114	Echo (ping) reply id=0x000e, seq=0/0, ttl=255 (request in 19)
21	25.0797370	5.1.1.1	16.1.1.1	ICMP	114	Echo (ping) request id=0x000e, seq=1/256, ttl=251 (reply in 22)
22	25.1000410	16.1.1.1	5.1.1.1	ICMP	114	Echo (ping) reply id=0x000e, seq=1/256, ttl=255 (request in 21)
23	25.2407370	5.1.1.1	16.1.1.1	ICMP	114	Echo (ping) request id=0x000e, seq=2/512, ttl=251 (reply in 24)
24	25.2603800	16.1.1.1	5.1.1.1	ICMP	114	Echo (ping) reply id=0x000e, seq=2/512, ttl=255 (request in 23)
25	25.4004440	5.1.1.1	16.1.1.1	ICMP	114	Echo (ping) request id=0x000e, seq=3/768, ttl=251 (reply in 26)
26	25.4201910	16.1.1.1	5.1.1.1	ICMP	114	Echo (ping) reply id=0x000e, seq=3/768, ttl=255 (request in 25)
27	25.5407570	5.1.1.1	16.1.1.1	ICMP	114	Echo (ping) request id=0x000e, seq=4/1024, ttl=251 (reply in 28)
28	25.5598490	16.1.1.1	5.1.1.1	ICMP	114	Echo (ping) reply id=0x000e, seq=4/1024, ttl=255 (request in 27)

```
Frame 19: 114 bytes on wire (912 bits), 114 bytes captured (912 bits) on interface 0
Ethernet II, Src: ca:06:1f:ac:00:1d (ca:06:1f:ac:00:1d), Dst: ca:07:19:78:00:1c (ca:07:19:78:00:1c)
Internet Protocol Version 4, Src: 5.1.1.1 (5.1.1.1), Dst: 16.1.1.1 (16.1.1.1)
   Version: 4
   Header Length: 20 bytes
   Differentiated Services Field: 0x98 (DSCP 0x26: Assured Forwarding 43; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))
   Total Length: 100
   Identification: 0x0046 (70)
   Flags: 0x00
   Fragment offset: 0
   Time to live: 251
   Protocol: ICMP (1)
   Header checksum: 0xa7b7 [validation disabled]
```

Figure 7-11

Now let us take a look on policy-maps on CE routers

```
Service-policy output: shaping

  Class-map: VOICE (match-all)
    10 packets, 1000 bytes
    5 minute offered rate 0 bps, drop rate 0 bps
    Match:  dscp ef (46)
    Queueing
    queue limit 64 packets
    (queue depth/total drops/no-buffer drops) 0/0/0
    (pkts output/bytes output) 10/1140
    shape (peak) cir 50000000, bc 5000000, be 5000000
    target shape rate 100000000

  Class-map: VIDEOCON (match-all)
    0 packets, 0 bytes
    5 minute offered rate 0 bps, drop rate 0 bps
    Match:  dscp af41 (34)
    Queueing
    queue limit 64 packets
    (queue depth/total drops/no-buffer drops) 0/0/0
    (pkts output/bytes output) 0/0
    shape (peak) cir 50000000, bc 5000000, be 5000000
    target shape rate 100000000

  Class-map: CRITICALDATA (match-all)
    0 packets, 0 bytes
    5 minute offered rate 0 bps, drop rate 0 bps
    Match:  dscp af31 (26)
    Queueing
    queue limit 64 packets
    (queue depth/total drops/no-buffer drops) 0/0/0
    (pkts output/bytes output) 0/0
    shape (peak) cir 50000000, bc 5000000, be 5000000
    target shape rate 100000000

  Class-map: CALLSIGNAL (match-all)
    0 packets, 0 bytes
    5 minute offered rate 0 bps, drop rate 0 bps
    Match:  dscp cs3 (24)
    Queueing
    queue limit 64 packets
    (queue depth/total drops/no-buffer drops) 0/0/0
    (pkts output/bytes output) 0/0
    shape (peak) cir 50000000, bc 5000000, be 5000000
    target shape rate 100000000

  Class-map: VIDEOSTR (match-all)
```

Figure 7-12

Considering about Oversubscription: Due to the limitation of bandwidth on MPLS core network at 100Mbps. If ISP sell 100Mbps, it might not reach 100Mbps for two customers connected to PE router. Thus, we shape the traffic at the egress of the CE routers configured with peak which results the target rate= (1+Be/Bc) *CIR=100000000bps. It brings capability of reaching 100Mbps when the link is idle, and it also can limit the rate to 50Mbps when the two node having traffic to be sent out simultaneously. In case that too much traffic is from both sides of customers, so we implement policing on the ingress interface to monitor the traffic flow depending on the service level to allocate the maximum rate. The detail of policy-map configurations shows as below:

PE1-WEI#sh policy-map interface

GigabitEthernet1/0

Service-policy input: set-MPLS-EXP

Class-map: BESTEFF (match-any)

0 packets, 0 bytes

5 minute offered rate 0 bps, drop rate 0 bps

Match: dscp default (0)

0 packets, 0 bytes

5 minute rate 0 bps

police:

cir 2000000 bps, bc 62500 bytes

pir 50000000 bps, be 1562500 bytes

conformed 0 packets, 0 bytes; actions:

set-mpls-exp-imposition-transmit 0

exceeded 0 packets, 0 bytes; actions:

set-mpls-exp-imposition-transmit 0

violated 0 packets, 0 bytes; actions:

drop

conformed 0 bps, exceed 0 bps, violate 0 bps

Class-map: BD&SCA (match-any)

0 packets, 0 bytes

5 minute offered rate 0 bps, drop rate 0 bps

Match: dscp af11 (10)

0 packets, 0 bytes

5 minute rate 0 bps

```
  Match:  dscp cs1 (8)
    0 packets, 0 bytes
    5 minute rate 0 bps
  police:
      cir 4000000 bps, bc 125000 bytes
      pir 50000000 bps, be 1562500 bytes
    conformed 0 packets, 0 bytes; actions:
      set-mpls-exp-imposition-transmit 1
    exceeded 0 packets, 0 bytes; actions:
      set-mpls-exp-imposition-transmit 0
    violated 0 packets, 0 bytes; actions:
      drop
    conformed 0 bps, exceed 0 bps, violate 0 bps

Class-map: VIDEODA (match-any)
  0 packets, 0 bytes
  5 minute offered rate 0 bps, drop rate 0 bps
  Match:  dscp af41 (34)
    0 packets, 0 bytes
    5 minute rate 0 bps
  police:
      cir 15000000 bps, bc 468750 bytes
      pir 50000000 bps, be 1562500 bytes
    conformed 0 packets, 0 bytes; actions:
      set-mpls-exp-imposition-transmit 4
    exceeded 0 packets, 0 bytes; actions:
      set-mpls-exp-imposition-transmit 3
    violated 0 packets, 0 bytes; actions:
      drop
    conformed 0 bps, exceed 0 bps, violate 0 bps
Class-map: CD&CS (match-any)
  0 packets, 0 bytes
  5 minute offered rate 0 bps, drop rate 0 bps
  Match:  dscp af31 (26)
    0 packets, 0 bytes
```

```
    5 minute rate 0 bps
  Match:  dscp cs3 (24)
    0 packets, 0 bytes
    5 minute rate 0 bps
  police:
      cir 5000000 bps, bc 156250 bytes
      pir 50000000 bps, be 1562500 bytes
    conformed 0 packets, 0 bytes; actions:
      set-mpls-exp-imposition-transmit 3
    exceeded 0 packets, 0 bytes; actions:
      set-mpls-exp-imposition-transmit 2
    violated 0 packets, 0 bytes; actions:
      drop
    conformed 0 bps, exceed 0 bps, violate 0 bps
Class-map: NM (match-any)
  0 packets, 0 bytes
  5 minute offered rate 0 bps, drop rate 0 bps
  Match:  dscp cs4 (32)
    0 packets, 0 bytes
    5 minute rate 0 bps
  Match:  dscp cs2 (16)
    0 packets, 0 bytes
    5 minute rate 0 bps
  police:
      cir 9000000 bps, bc 281250 bytes
      pir 50000000 bps, be 1562500 bytes
    conformed 0 packets, 0 bytes; actions:
      set-mpls-exp-imposition-transmit 2
    exceeded 0 packets, 0 bytes; actions:
      set-mpls-exp-imposition-transmit 1
    violated 0 packets, 0 bytes; actions:
      drop
    conformed 0 bps, exceed 0 bps, violate 0 bps
Class-map: VOICE (match-any)
  621 packets, 868102 bytes
```

5 minute offered rate 20000 bps, drop rate 0 bps

Match: dscp ef (46)

621 packets, 868102 bytes

5 minute rate 20000 bps

police:

cir 15000000 bps, bc 468750 bytes

pir 50000000 bps, be 1562500 bytes

conformed 621 packets, 868102 bytes; actions:

set-mpls-exp-imposition-transmit 5

exceeded 0 packets, 0 bytes; actions:

set-mpls-exp-imposition-transmit 4

violated 0 packets, 0 bytes; actions:

drop

conformed 43000 bps, exceed 0 bps, violate 0 bps

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

37 packets, 3922 bytes

5 minute offered rate 0 bps, drop rate 0 bps

Match: anyFurthermore, on account of service level, we are using CB-WFQ and LLQ in here, where voice traffic (DSCP EF) and video conferencing with PQ always been scheduling first, other traffic configured with CB-WFQ with assigning guaranteed bandwidth to each class. The LLQ shows:

FastEthernet1/2

Service-policy output: outbound

queue stats for all priority classes:

queue limit 64 packets

(queue depth/total drops/no-buffer drops) 0/0/0

(pkts output/bytes output) 1185/896758

Class-map: MPLS-5 (match-all)

621 packets, 868102 bytes

5 minute offered rate 4000 bps, drop rate 0 bps

Match: mpls experimental topmost 5

Priority: 25% (25000 kbps), burst bytes 625000, b/w exceed drops: 0

Class-map: MPLS-0 (match-all)

1 packets, 70 bytes

5 minute offered rate 0 bps, drop rate 0 bps

Match: mpls experimental topmost 0

Queueing

queue limit 64 packets

(queue depth/total drops/no-buffer drops) 0/0/0

(pkts output/bytes output) 1/78

bandwidth 3% (3000 kbps)

Exp-weight-constant: 9 (1/512)

Mean queue depth: 0 packets

class	Transmitted	Random drop	Tail drop	Minimum	Maximum	Mark
	pkts/bytes	pkts/bytes	pkts/bytes	thresh	thresh	prob
0	1/78	0/0	0/0	20	40	1/10
1	0/0	0/0	0/0	22	40	1/10
2	0/0	0/0	0/0	24	40	1/10
3	0/0	0/0	0/0	26	40	1/10
4	0/0	0/0	0/0	28	40	1/10
5	0/0	0/0	0/0	30	40	1/10
6	0/0	0/0	0/0	32	40	1/10
7	0/0	0/0	0/0	34	40	1/10

Class-map: MPLS-1 (match-all)

0 packets, 0 bytes

5 minute offered rate 0 bps, drop rate 0 bps

Match: mpls experimental topmost 1

Queueing

queue limit 64 packets

(queue depth/total drops/no-buffer drops) 0/0/0

(pkts output/bytes output) 0/0

bandwidth 5% (5000 kbps)

Exp-weight-constant: 9 (1/512)

Mean queue depth: 0 packets

class	Transmitted	Random drop	Tail drop	Minimum	Maximum	Mark
	pkts/bytes	pkts/bytes	pkts/bytes	thresh	thresh	prob
0	0/0	0/0	0/0	20	40	1/10
1	0/0	0/0	0/0	22	40	1/10
2	0/0	0/0	0/0	24	40	1/10
3	0/0	0/0	0/0	26	40	1/10

4	0/0	0/0	0/0	28	40	1/10
5	0/0	0/0	0/0	30	40	1/10
6	0/0	0/0	0/0	32	40	1/10
7	0/0	0/0	0/0	34	40	1/10

Class-map: MPLS-2 (match-all)

0 packets, 0 bytes

5 minute offered rate 0 bps, drop rate 0 bps

Match: mpls experimental topmost 2

Queueing

queue limit 64 packets

(queue depth/total drops/no-buffer drops) 0/0/0

(pkts output/bytes output) 0/0

bandwidth 20% (20000 kbps)

Exp-weight-constant: 9 (1/512)

Mean queue depth: 0 packets

class	Transmitted pkts/bytes	Random drop pkts/bytes	Tail drop pkts/bytes	Minimum thresh	Maximum thresh	Mark prob
0	0/0	0/0	0/0	20	40	1/10
1	0/0	0/0	0/0	22	40	1/10
2	0/0	0/0	0/0	24	40	1/10
3	0/0	0/0	0/0	26	40	1/10
4	0/0	0/0	0/0	28	40	1/10
5	0/0	0/0	0/0	30	40	1/10
6	0/0	0/0	0/0	32	40	1/10
7	0/0	0/0	0/0	34	40	1/10

Class-map: MPLS-3 (match-all)

0 packets, 0 bytes

5 minute offered rate 0 bps, drop rate 0 bps

Match: mpls experimental topmost 3

Queueing

queue limit 64 packets

(queue depth/total drops/no-buffer drops) 0/0/0

(pkts output/bytes output) 0/0

bandwidth 15% (15000 kbps)

Exp-weight-constant: 9 (1/512)

Mean queue depth: 0 packets

class	Transmitted	Random drop	Tail drop	Minimum	Maximum	Mark
	pkts/bytes	pkts/bytes	pkts/bytes	thresh	thresh	prob
0	0/0	0/0	0/0	20	40	1/10
1	0/0	0/0	0/0	22	40	1/10
2	0/0	0/0	0/0	24	40	1/10
3	0/0	0/0	0/0	26	40	1/10
4	0/0	0/0	0/0	28	40	1/10
5	0/0	0/0	0/0	30	40	1/10
6	0/0	0/0	0/0	32	40	1/10
7	0/0	0/0	0/0	34	40	1/10

Class-map: MPLS-4 (match-all)

0 packets, 0 bytes

5 minute offered rate 0 bps, drop rate 0 bps

Match: mpls experimental topmost 4

Priority: 20% (20000 kbps), burst bytes 500000, b/w exceed drops: 0

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

583 packets, 45835 bytes

5 minute offered rate 0 bps, drop rate 0 bps

Match: any

queue limit 64 packets

(queue depth/total drops/no-buffer drops) 0/0/0

(pkts output/bytes output) 583/49688

7.2.2 Pipe mode

Pipe mode supply 2 layers of QoS. Ingress PE router defines a newly imposed EXP bits without taking default DSCP mapping to EXP. In this way, the ISP might have their own policy to schedule packets without changing DSCP value. When the EXP of topmost label has change as policy requesting, the EXP of the topmost label is copied into a newly exposed label. Since packets exit the ISP MPLS core network, the packet simply starts to remove label and remain unchanged DSCP value on the egress PE. We have to notice that the outbound interface of PE router (PE to CE) copes with the newly exposed IP packets for congestion management and avoidance depending on MPLS EXP bits from the recently popped label. This mode is usually used when MPLS core network has different QoS policy with customer network, and EXP marking is just temporarily used traversing the core.

The illustration of EXP and DSCP swapping shows:

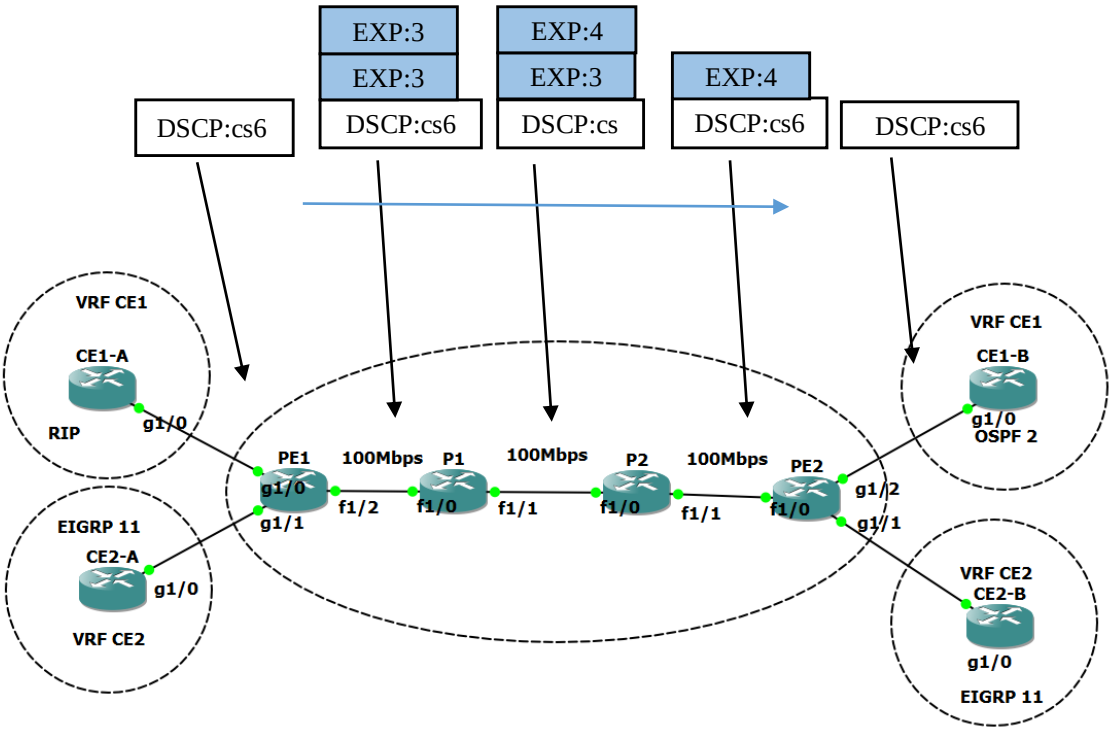


Figure 7-13

http://www.cisco.com/c/en/us/support/docs/multiprotocol-label-switching-mpls/mpls/47815-diffserv-tunnel.html

The connectivity configuration is based on previous setting. Here we only give class-map and policy-map configuration to each of ISP routers.

Cisco provides a five service EXP mapping for MPLS core network:

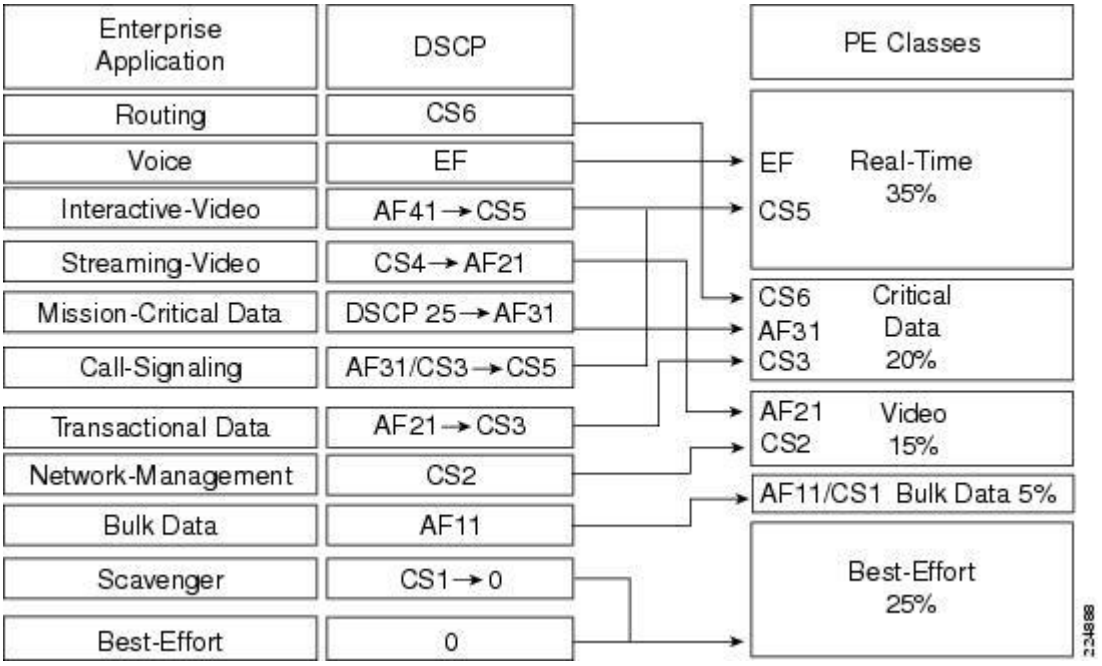


Figure 7-14

http://www.cisco.com/c/en/us/td/docs/solutions/Enterprise/WAN_and_MAN/QoS_SRND/QoS-SRND-Book/VPNQoS.html

Representation of DSCP for different service mapping to EXP 3 bits, customized table:

Service	DSCP	EXP	CB-WFQ
Real-time	EF voice	5	Priority 35%
	CS5 interact video		
Critical data	CS6	3	Bandwidth 20%
	AF31	3	
	CS3	3	
Video	AF2	2	Bandwidth 15%
	CS2	2	
Bulk data	AF11	1	Bandwidth 5%
	CS1	1	

Best effort	0	0	Bandwidth 25%

Table 7-15

The policy-map and class-map on each router:

PE1: class-map match-any REALTIME
 match ip dscp ef
 match ip dscp cs5
class-map match-all MPLS-5
 match mpls experimental topmost 5
class-map match-all MPLS-4
 match mpls experimental topmost 4
class-map match-all MPLS-3
 match mpls experimental topmost 3
class-map match-all MPLS-2
 match mpls experimental topmost 2
class-map match-all MPLS-1
 match mpls experimental topmost 1
class-map match-all MPLS-0
 match mpls experimental topmost 0
class-map match-any BULK-DATA
 match ip dscp af11
 match ip dscp cs1
class-map match-any CRITICAL-DATA
 match ip dscp cs6
 match ip dscp af31
 match ip dscp cs3
class-map match-any VIDEO
 match ip dscp af21
 match ip dscp cs2
!
!
policy-map outbound
 class MPLS-5
  priority percent 35
 class MPLS-3
  bandwidth percent 20
  random-detect
 class MPLS-2
  priority percent 15
 class MPLS-1
  bandwidth percent 5
  random-detect
policy-map set-MPLS-EXP
 class REALTIME
  police cir 17500000 pir 50000000
   conform-action set-mpls-exp-imposition-transmit 5
   exceed-action set-mpls-exp-imposition-transmit 3
   violate-action drop
 class CRITICAL-DATA
  police cir 10000000 pir 50000000
   conform-action set-mpls-exp-imposition-transmit 3
   exceed-action set-mpls-exp-imposition-transmit 2
   violate-action drop
 class VIDEO
  police cir 7500000 pir 50000000
   conform-action set-mpls-exp-imposition-transmit 2
   exceed-action set-mpls-exp-imposition-transmit 1
   violate-action drop
 class BULK-DATA
  police cir 2500000 pir 50000000

conform-action set-mpls-exp-imposition-transmit 1

exceed-action set-mpls-exp-imposition-transmit 0

violate-action drop

P1: class-map match-all MPLS-IN

match mpls experimental topmost 3

policy-map MPLS-IN

class MPLS-IN

set mpls experimental topmost 4

P2: class-map match-all MPLS-5

match mpls experimental topmost 5

class-map match-all MPLS-4

match mpls experimental topmost 4

class-map match-all MPLS-3

match mpls experimental topmost 3

class-map match-all MPLS-2

match mpls experimental topmost 2

class-map match-all MPLS-1

match mpls experimental topmost 1

class-map match-all MPLS-0

match mpls experimental topmost 0

!

!

policy-map qos-group-in

class MPLS-4

set qos-group mpls experimental topmost

class MPLS-1

set qos-group mpls experimental topmost

class MPLS-2

set qos-group mpls experimental topmost

class MPLS-3

set qos-group mpls experimental topmost

class MPLS-5

set qos-group mpls experimental topmost

class MPLS-0

set qos-group mpls experimental topmost

policy-map qos-group-out

class MPLS-5

priority percent 35

set mpls experimental topmost qos-group

class MPLS-4

bandwidth percent 20

random-detect

set mpls experimental topmost qos-group

class MPLS-3

bandwidth percent 20

random-detect

set mpls experimental topmost qos-group

class MPLS-2

priority percent 15

set mpls experimental topmost qos-group

class MPLS-1

bandwidth percent 5

random-detect

set mpls experimental topmost qos-group

PE2: class-map match-all MPLS-5

match mpls experimental topmost 5

class-map match-all MPLS-4

match mpls experimental topmost 4

class-map match-all MPLS-3

match mpls experimental topmost 3

class-map match-all MPLS-2

match mpls experimental topmost 2

class-map match-all MPLS-1

match mpls experimental topmost 1

class-map match-all MPLS-0

match mpls experimental topmost 0

policy-map qos-group-in

class MPLS-4

set qos-group mpls experimental topmost

```
 set discard-class 4                                  set discard-class 0
class MPLS-1                                         policy-map outbound
 set qos-group mpls experimental topmost              class MPLS-5
 set discard-class 1                                   priority percent 35
class MPLS-2                                         class MPLS-4
 set qos-group mpls experimental topmost              bandwidth percent 20
 set discard-class 2                                  random-detect discard-class-based
class MPLS-3                                         class MPLS-3
 set qos-group mpls experimental topmost              bandwidth percent 20
 set discard-class 3                                  random-detect discard-class-based
class MPLS-5                                         class MPLS-2
 set qos-group mpls experimental topmost              priority percent 15
 set discard-class 5                                  class MPLS-1
class MPLS-0                                          bandwidth percent 5
 set qos-group mpls experimental topmost              random-detect discard-class-based
```

Now take a look how the EXP and DSCP value changed. Ping 16.1.1.1 from 5.1.1.1 with DSCP CS6.

On P1 debug mpls packet:

```
P1-WEI#
*Mar 25 05:52:14.991: MPLS turbo: Et1/0: rx: Len 122 Stack {201 3 254} {411 3 254} - ipv4 data
*Mar 25 05:52:14.991: MPLS turbo: Et1/1: tx: Len 122 Stack {300 4 253} {411 3 254} - ipv4 data
*Mar 25 05:52:15.063: MPLS turbo: Et1/1: rx: Len 122 Stack {200 6 253} {103 6 254} - ipv4 data
*Mar 25 05:52:15.063: MPLS turbo: Et1/0: tx: Len 118 Stack {103 6 252} - ipv4 data
*Mar 25 05:52:15.095: MPLS turbo: Et1/0: rx: Len 122 Stack {201 3 254} {411 3 254} - ipv4 data
*Mar 25 05:52:15.099: MPLS turbo: Et1/1: tx: Len 122 Stack {300 4 253} {411 3 254} - ipv4 data
*Mar 25 05:52:15.163: MPLS turbo: Et1/1: rx: Len 122 Stack {200 6 253} {103 6 254} - ipv4 data
*Mar 25 05:52:15.167: MPLS turbo: Et1/0: tx: Len 118 Stack {103 6 252} - ipv4 data
*Mar 25 05:52:15.195: MPLS turbo: Et1/0: rx: Len 122 Stack {201 3 254} {411 3 254} - ipv4 data
*Mar 25 05:52:15.195: MPLS turbo: Et1/1: tx: Len 122 Stack {300 4 253} {411 3 254} - ipv4 data
*Mar 25 05:52:15.267: MPLS turbo: Et1/1: rx: Len 122 Stack {2
P1-WEI#00 6 253} {103 6 254} - ipv4 data
*Mar 25 05:52:15.267: MPLS turbo: Et1/0: tx: Len 118 Stack {103 6 252} - ipv4 data
*Mar 25 05:52:15.299: MPLS turbo: Et1/0: rx: Len 122 Stack {201 3 254} {411 3 254} - ipv4 data
*Mar 25 05:52:15.299: MPLS turbo: Et1/1: tx: Len 122 Stack {300 4 253} {411 3 254} - ipv4 data
*Mar 25 05:52:15.371: MPLS turbo: Et1/1: rx: Len 122 Stack {200 6 253} {103 6 254} - ipv4 data
*Mar 25 05:52:15.371: MPLS turbo: Et1/0: tx: Len 118 Stack {103 6 252} - ipv4 data
*Mar 25 05:52:15.399: MPLS turbo: Et1/0: rx: Len 122 Stack {201 3 254} {411 3 254} - ipv4 data
*Mar 25 05:52:15.399: MPLS turbo: Et1/1: tx: Len 122 Stack {300 4 253} {411 3 254} - ipv4 data
*Mar 25 05:52:15.467: MPLS turbo: Et1/1: rx: Len 122 Stack {200 6 253} {103 6 254} - ipv4 data
*Mar 25 05:52:15.471: MPLS turbo: Et1/0: tx: Len 118 Stack {103 6 252} - ipv4 data
P1-WEI#
```

Figure 7-16

Wireshark between PE1 and P1:

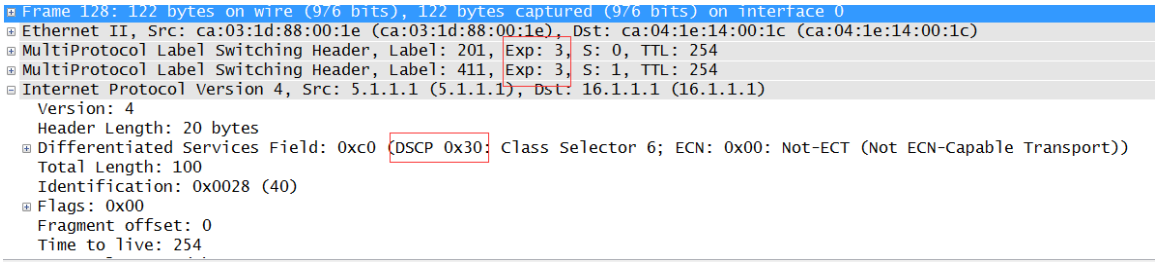


Figure 7-17

EXP didn't exactly copy the default mapping value from DSCP, which means Pipe mode provides MPLS core maintain their own service policies.

P2:

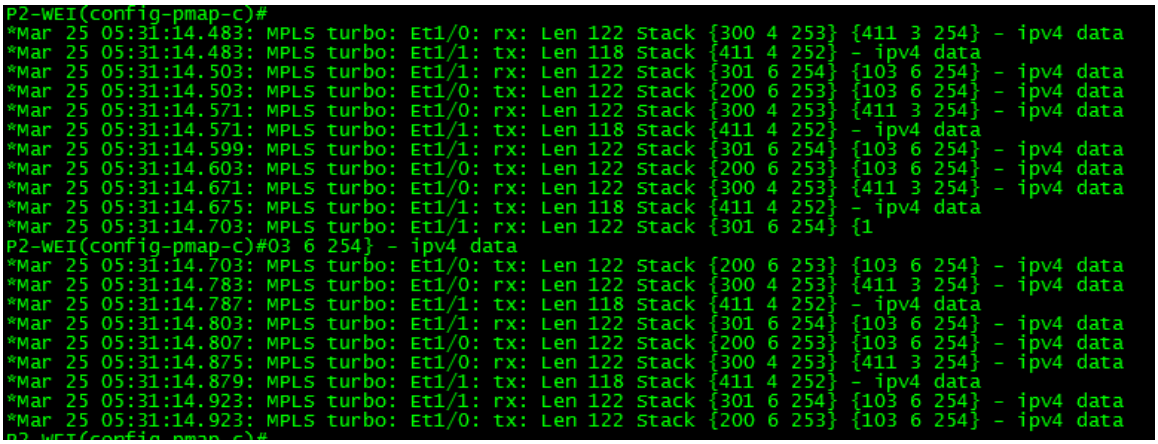


Figure 7-18

Wireshark between P1 and P2:

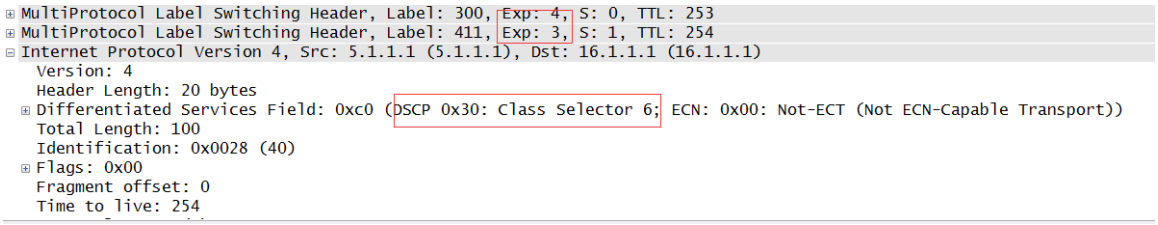


Figure 7-19

The policy took the effect on EXP swapping.

PE2:

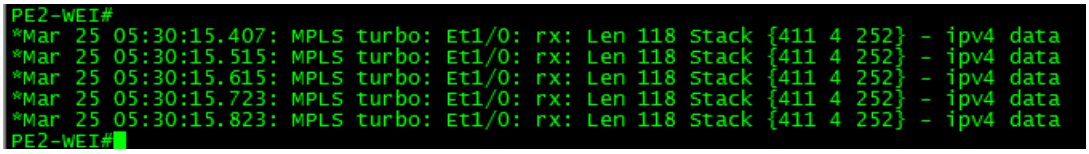


Figure 7-20

Wireshark between P2 and PE2:

```
⊞ Ethernet II, Src: ca:05:1f:20:00:1d (ca:05:1f:20:00:1d), Dst: ca:06:1f:ac:00:1c (ca:06:1f:ac:00:1c)
⊞ MultiProtocol Label Switching Header, Label: 411, Exp: 4, S: 1, TTL: 252
⊟ Internet Protocol Version 4, Src: 5.1.1.1 (5.1.1.1), Dst: 16.1.1.1 (16.1.1.1)
    Version: 4
    Header Length: 20 bytes
  ⊞ Differentiated Services Field: 0xc0 (DSCP 0x30: Class Selector 6; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))
    Total Length: 100
    Identification: 0x0028 (40)
  ⊞ Flags: 0x00
    Fragment offset: 0
    Time to live: 254
    Protocol: ICMP (1)
```

Figure 7-21

The newly exposed mpls label copied the former EXP value of mpls. At PE2 egress interface will take this EXP value to decide outbound queueing.

7.2.3 Short Pipe mode

Short Pipe mode is using same approach to treat and mark the packet across the core, and it is applied to when the occurrence of different regulation between customer network and MPLS core network. The only difference is that the newly outgoing exposed packets on egress PE router will take the DSCP value in IP packets for queueing and congestion early detection.

The illustration of EXP and DSCP swapping shows:

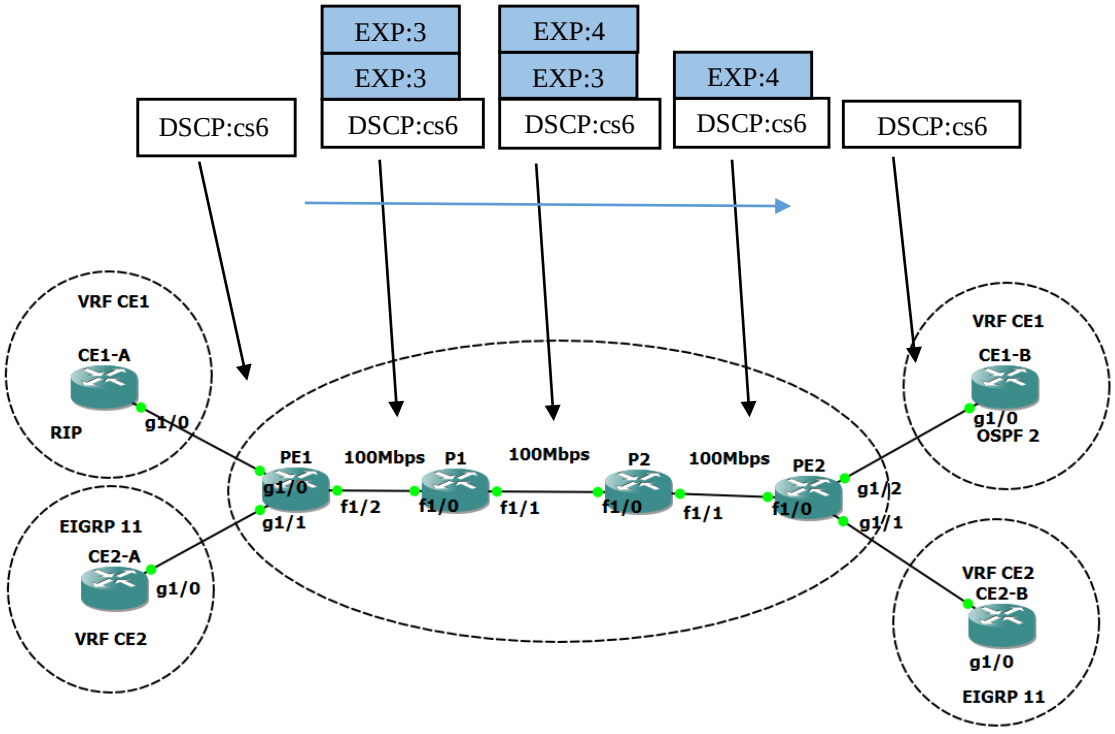


Figure 7-22

http://www.cisco.com/c/en/us/support/docs/multiprotocol-label-switching-mpls/mpls/47815-diffserv-tunnel.html

For short Pipe mode, we just need to change the egress policy-map configuration on PE2. It has to take DSCP value to determine outbound queueing.

```
PE2: policy-map outbound
 class MPLS-5
  priority percent 35
 class MPLS-4
  bandwidth percent 20
  random-detect dscp-based
 class MPLS-3
  bandwidth percent 20
  random-detect dscp-based
 class MPLS-2
  priority percent 15
 class MPLS-1
  bandwidth percent 5
  random-detect dscp-based
```

customers are sending traffic to the edge of MPLS core where MPLS label pushed with EXP value. The ingress PE router either copies DSCP from IP packets or uses their own policy, and ISP routers take EXP to determine how to treat packets at egress for WFQ, WRED, and other QoS services. When a packet is leaving MPLS core, it simply pops the label and expose the IP header for IP network QoS scheduling.

Conclusion

QoS is an essential approach to solve the issue of network congestion instead of best effort in traditional network when there is limited network resource to be allocated, and satisfy different requirements for various service such as real-time applications need low latency, less packet lost, and low jitter. Therefore, QoS can guarantee the quality for certain sensitive service. We have introduced three main QoS service modes, and mainly focused on DiffServ to spread many concepts behind it including classification, marking, congestion management, congestion avoidance, policing, and shaping. In addition, the examples of each of concepts have been given individually to test the implementation. For instance, WFQ shows how the weight, where stores in the IP header or MPLS EXP bits, affects the action of software queueing to schedule the packets, and the higher weight gets better treatment than others. Also, shaping and policing cooperate with each other to be able to facilitate ISP to implement over-subscription without huge cost of upgrading hardware. Over-subscription provides an efficiency of utilize network resource, and the customers can purchase the bandwidth they need and allow ISP to add charging to it. However, the bandwidth they can actually purchase should be less than UNI (users network interface, physical port) speed. There is a combination of each concept have been implemented in MPLS DiffServ tunneling modes in above content. For example, how ingress router limits the flow by policing and mark them to relevant EXP, different classes of traffic are randomly dropped and the sequence of scheduling packets in the queue when congestion occurs, and shaping the traffic at egress direction to help control the flow of traffic as well. Therefore, QoS is more effective and efficient to dedicate to control traffic flows and allocate resource to ensure minimum latency, jitter, and packet loss, and it work on serval layers' devices to fulfil different scenarios.

References

1. http://www.ciscopress.com/articles/article.asp?p=471096&seqNum=6
2. http://www.cisco.com/c/dam/en/us/products/collateral/ios-nx-os-software/quality-of-service-qos/prod_presentation0900aecd80312b59.pdf
3. https://en.wikipedia.org/wiki/Differentiated_services
4. http://www.slideshare.net/proydesa/cisco-qos
5. Flannagan, M. E., Froom, R., & Turek, K. (2003). Cisco Catalyst QoS. [electronic resource] : quality of service in campus networks. Indianapolis, Ind. : Cisco Press, c2003.
6. http://www.cisco.com/c/en/us/td/docs/nsite/enterprise/wan/wan_optimization/wan_opt_sg/chap05.html
7. http://www.cisco.com/c/en/us/products/collateral/ios-nx-os-software/network-based-application-recognition-nbar/product_bulletin_c25-627831.html
8. https://www.tucny.com/Home/dscp-tos
9. https://en.wikipedia.org/wiki/IEEE_P802.1p
10. https://networklessons.com/quality-of-service/ip-precedence-dscp-values/
11. http://www.cisco.com/c/en/us/support/docs/quality-of-service-qos/qos-policing/22833-qos-faq.html#class
12. http://h3c.com/portal/Products___Solutions/Products/Switches/H3C_S5120-EI_Series_Switches/White_Paper/200812/689004_57_0.htm#_Toc215923183
13. https://sites.google.com/site/amitsciscozone/home/qos/priority-queuing
14. https://en.wikipedia.org/wiki/Fair_queuing
15. http://www.cisco.com/c/dam/en/us/products/collateral/ios-nx-os-software/quality-of-service-qos/prod_presentation0900aecd80312b59.pdf
16. https://www.nanog.org/meetings/nanog36/presentations/sathiamurthi.pdf
17. http://sdiwc.net/digital-library/bandwidth-guarantee-using-class-based-weighted-fair-queue-cbwfq-scheduling-algorithm
18. https://www.maxcdn.com/one/visual-glossary/tcp-slow-start/
19. https://books.google.ca/books?id=MSGP9lEd5lAC&pg=PA67&lpg=PA67&dq=Introducing+Congestion+Avoidance+and+RED&source=bl&ots=djWkGALOXN&sig=mO3DAcypSN4Gi1eGZC-Tclo4xMw&hl=en&sa=X&ved=0ahUKEwjJ9r7lr7_LAhVM5GMKHeYmC4AQ6AEIUjAJ#v=onepage&q=Introducing%20Congestion%20Avoidance%20and%20RED&f=false
20. http://www.cisco.com/c/dam/en_us/training-events/le31/le46/cln/qlm/CCVP/qos/congestion-avoidance-introducing-red-and-wred/player.html
21. http://www.cisco.com/c/en/us/support/docs/quality-of-service-qos/qos-policing/19645-policevsshape.html

22. http://www.cisco.com/c/en/us/support/docs/multiprotocol-label-switching-mpls/mpls/47815-diffserv-tunnel.html
23. http://www.cisco.com/c/en/us/td/docs/solutions/Enterprise/WAN_and_MAN/QoS_SRND/QoS-SRND-Book/VPNQoS.html
24. https://books.google.ca/books?id=SIElAgAAQBAJ&pg=PA704&lpg=PA704&dq=qos+oversubscription&source=bl&ots=pKquqzRBkJ&sig=tkbOR926Y02H-UnNhVFTrwFNV44&hl=en&sa=X&ved=0ahUKEwiIwPbn4tLLAhVEt4MKHZSwDjoQ6AEIPzAG#v=onepage&q=qos%20oversubscription&f=false
25. https://ccdewiki.wordpress.com/2013/05/31/three-qos-models-in-mpls/
26. https://www.mef.net/Assets/White_Papers/Understanding_MEF_6.2_Bandwidth_Profiles_FINAL.pdf
27. https://www.mef.net/Assets/White_Papers/Bandwidth-Profiles-for-Ethernet-Services.pdf
28. http://blog.ine.com/2008/08/26/understanding-the-shape-peak-command/#more-236
29. https://networklessons.com/quality-of-service/qos-traffic-shaping-explained/

Figures references

Figure No.	Content	Reference
Figure 1-1	Traditional networks before converged	http://www.slideshare.net/proydesa/cisco-qos
Figure 1-2	Converged networks	http://www.slideshare.net/proydesa/cisco-qos
Figure 1-3	Differentiated Service Architecture	https://www.nanog.org/meetings/nanog36/presentations/sathiamurthi.pdf
Figure 3-1	Traffic classification	http://www.slideshare.net/proydesa/cisco-qos
Figure 3-2	DiffServ in Ethernet frame header	http://www.cisco.com/c/dam/en/us/products/collateral/ios-nx-os-software/quality-of-service-qos/prod_presentation0900aecd80312b59.pdf/
Table 3-3	CoS value for various types of service	
Figure 3-4	DiffServ in MPLS header	http://www.cisco.com/c/dam/en/us/products/collateral/ios-nx-os-software/quality-of-service-qos/prod_presentation0900aecd80312b59.pdf
Table 3-5	EXP value for various types of service	http://babaawesam.com/2013/08/07/mpls-vpn-qos-with-gns3-and-virtualbox/
Figure 3-6	DiffServ in IP header	http://www.slideshare.net/proydesa/cisco-qos
Figure 3-7	PHB defined in IPv4 header	https://en.wikipedia.org/wiki/Type_of_service
Figure 3-8	Example of DSCP value for EF PHB	https://en.wikipedia.org/wiki/Type_of_service

Figure 3-9	Example of DSCP value for AF PHB	https://en.wikipedia.org/wiki/Type_of_service
Table 3-10	AF behavior group	https://en.wikipedia.org/wiki/Differentiated_services
Table 3-11	Mapping between IPP, DSCP, and CoS	https://www.tucny.com/Home/dscp-tos
Figure 4-1	Congestion management example	
Figure 4-2	FIFO queueing	http://h3c.com/portal/Products___Solutions/Products/Switches/H3C_S5120-EI_Series_Switches/White_Paper/200812/689004_57_0.htm#_Toc215923183
Figure 4-3	FIFO applied on interface	
Figure 4-4	Priority queueing	http://h3c.com/portal/Products___Solutions/Products/Switches/H3C_S5120-EI_Series_Switches/White_Paper/200812/689004_57_0.htm#_Toc215923183
Figure 4-5	Scenario for PQ testing	
Figure 4-7	Show default queueing priority	
Figure 4-8	Show queueing priority	
Figure 4-9	Assign priority list to interface	
Figure 4-10	Show interface to see the effect	
Figure 4-11	Priority queueing testing result	
Figure 4-12	Priority queueing testing result	
Figure 4-13	Priority queueing testing result	

Figure 4-14	Dropping policy in WFQ	http://blog.globalknowledge.com/technology/unified-communications/quality-of-service-part-10-%E2%80%93-weighted-fair-queuing/
Figure 4-15	WFQ scheduling	http://virtualrack.blogspot.ca/2011/11/notes-wfq.html
Figure 4-16	Scenario of WFQ	
Figure 4-17	Show policy-map	
Figure 4-18	Show queue on interface which applied to WFQ	
Figure 4-19	Show interface to see queueing strategy	
Figure 4-20	Create class-map to match possible IPP	
Figure 4-21	Show policy-map interface to see the matchups	
Figure 4-22	CBWFQ scheduling	http://lh3.ggpht.com/-gOaonvekK0U/TrK2vDx47BI/AAAAAAAAANs/v0G0oU-Ha70/s1600-h/image%25255B4%25255D.png http://slideplayer.com/slide/9996714/#
Figure 4-23	CBWFQ testing	
Figure 4-24	Class-map and policy-map created on router	
Figure 4-25	Show policy-map interface	
Figure 4-26-a	Throughput testing	
Figure 4-26-b		
Figure 4-28	LLQ scheduling	http://core0.staticworld.net/images/idge/imported/article/nww/2008/03/07fig05-100279669-orig.jpg
Table 4-29	Queueing mechanism comparison	http://h3c.com/portal/Products___Solutions/Products/Switches/H3C_S5120-

		EI_Series_Switches/White_Paper/200812/689004_57_0.htm#_Toc215923183
Figure 5-1	the utilization variation in TCP synchronization	http://slideplayer.com/slide/9996714/#
Figure 5-2	The comparison of RED enabled before and after	http://slideplayer.com/slide/9996714/#
Figure 5-3	the relation between IPP value and drop probability	http://player.myshared.ru/9/885523/data/images/img14.jpg
Figure 5-4	the relation between DSCP value and drop probability	http://player.myshared.ru/9/885523/data/mages/img17.jpg
Figure 5-5	WRED working procedure	http://player.myshared.ru/9/885523/data/images/img13.jpg
Figure 5-6	Scenario for WRED	
Figure 5-7	Different flow of IPP	
Figure 5-8	Create access-list	
Figure 5-9	Show class-map has been created	
Figure 5-10	policy-map for WRED	
Figure 5-11	Throughput of WRED	
Figure 5-12	Show policy-map with random-detect	
Figure 5-13	Different flow of DSCP	
Figure 5-14-a	Access-list	
Figure 5-14-b	Class-map to match access-list	
Figure 5-14-c	Policy-map to random-detect dscp-based	
Figure 5-15	Show policy-map	

Figure 5-16	Throughput of random-detect	
Figure 5-17	Show policy-map interface to see the result	
Figure 6-1	Single token bucket, single rate, two colors	http://www.cisco.com/c/dam/en/us/products/collateral/ios-nx-os-software/quality-of-service-qos/prod_presentation0900aecd80312b59.pdf
Figure 6-2	dual token buckets, single rate, three colors	http://www.cisco.com/c/dam/en/us/products/collateral/ios-nx-os-software/quality-of-service-qos/prod_presentation0900aecd80312b59.pdf
Figure 6-3	dual token buckets, dual rates, three colors	http://www.cisco.com/c/dam/en/us/products/collateral/ios-nx-os-software/quality-of-service-qos/prod_presentation0900aecd80312b59.pdf
Figure 6-4	a difference after policing and before	http://www.cisco.com/c/en/us/support/docs/quality-of-service-qos/qos-policing/19645-policevsshape.html
Figure 6-5	Scenario of CB-policing	
Figure 6-6	Configurations of class-map and policy-map for 1 bucket and 1 rate	
Figure 6-7	Show policy map to see the matchups for 1 bucket, 1 rate	
Figure 6-8	Throughput of CB-policing for 1 bucket and 1 rate, bucket size 1	
Figure 6-9	Throughput of CB-policing for 1 bucket and 1 rate, bucket size 2	

Figure 6-10	Show policy-map for 1 bucket, 1 rate	
Figure 6-11	Show policy-map for 1 rate and 2 buckets	
Figure 6-12	Throughput of CB-policing for 2 buckets and 1 rate	
Figure 6-13	Wireshark to check exceed traffic DSCP value	
Figure 6-14	Configurations of 2 buckets, 2 rates	
Figure 6-15	Show policy-map for 2 buckets, 2 rates	
Figure 6-16	Throughput of CB-policing for 2 buckets and 2 rates	
Figure 6-17	Shaping with time arrangement	https://networklessons.com/quality-of-service/qos-traffic-shaping-explained/
Figure 6-18	Shaping algorithm	http://www.cisco.com/c/en/us/td/docs/ios/12_2/qos/configuration/guide/fqos_c/qcfpolsh.html
Figure 6-19	a difference after shaping and before	http://www.cisco.com/c/en/us/support/docs/quality-of-service-qos/qos-policing/19645-policevsshape.html
Figure 6-20	Scenario of CB-shaping	
Figure 6-21-a	Assign policy-map to interface	
Figure 6-21-b	Class-map to match ipp1	
Figure 6-21-c	Show policy-map for average rate	
Figure 6-22	Show policy-map interface for shaping average rate matchups	

Figure 6-23	Throughput of shaping for average rate	
Figure 6-24	Throughput of shaping for average rate with another traffic rate	
Figure 6-25	Show policy-map for peak rate	
Figure 6-26	Show policy-map interface for shaping peak rate matchups	
Figure 6-27	Throughput of shaping for peak rate	
Figure 6-28	Throughput of shaping for peak rate with another traffic rate	
Figure 6-29	Configurations for multiple layers' policy-map	
Figure 6-30	Throughput of multiple layers' policy-map	
Figure 6-31	token bucket algorithm for over-subscription	http://www.cisco.com/c/dam/en/us/products/collateral/ios-nx-os-software/quality-of-service-qos/prod_presentation0900aecd80312b59.pdf
Table 6-32	Ingress and egress bandwidth profiles	https://www.mef.net/Assets/White_Papers/Understanding_MEF_6.2_Bandwidth_Profiles_FINAL.pdf
Figure 6-33	Per-UNI Ingress bandwidth profile	https://www.mef.net/Assets/White_Papers/Bandwidth-Profiles-for-Ethernet-Services.pdf
Figure 6-34	Configuration for EVC ingress bandwidth profile	

Figure 6-35	Per-EVC Ingress bandwidth profile	https://www.mef.net/Assets/White_Papers/Bandwidth-Profiles-for-Ethernet-Services.pdf
Figure 6-36	Configuration for DSCP ingress bandwidth profile	
Figure 6-37	Per-DSCP Ingress bandwidth profile	https://www.mef.net/Assets/White_Papers/Bandwidth-Profiles-for-Ethernet-Services.pdf
Figure 6-38	Policy-map for peak shaping	https://www.mef.net/Assets/White_Papers/Bandwidth-Profiles-for-Ethernet-Services.pdf
Figure 6-39	Scenario for combination of shaping and policing for over-subscription	
Figure 6-40	Show policy-map on CE1	
Figure 6-41	Show policy-map on PE1	
Figure 6-42	Throughput of combination	
Figure 6-43	Show policy-map interface on PE1	
Figure 6-44	Show policy-map interface on CE1	
Figure 6-45	Show policy-map interface on CE1-B	
Figure 6-46	CB-WFQ	
Figure 6-47	PC1 throughput	
Figure 6-48	PC2 throughput	
Table 7-1	Default mapping between DiffServ PHB and EXP	
Figure 7-2	The illustration explains DiffServ and EXP exchange between each other	http://h3c.com/portal/Products___Solutions/Products/Switches/H3C_S5120-EI_Series_Switches/White_Paper/200812/689004_57_0.htm#_Toc215923183

Figure 7-3	The basic architecture of MPLS VPN network	http://www.cisco.com/c/en/us/support/docs/multiprotocol-label-switching-mpls/mpls/47815-diffserv-tunnel.html
Figure 7-4	Uniform mode	http://www.cisco.com/c/en/us/support/docs/multiprotocol-label-switching-mpls/mpls/47815-diffserv-tunnel.html
Figure 7-5	Connectivity from CE1 to CE2	
Figure 7-6	Show ip route on CE1-A	
Figure 7-7-a	On PE1 debug mpls packet and wireshark to see the label between PE1 and P1	
Figure 7-7-b		
Figure 7-8-a	On P1: the topmost label has changed, wireshark between P1 and P2	
Figure 7-8-b		
Figure 7-9	Debug mpls packets on P2	
Figure 7-10-a	On PE2: the EXP bits has been swapped to 4 according to configuration. Wireshark between P2 and PE2	
Figure 7-10-b		
Figure 7-11	the packet left PE2 and DSCP has been changed to AF43	
Figure 7-12	policy-maps on CE routers	
Figure 7-13	Pipe mode	http://www.cisco.com/c/en/us/support/docs/multiprotocol-label-switching-mpls/mpls/47815-diffserv-tunnel.html
Figure 7-14	Cisco provides a five service EXP mapping for MPLS core network	http://www.cisco.com/c/en/us/td/docs/solutions/Enterprise/WAN_and_MAN/QoS_SRND/QoS-SRND-Book/VPNQoS.html
Figure 7-15	Representation of DSCP for different service mapping to	

	EXP 3 bits, customized table	
Figure 7-16	On P1 debug mpls packet	
Figure 7-17	Wireshark between PE1 and P1	
Figure 7-18	On P2 debug mpls packet	
Figure 7-19	Wireshark between P1 and P2	
Figure 7-20	On PE2 debug mpls packet	
Figure 7-21	Wireshark between P2 and PE2	
Figure 7-22	Short pipe mode	http://www.cisco.com/c/en/us/support/docs/multiprotocol-label-switching-mpls/mpls/47815-diffserv-tunnel.html