QoS (Quality of Service)

Quality of Service Models

Best Effort · No QoS policies are implemented

Integrated Services (IntServ)
Resource Reservation Protocol (RSVP) is used to reserve bandwidth per-flow across all nodes in a path

Differentiated Services (DiffServ)
Packets are individually classified and marked; policy decisions are made independently by each node in a path

Why is QoS Needed?

- Root Cause: Resource Contention
  - Multiple flows sharing the same link
  - Same or multiple applications
  - Each application has its own requirements
- Contention results in Queueing
  - Packets may be delayed or dropped
  - Effective flow throughput decreases
  - Delay or Jitter may exceed thresholds

Quality of Service (QoS)

- *Quality of service is the ability to provide different priority to different applications, users, or data flows, or to guarantee a certain level of performance to a data flow.*
- E.g. different service levels for different types or "classes" of traffic flows

Possible Solutions

- Best Solution: Avoid Contention
  - Don't over-provision
  - Not always possible
- Next Best Solution: QoS
  - Network congestion is controlled
  - Delay/Loss/Jitter/Throughput are controlled
  - Only alleviates temporary congestion

Precedence/DSCP

	Binary	DSCP	Prec.
56	111000	Reserved	7
48	110000	Reserved	6
46	101110	EF	5
32	100000	CS4	4
34	100010	AF41	
36	100100	AF42	
38	100110	AF43	
24	011000	CS3	3
26	011010	AF31	
28	011100	AF32	
30	011110	AF33	
16	010000	CS2	2
18	010010	AF21	
20	010100	AF22	
22	010110	AF23	
8	001000	CS1	1
10	001010	AF11	
12	001100	AF12	
14	001110	AF13	
0	000000	BE	0

RFC 1633 - Integrated Services in the Internet Architecture: an Overview

What is IntServ?

- Connection-oriented model
- Every flow has an explicit reservation end-to-end
- Does not scale well because network must maintain too much state

IntServ use case is MPLS TE

- RFC 2205 - Resource ReSerVation Protocol (RSVP)
- RFC 3209 - RSVP-TE: Extensions to RSVP for LSP Tunnels

RFC 2475 - An Architecture for Differentiated Services

What is DiffServ?

- Connectionless model
- Traffic is grouped into classes
- QoS behavior is defined by traffic's class
- Called Per-Hop Behavior (PHB)

Classification & Marking

- In order for DiffServ to properly work, traffic must be placed into correct classes
  - I.e. "Classification"
- Traffic classification normally occurs at network ingress edge
  - Typically a manual process we must enforce
- Classification can be encoded inside packet itself
  - Known as packet's "marking"

Classification Types

- Classification & Marking can happen at multiple places
- Layer 2 Class of Service (CoS)
  - 802.1q Ethernet header
- Layer 3 IP Type of Service (ToS)
  - IP Precedence & Differentiated Services Code Point (DSCP)
- Layer 4
  - TCP & UDP ports
- Upper layers
  - Network Based Application Recognition (NBAR)
  - Deep Packet Inspection (DPI)

7 - Reserved
6 - Reserved
5 - Voice Bearer
4 - Videoconferencing
3 - Call Signaling — RTCP
2 - High-priority Data — Citrix, SNA
1 - Medium-priority Data — SQL, AD
0 - Best-Effort Data — P-2-P, WEB

QoS Tools

- Used to Implement QoS Models
  - Many tools rely on correct QoS classification & marking
- Different Tools for
  - Network Edge
  - Network Core
- Tools fall into three main categories
  - Admission Control
  - Congestion Management
  - Congestion Avoidance

Traffic Shaping

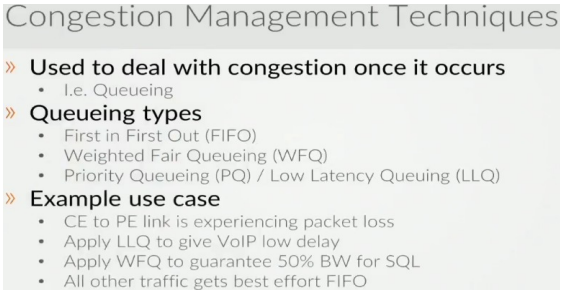
- Used to normalize outbound traffic flow
  - Smooth out traffic bursts
  - Prepares traffic for ingress policing
  - Delay and Queue exceeding traffic
- Example use case
  - PE connects to CE with GigE port
  - Circuit is provisioned at 250Mbps
  - CE applies outbound shaper at port level
    - If traffic <= 250Mbps, transmit
    - If traffic > 250Mbps, queue for later transmission

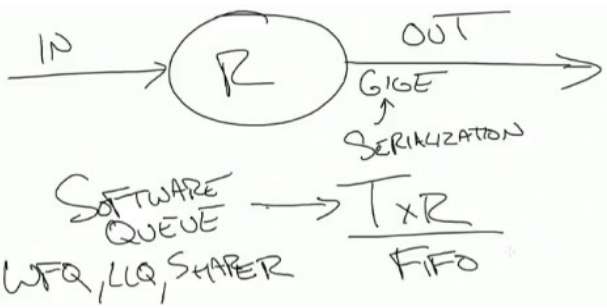
Traffic Policing

- Used to limit inbound and outbound traffic flows
  - Traffic that exceeds the rate can be dropped, marked, or re-marked
  - Typically applied on ingress edge
- Example use case
  - PE connects to CE with GigE port
  - Circuit is provisioned at 250Mbps
  - PE applies inbound policer at port level
    - If traffic <= 250Mbps, transmit
    - If traffic > 250Mbps, drop

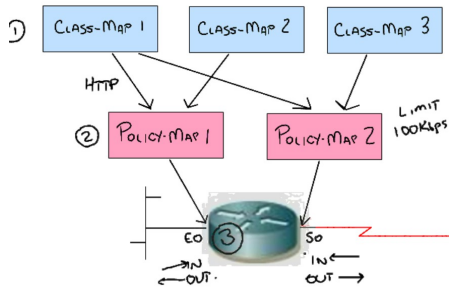
Congestion Avoidance Techniques

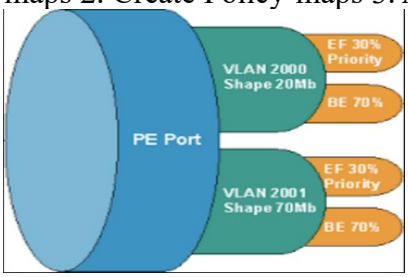
- Try to prevent congestion before it occurs
  - I.e. packet drop strategy
- Drop strategy types
  - Weighted Random Early Detection (WRED)
  - Tail Drop
- Example use case
  - CE to PE link is experiencing packet loss
  - Apply WRED to selectively drop low priority TCP flows
  - Senders go into TCP Slow Start
  - Congestion management is offloaded to the end host



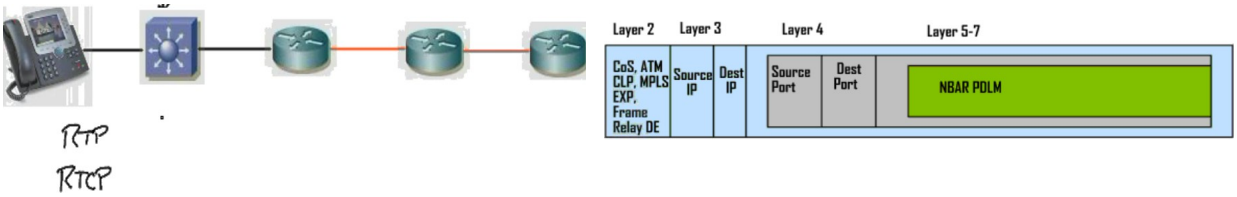


Things that require QoS: Data/Voice(G.711 (64 Kbps))/Video
Problems: Lack of bandwidth/packet loss/delay(150ms one way delay)/jitter(delay variation)
Methods of QoS: CLI/MQC(Modular QoS CLI)->HQF (Hierarchical Queuing Framework)/Auto QoS/QPM(QoS Policy Manager)
Classification: Involves identifying (inspecting packets) and grouping different traffic types.
Marking: Tag or write the packets so it can be quickly recognized elsewhere in the network.
Policing: Drops or marks packet when limit is reached.
Shaping: Queues packet when limit is reached.
Congestion Avoidance Tools: FIFO(First In First Out) + Tail Drop/RED (Random Early Detection)/WRED (Weighted RED) Congestion Management: Queuing (TxR is Transmit Ring i.e. hardware queue)
Link Efficiency Tools: Compression/LFI (Link Fragmentation and Interleaving)
QoS with the MQC: 1. Create Class-maps 2. Create Policy-maps 3. Apply to an interface in or out





Class-maps (Classification): (There is always a default class-map that is match-any)
R(config)#class-map web-traffic !(by default it takes it as match-all | more like and AND)
R(config-cmap)#match protocol http !(NABR (Network Based Application Recognition))
R(config-cmap)#match packet length min 400 max 600 !(in bytes)
R(config)#class-map match-any web-traffic !(match-any is more like an OR)
R(config)#class-map match-any ftp-traffic
R(config-cmap)#match protocol ftp
R(config)#access-list 5 permit host 192.1681.9
R(config)#class-map host
R(config-cmap)#match access-group 5 !(to match access-list)
Policy-maps: (One policy can be used for many classes)
R(config)#policy-map LIMIT_HTTP
R(config-pmap)#class web-traffic
R(config-pmap-c)#police 500000 !(in kbps)
R(config-pmap)#class ftp-traffic
R(config-pmap-c)#bandwidth 500 !(guarantee bandwidth in kbps)
R(config-pmap)#class class-default
R(config-pmap-c)#fair-queue
R(config-pmap-c)#random-detect
Apply the Policy: (One policy map per interface per direction can be applied)
R(config)#int s0
R(config-if)#service-policy input LIMIT_HTTP

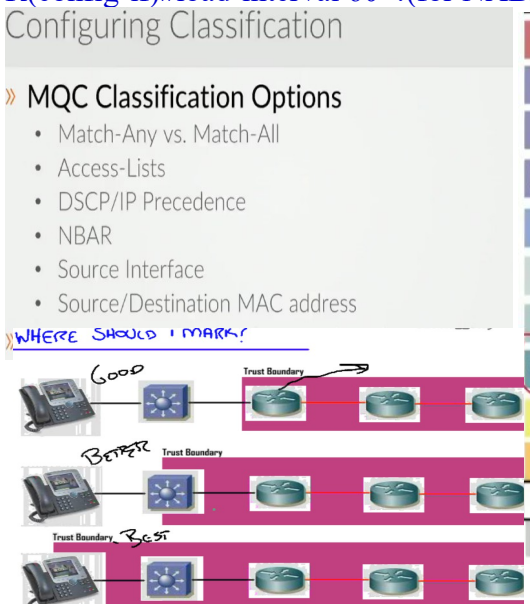


RTP (Real-time Transfer Protocol) | RTCP (Real-time Transfer Control Protocol)

R(config)#class-map test
R(config-cmap)#description This is a test class
R(config-cmap)#match any
R(config-cmap)#match not access-group 10 !(match anything that does not include access-list 10)

NBAR (Network Based Application Recognition):

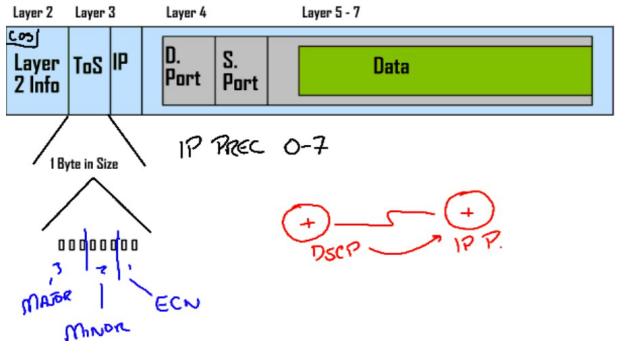
R(config)#class-map test
R(config-cmap)#match protocol rtp
!(NBAR matches based on application signatures i.e. DPI (Deep packet inspection))
!(avoids using access-list which is only ports based and doesn't work for dynamic apps)
!(PDLM (Packet Description Language Modules) i.e. application signatures)
R(config)#int s0
R(config-if)#ip nbar protocol discovery !(to turn the NBAR ON)
R(config-if)#load-interval 60 !(for NABR to discover every 1 min instead of default 5 mins)

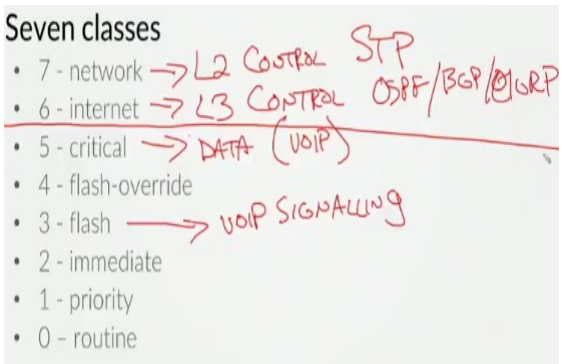


VoIP Telephony	EF	Required	Priority Queue (PQ)
Broadcast Video	CS5	Required	(Optional) PQ
Real-Time Interactive	CS4	Required	(Optional) PQ
Multimedia Conferencing	AF4	Required	BW Queue + DSCP WRED
Multimedia Streaming	AF3	Recommended	BW Queue + DSCP WRED
Network Control	CS6	EIGRP/OSPF	BW Queue
Signaling	CS3	H323, SIP	BW Queue
Ops/Admin/Mgmt (OAM)	CS2	SYSLOG/SNMP	BW Queue
Transactional Data	AF2	SQL/HTTP	BW Queue + DSCP WRED
Bulk Data	AF1		BW Queue + DSCP WRED
Best Effort	DF		Default Queue + RED
Scavenger	CS1		Min BW Queue

Types of Marking:

Layer 2:Typically stripped at router hops e.g.CoS 3bits/FR DE 1bit/ATM CLP 1bit/MPLS Exp 3bits
Layer 3: Passes through routers e.g. ToS bytes: IP Precedence 3bits/DSCP (Differentiated Services Code Point) 6bits



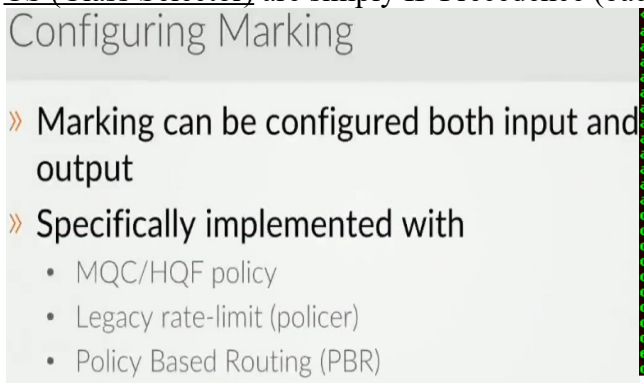


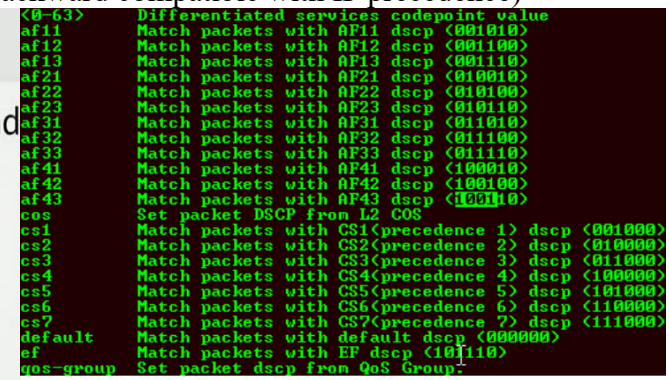
R(config)#class-map IP_PHONE_AUDIO
R(config-cmap)#match protocol rtp
R(config)#policy-map MARK_VOICE

R(config-pmap)#class IP_PHONE_AUDIO
R(config-pmap-c)#set ip precedence critical !(set to value of 5 | Voice Bearer)
R(config-pmap-c)#set ip dscp ef !(dscp value for VoIP | Expedited Forwarding)
R(config)#int fa0/1
R(config-if)#service-policy input MARK_VOICE !(marking is done on inbound direction)

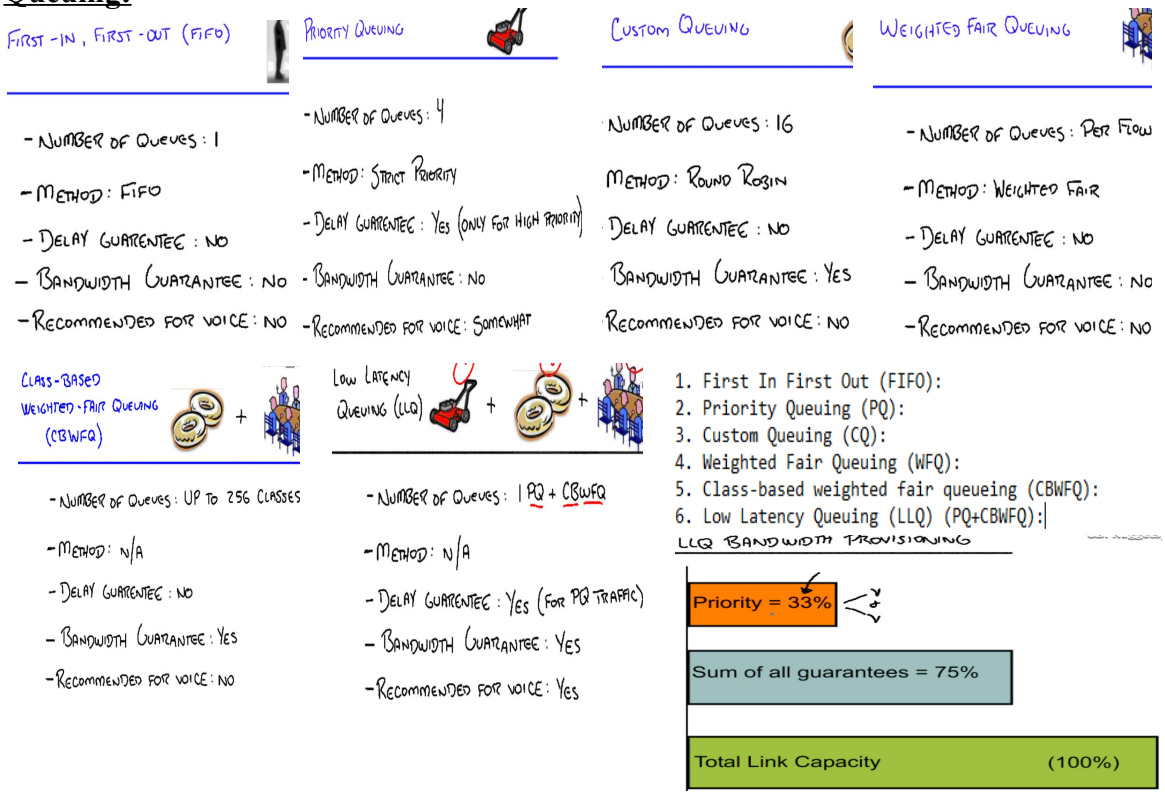
Layer 3: 1-byte ToS (Type of Service) DSCP (Differentiated Services Code Point) Marking:

IP Precedence can only use first 3-bits from the ToS byte where as DSCP can use the whole.
DSCP: 3 bits are major(higher is better)+3 bits are minor (drop preference)(higher is worse)+last 2 bits are ECN (Explicit Congestion Notification)
Default – 0 (no minor or drop preference)
AF (Assured Forwarding) AF12 where 1 is major and 2 is minor
EF (Expedited Forwarding) Value 5 (VoIP) – (no minor or drop preference)
CS (Class Selector) are simply IP Precedence (backward compatible with IP precedence)





Queuing:



Priority 33% is strict policed priority and wont go above this.
33%(voice and video)+42%(http+https+ftp etc.)=75% total
25% is always left for other (not classified) traffic.
(Ethernet interface has 65% rule)
(NOTE: NOT REQUIRED ANYMORE! with HQF/HQoS. Now it is 99% and only 1% is left for default class)
R2(config-if)#max-reserved-bandwidth 100 !(to bypass max 65% or 75% bandwidth limit)

FIFO Queueing

- Simplest and easiest to implement
  - Only parameter is queue-depth
- Configuration
  - Disable previous queueing strategy
    - I.e. no fair-queue
  - Define queue depth
    - hold-queue out
- Typically used as part of other solutions
  - E.g. CBWFQ/HQF

Fair Queueing

- Also known as max-min scheduling
- Services multiple requests for a shared resource
  - Step 1: Share resource equally
  - Step 2: Take excessive amounts
  - Step 3: Share excess equally among unsatisfied requests

Weighted Fair Queueing

- Max-min scheduling, but not equal
  - Allocate bandwidth per flow proportional to weight
- Flow is defined dynamically
  - Src/Dst IP + Src/Dst Port + ToS Byte
- Weight is IP Precedence + 1

Weighted Fair Queueing

- Configuration
  - fair-queue <CDT> <QUEUES>
  - hold-queue out <MAX BUFFERS>
- Congestive Discard Threshold (CDT)
  - Individual queue size threshold
- If number of flows > number of queues…
  - Flow collision occurs and queues are shared

CBWFQ/HQF

- Allows defining of custom flows
  - Class definition using MQC Syntax
  - bandwidth keyword defines class's "weight"
- Bandwidth is shared proportionally to weight
  - Relative sharing, not absolute reservation

CBWFQ/HQF (cont.)

- Every Queue in HQF is FIFO
  - Includes class-default
  - Buffer-limit with queue-limit command
    - Global buffer limit with hold-queue out
  - Can be turned into Fair-Queue
    - Command fair-queue <FLOWS>
    - All flows are equal, no weighting
    - Queue limit per flow is 1/4*queue-limit

CBWFQ/HQF (cont.)

- Reservations
  - Absolute with bandwidth [Kbps]
  - Relative with bandwidth percent [%]
    - Percent of interface "bandwidth" setting
  - All bandwidths must sum to interface "bandwidth"
- Class-Default
  - Always guaranteed at least 1% of interface BW
    - max-reserved-bandwidth now deprecated

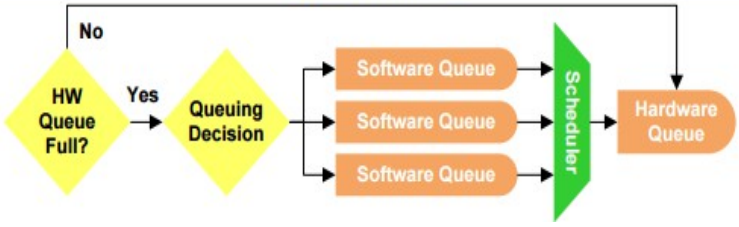
LLQ in HQF

- Priority Queue
  - Only one per HQF configuration
    - Designated with priority [X]
    - Always emptied first
  - Optionally policed to X Kbps only in times of congestion
    - Congestion defined as having TX-Ring full
  - Multiple classes can have priority
    - Share single queue but could be policed differently

LLQ in HQF (cont.)

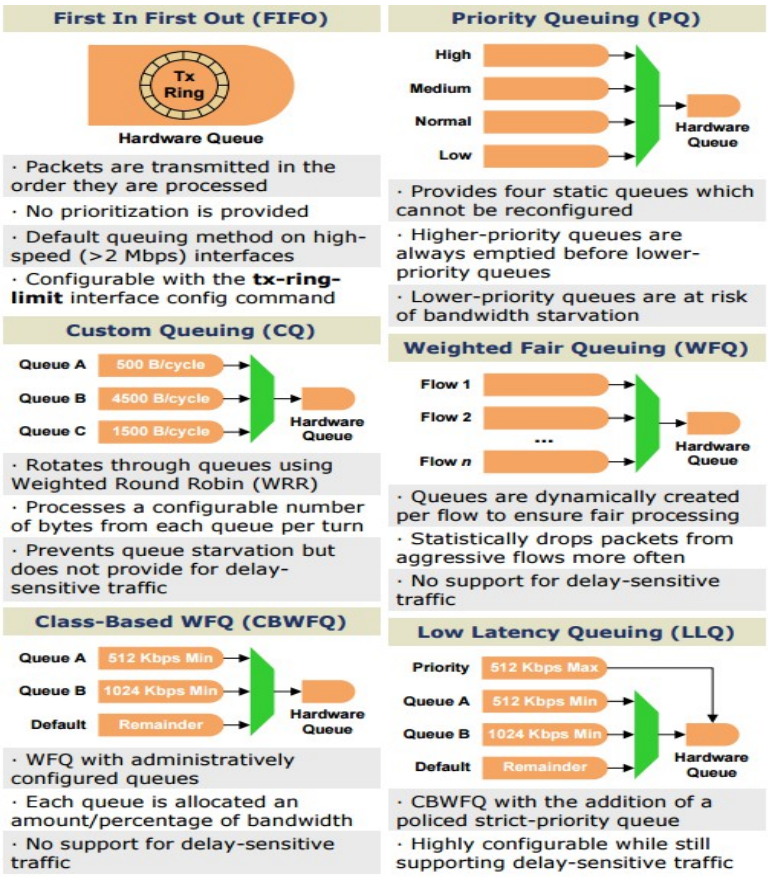
- Remaining Bandwidth
  - Commonly used with LLQ
  - Bandwidth remaining after LLQ allocations
  - Command bandwidth remaining X
  - Calculated as Interface_BW - LLQ_BW

QoS Flowchart



Queuing Comparison

	FIFO	PQ	CQ	WFQ	CBWFQ	LLQ
Default on Interfaces	>2 Mbps	No	No	<=2 Mbps	No	No
Number of Queues	1	4	Configured	Dynamic	Configured	Configured
Configurable Classes	No	Yes	Yes	No	Yes	Yes
Bandwidth Allocation	Automatic	Automatic	Configured	Automatic	Configured	Configured
Provides for Minimal Delay	No	Yes	No	No	No	Yes
Modern Implementation	Yes	No	No	No	Yes	Yes

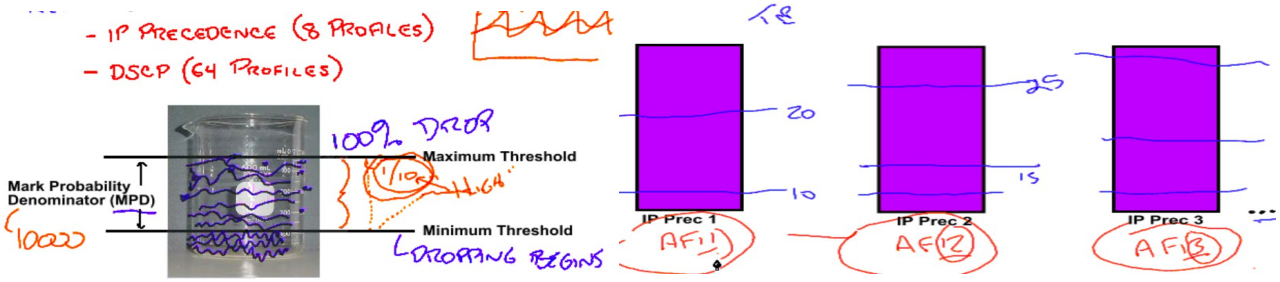


LLQ (PQ+CBWFQ):

```
R3(config)#policy-map LLQ
R3(config-pmap)#class VoIP
R3(config-pmap-c)#priority percent 10
R3(config-pmap-c)#class HTTP
R3(config-pmap-c)#bandwidth remaining percent 10
R3(config-pmap-c)#class SMTP
R3(config-pmap-c)#bandwidth remaining percent 50
R3(config-pmap-c)#class class-default
R3(config-pmap-c)#bandwidth remaining percent 40
R3(config-pmap-c)#fair-queue
R3(config-pmap-c)#int fa0/0
R3(config-if)#service-policy output LLQ
!(Congestion management only happens when there is congestion on the link)
```

Tail Drop Flaws: 1. TCP Synchronization 2. Traffic Starvation 3. Unbiased Dropping

RED (Random Early Detection): Randomly drops packets from tcp flows to minimize TCP synchronization. Dropping becomes more aggressive as the queue fills.



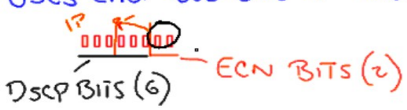
WRED (Weighted Random Early Detection): Allows multiple RED profiles.
Different queues of WRED to different classes of traffic e.g. different IP Precedence.

```
R3(config-pmap-c)#random-detect    !(uses Cisco default values | IP Precedence based)
```

R3(config-pmap-c)#random-detect dscp-based !(recommended to use Cisco default values)
R3(config-pmap-c)#random-detect dscp-based af11 10 50 10
!(10 is min threshold 50 is max threshold and 10 is MPD)

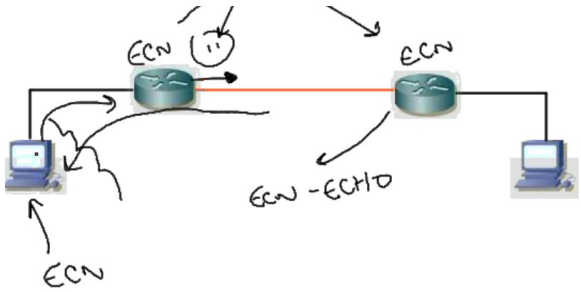
WRED ECN (Explicit Congestion Notification) Enhancements:

- Explicit Congestion Notification (ECN) adds a "proactiveness" to WRED
- Uses last two bits of ToS byte



- Has one of four markings

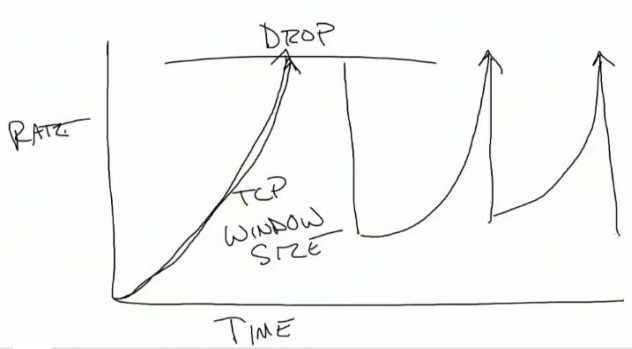
00 - Not ECN-Capabile
01 - Endpoints are ECN Capabile
10 - Endpoints are ECN Capabile
11 - Congestion Experienced



R3(config-pmap-c)#random-detect ecn !(makes the router ECN compatible)
!(ECN needs to be enabled on the PCs for it to work)

Tail Drop

- By default, all queues use tail drop
  - When the queue is full, new packets trying to enter the tail of the queue are denied admission
- Tail drop treats all packets equally
  - No classification is performed
- Tail drop can result in global TCP synchronization
  - Segments from lots of TCP flows are simultaneously dropped
  - Senders all go into TCP slow start at the same time
  - Result is saw-tooth traffic pattern



Random Early Detection

- RED is a congestion avoidance technique
  - Selectively drop flows from the queue before the buffer is 100% full
  - Goal is to send individual senders into slow start
  - Result is more even traffic patterns
- WRED adds weighting to drop algorithm
  - Packets with higher weight are less likely to be dropped

How WRED Works

- WRED tracks average queue depth
  - Smoothened based on weight factor
  - avg=(old_avg*(1-1/2^n))+(q_size*1/2^n)
- Drops packets based on Mark Probability Denominator
  - Probability = 1/Mark_Probability_Denominator
  - Drop probability increases as queue depth increases
  - If queue depth exceeds maximum, tail drop occurs
- Configured as **random-detect**
  - Can be combined with other queueing mechanisms

Link Efficiency Tools:

1. Header Compression: 1. For Voice 2. For Data
2. LFI Tools: 1. For PPP Links 2. For Frame Relay

Compression

- Two major types:
  - Payload Compression
    - Through codecs and traditional compression types (SUP)
  - Header Compression
    - Through features such as RTP/TCP header compression

1. G.711 - 64K
   G.723 - 6.3K
2. G.729 - (8K)
   - STAC (Predictor)
   - LZS (MEM)
   - MPPC (MFT)

Header Compression

TCP Header Compression (Telnet)

- Compress TCP headers

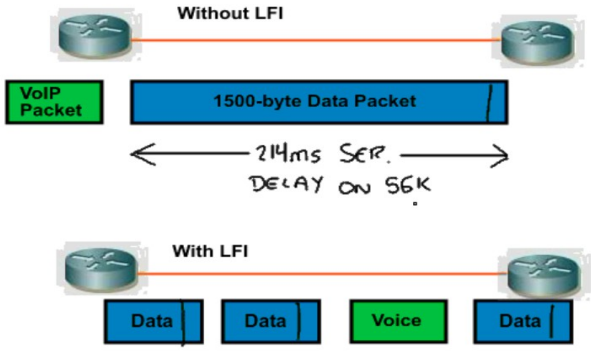
40byte
2-4 bytes

RTP Header Compression (cRTP)

- Compress Real-Time Transport Protocol (RTP) headers

R3(config)#class-map IP_PHONE_AUDIO
R3(config-cmap)#match protocol rtp
R3(config)#policy-map COMPRESS_VOICE
R3(config-pmap)#class IP_PHONE_AUDIO
R3(config-pmap-c)#compression header ip rtp !(IPHC: IP Header Compression)
!(You will have to compress the rtp headers on both sides otherwise the calls fail)

LFI (Link Fragmentation and Interleaving):

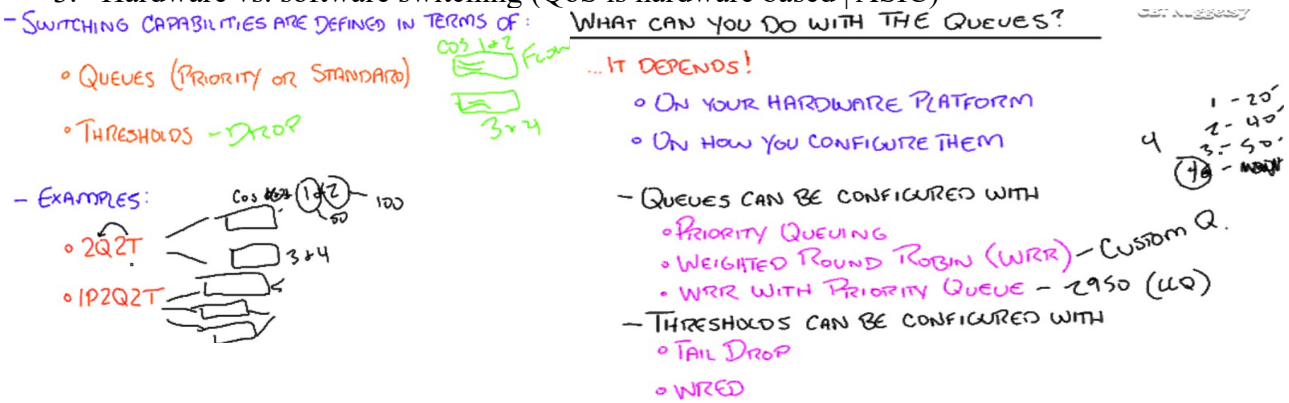


	1 Byte	64 Bytes	128 Bytes	256 Bytes	512 Bytes	1024 Bytes	1500 Bytes
56 kbps	143 us	9 ms	18 ms	36 ms	72 ms	144 ms	214 ms
64 kbps	125 us	8 ms	16 ms	32 ms	64 ms	128 ms	187 ms
128 kbps	62.5 us	4 ms	8 ms	16 ms	32 ms	64 ms	93 ms
256 kbps	31 us	2 ms	4 ms	8 ms	16 ms	32 ms	46 ms
512 kbps	15.5 us	1 ms	2 ms	4 ms	8 ms	16 ms	23 ms
768 kbps	10 us	640 us	1.28 ms	2.56 ms	5.1 ms	10.2 ms	15 ms
1536 kbps	5 us	320 us	640 us	1.28 ms	2.56 ms	5.12 ms	7.5 ms

Do not enable LFI on links greater than 768 kbps
R3(config)#int s0/0
R3(config-if)#encapsulation ppp
R3(config-if)#ppp multilink
R3(config-if)#no ip address
R3(config-if)#ppp multilink group 1
R3(config)#int multilink 1
R3(config-if)#ip address 10.1.1.1 255.255.255.0
R3(config-if)#bandwidth 768 !(in kbps)
R3(config-if)#ppp multilink fragment delay 10 !(ms)
R3(config-if)#ppp multilink interleave
R3(config-if)#ppp multilink group 1
R3#sh ppp multilink

Implementing QoS on Catalyst Switches:

1. Allows traffic to be classified+marked as close to the source as possible
2. Policing preformed as close to the source as possible (and more options)
3. Hardware vs. software switching (QoS is hardware based | ASIC)



Model 2950: 1P3Q | 4Q + Model 3550: 1P3Q2T | 4Q2T + Model 3750: 4Q3T +Model 4000: 1P3Q2T | 4Q2T + Model 6500: 2Q2T | 1P2Q2T | 1P3Q1T | 1P2Q1T
3550(config)#mls qos !(turning ON qos on a switch)
3550(config)#int fa0/1
3550(config-if)#mls qos trust device cisco-phone !(CDP needs to be enabled)
3550(config-if)#mls qos trust device ip-precedence

3550(config-if)#mls qos cos 5
3550(config-if)#auto qos voip cisco-phone !(options: cisco-phone | cisco-softphone)

WRR (Weighted Round Robin):

3550(config-if)#wrr-queue cos-map 1 0 1 2 !(CoS values of 0, 1 and 2 goes into queue 1)
3550(config-if)#wrr-queue cos-map cos-map 2 3 4
3550(config-if)#wrr-queue cos-map cos-map 3 6 7
3550(config-if)#wrr-queue cos-map cos-map 4 5
3550(config-if)#wrr-queue bandwidth 5 6 10 1 !(Queue#1 is 5 and so on... | Relative values)
3550(config-if)#priority-queue out !(the last queue becomes the priority queue)

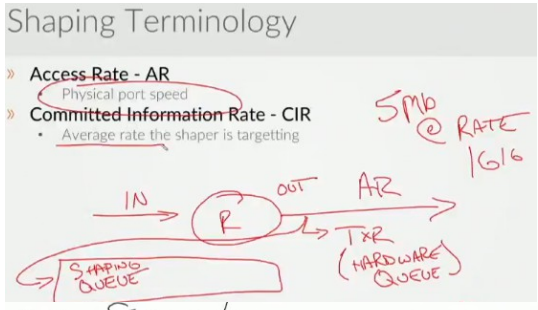
Shaping:

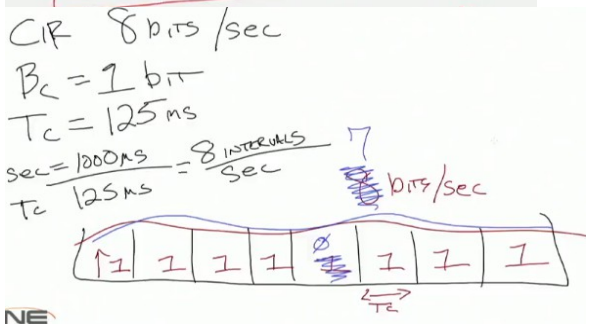
Traffic Shaping Overview

- Goal is to normalize traffic flow
  - Smooth out traffic bursts
  - Prepares traffic for ingress policing
  - Delay and Queue exceeding traffic

Shaping Terminology

- Access Rate - AR
  - Physical port speed
- Committed Information Rate - CIR
  - Average rate the shaper is targetting
- Time Committed - Tc
  - Time interval in ms to emit traffic bursts
  - Bursts always emitted at Access Rate (AR)
- Burst Committed - Bc
  - Amount of bits that could be sent every Tc
- Burst Excessive - Be
  - Amount of bits over Bc that could be sent during Tc
  - Must be accumulated by idle periods





Traffic Policing Parameters

- The larger the Tc the more bursting is allowed
  - Bc = CIR*Tc is maximum burst size allowed momentarily (in bytes)
- Be – excessive burst
  - Max amount of bytes allowed above Bc during Tc
  - Only allowed if Bc was not fully utilized before

MQC Generic Traffic Shaping

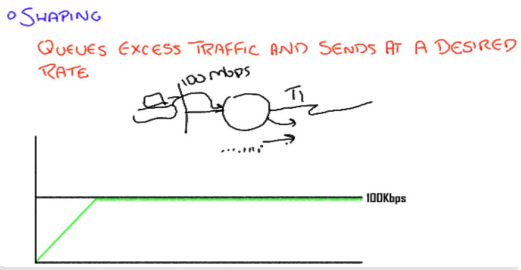
- Configured using MQC syntax
  - shape average <CIR> [Bc] [Be]
  - Tc is found implicitly as Bc/CIR
- Default shaper queue is FIFO
  - Can be turned into HQF by associating a child policy-map with shaped class
  - Specify HQF settings in the child policy-map
  - I.e. nested policies

```
R2(config-if)#band
R2(config-if)#bandwidth ?
  <1-10000000>  Bandwidth in kilobits
  inherit       Specify how bandwidth is inherited
  receive       Specify receive-side bandwidth

R2(config-if)#bandwidth 750000
R2(config-if)#do show policy-map int gig1
 GigabitEthernet1

  Service-policy output: SHAPE

    Class-map: class-default (match-any)
      12 packets, 1211 bytes
      5 minute offered rate 0000 bps, drop rate 0000 bps
      Match: any
      Queueing
      queue limit 104 packets
      (queue depth/total drops/no-buffer drops) 0/0/0
      (pkts output/bytes output) 12/1211
      shape (average) cir 375000000, bc 1500000, be 1500000
      target shape rate 375000000
R2(config-if)#
```



Shaping Formulas

- Single-Rate Shaper (sub-rate)
  - AIR = interface (port) speed
  - CIR = Bc/Tc = average speed (shaping rate)
  - EIR = (AIR-CIR) = excessive rate (sporadic)
  - Be = EIR*Tc = excessive burst
    - May be prohibited by setting Be=0

Policing:

Traffic Policing

- Used to meter a packet flow rate
  - Marks packets that exceed metered rate
  - Drop is a mark action
- Normally an ingress operation
  - E.g. PE ingress from CE
- Policing has two parameters
  - Metering rate – CIR
  - Averaging interval – Tc

Traffic Policing Parameters

- The larger the Tc the more bursting is allowed
  - Bc = CIR*Tc is maximum burst size allowed momentarily (in bytes)
- Be – excessive burst
  - Max amount of bytes allowed above Bc during Tc
  - Only allowed if Bc was not fully utilized before

Single-Rate Policing Syntax

- Configuration
  - police [cir] [<CIR>] [<Bc>] [<Be>]
  - CIR in bps while bursts are in bytes
- Applied to an MQC class
  - Three actions (colors): conform, exceed, violate
  - Exceed: flow exceeds Bc but under Bc+Be
  - Violate: burst size exceeds Bc+Be

Dual-Rate Policing Syntax

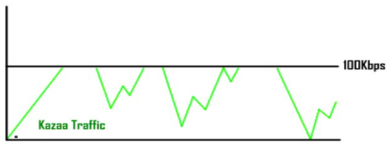
- Configuration
  - police cir [<CIR>] bc [<Bc>] pir [<PIR>] be [<Be>]
- Normally used to implement two-rate access
  - Customer is guaranteed CIR
  - Allowed to send up to PIR
  - Traffic between PIR and CIR remarked
    - E.g. lower DSCP

Shaping and Policing Together

- Operations are Complimentary
  - Shaping is done egress
  - Policing is done ingress
- Parameters should match
  - Shaping is set to match policing
  - Same CIR, Bc and Be
    - Policing values could be greater actually

- Policing

  Drops or (re) marks exceeding traffic



Policing:

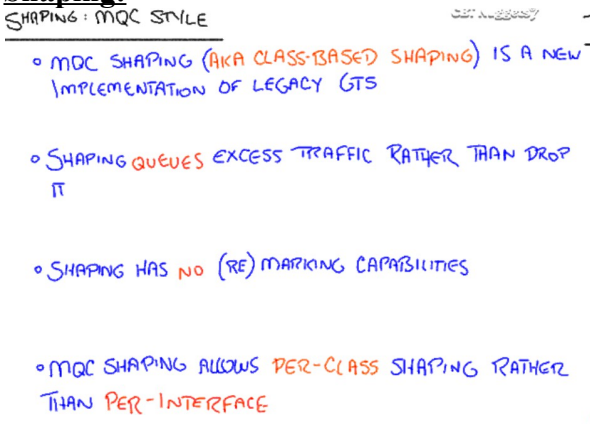
R3(config)#class-map P2P
R3(config-cmap)#match protocol edonkey
R3(config)#policy-map P2P_POLICE
R3(config-pmap)#class P2P
R3(config-pmap-c)#police 56000 confirm-action transmit exceed-action drop !(cir 56000 in bps)
R3(config)#int s0/0
R3(config-if)#service-policy input P2P !(Policing can be applied as an input instead of output)

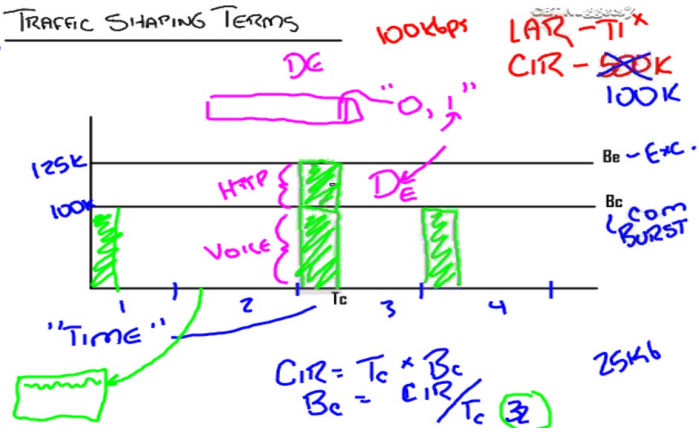
3 Token Bucket model:

conform-action: Bc (Committed Burst rate)
exceed-action: Be (Excess Burst) (Bandwidth that was below Bc | Banked bandwidth)
violate-action: Drop

R3(config-pmap-c)#police 150000 conform-action transmit exceed-action set-prec-transmit 0 violate-action drop !(CIR (Committed Information Rate) is 150000 bps Bc 4687 Be 4687)
!(Bc and Be is calculated by dividing CIR by 32)

Shaping:

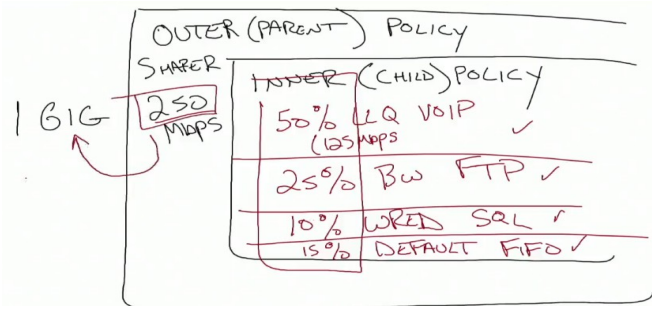




LAR (Local Access Rate) e.g. E1 speed
CIR (Committed Information Rate) lower than LAR
Tc (Time Committed) e.g. ¼ (quarter) of a second.
e.g. If CIR 100 kbps then Bc is 25 Kbps
Bc (Committed Burst) (Average) (ISP has committed to give us)
Be (Excess Burst)(Peak)(Banked bandwidth) (Decided in a contract e.g. 25 Kbps)
CIR=Bc / Tc
Bc = CIR x Tc
(Cisco by default uses 4th of a second for Tc)
De (Discard Eligible) (ISP mark the marks the packets with DE 1 and could be dropped)(We can decide which traffic we want to mark with DE bit to avoid getting imp traffic dropped as Be)
R1(config-pmap-c)#shape peak 500000 !(CIR is 500000 Kbps Cisco calculates the Bc and Be)
R1(config-pmap-c)#shape average 500000 !(CIR is 500000 Kbps Cisco calculates the Bc and Be)
R1(config-pmap-c)#shape average percent 50

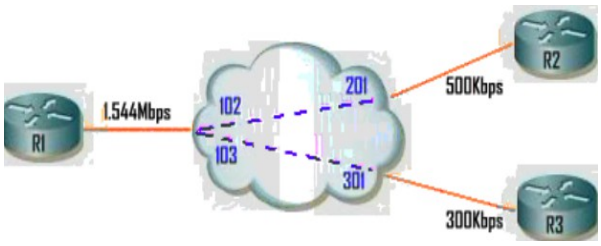
Nested Classes/HQF or HQoS (Hierarchical Queuing Framework or Hirarchical QoS):

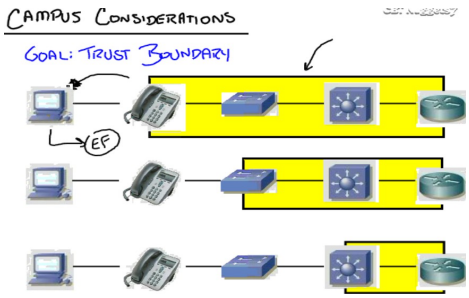
```
Router(config)# class-map match-any class2
Router(config-cmap)# match protocol ip
Router(config-cmap)# match qos-group 3
Router(config-cmap)# match access-group 2
Router(config-cmap)# exit
Router(config)# class-map match-all class1
Router(config-cmap)# match class-map class2
Router(config-cmap)# match destination-address mac 00.00.00.00.00.00
Router(config-cmap)# exit
```




```
R1(config)#policy-map PRIORITY
R1(config-pmap)#class DATA
R1(config-pmap-c)#bandwidth 50
R1(config-pmap)#class VOICE
R1(config-pmap-c)#priority 300
R1(config-pmap)#class class-default
R1(config-pmap-c)#fair-queue
```

```
R1(config)#class-map ALL_TRAFFIC
R1(config-cmap)#match class DATA
R1(config-cmap)#match class VOICE
R1(config-cmap)#match class class-default
R1(config)#policy-map SHAPE_500
R1(config-pmap)#class ALL_TRAFFIC
R1(config-pmap-c)#shape average 500000
R1(config-pmap-c)#service-policy PRIORITY
R1(config)#int s0/0.1
R1(config-subif)#service-policy output SHAPE_500
```





WAN CONSIDERATIONS

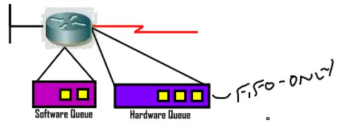
- WAN USES SOFTWARE QoS
  - MARK TRAFFIC EARLY
  - OLDER DEVICES MAY BE SWAMPED
- THE PERCENTAGE GAME
  - 25% OF BANDWIDTH LEFT FOR BEST-EFFO
  - LLQ'S 33% RULE
    - × OVER-RESERVATION AFFECTS DATA
    - × OVER-RESERVATION AFFECTS VOICE

WAN CONSIDERATIONS

- SERIALIZATION DELAY
  - GOAL IS 10-15MS DELAY PER-HOP
  - NECESSARY ON LINKS ≤ 768Kbps
- cRTP
  - MOST CPU-INTENSE QoS STRATEGY
  - USE ONLY ON LINKS ≤ 768Kbps
- PAK_PRIORITY
  - ROUTER AUTOMATICALLY TAGS ALL "MISSION-CRITICAL" TRAFFIC WITH IPPS/CS6
  - PAK_PRIORITY IS A UNCHANGABLE, INTERNAL MECHANISM

WAN CONSIDERATIONS

- Tx-RING TUNING
  - THE HARDWARE QUEUE OF A ROUTER



  - IS TUNED AUTOMATICALLY IN NEWER IOS VERSIONS
  - TYPICALLY HOLDS 32-64 PACKETS BY DEFAULT
  - REDUCE TO 3 ON SLOW (≤ T1/E1) LINKS

WAN SPEEDS

- HIGH SPEED LINKS (>T1/E1)
  - NO LFI
  - NO cRTP
  - 5 TO 11 TRAFFIC CLASSES RECOMMENDED

J SPEEDS

- 768Kbps OR LESS
  - VIDEO NOT RECOMMENDED
  - LFI FOR VoIP
  - CAREFULLY USE cRTP
  - REDUCE Tx-RING TO 3
  - 3 TO 5 TRAFFIC CLASSES RECOMMENDED

WAN SPEEDS

- LINKS BETWEEN 768Kbps AND T1/E1
  - VoIP & VIDEO GET LLQ PRIORITY
  - NO LFI
  - OPTIONAL cRTP
  - REDUCE Tx-RING TO 3
  - 3 TO 5 TRAFFIC CLASSES RECOMMENDED

WHAT IS AUTOQoS?

- A CISCO-DEVELOPED QoS TEMPLATE
- GOOD FOR:
  - FAST QoS DEPLOYMENT
  - LARGE SCALE CONSISTANCY
  - SERVICE PROVIDERS DEPLOYING MANAGED VoIP

REQUIREMENTS TO USE AUTOQoS

- MAINLINE ROUTER/SWITCH WITH RECENT IOS
- CEF MUST BE ENABLED
- BANDWIDTH MUST BE CONFIGURED ON THE INTERFACE
- IP ADDRESS MUST BE ASSIGNED
- INTERFACE MUST BE NON-ADMINISTRATIVELY DOWN

Example (Shaping):

```
class-map match-all HTTP
        match protocol http
class-map match-all SQL
        match protocol sqlserver
class-map match-all BITTORRENT
        match protocol bittorrent
class-map match-all VOIP
        match protocol rtp

policy-map INNER_POLICY
        class HTTP
                bandwidth percent 50    !(500Mbps for a gig link)
        class SQL
```

```
            bandwidth percent 30    !(300Mbps for a gig link)
        class BITORRENT
            bandwidth percent 10    !(100Mbps for a gig link)
        class VOIP
            priority percent 5    !(50Mbps for a gig link)
        class class-default
            bandwidth percent 1    !(this is there by default)(1Mbps for a gig link)

int gig1
service-policy output INNER_POLICY

!(bandwidth percent is based on software defined bandwidth value and class reservations will
change if we change the interface bandwidth value)
int gig1
bandwidth 500000    !(in kbps)
--------------
policy-map SHAPE_100Mbps
        class class-default
            shape average 100 m   !(100Mbps)
            service-policy INNER_POLICY

policy-map SHAPE_200Mbps
        class class-default
            shape average 200 m   !(200Mbps)
            service-policy INNER_POLICY
---------------
```

AutoQoS:

```
R(config)#int s0/0
R(config-if)#bandwidth 500
R(config-if)#auto qos voip
```

Verification and Troubleshooting:

```
R1(config)#ip telnet tos B8   !(to test dscp field on wireshark captured traffic)
R1#ping    !(extended ping to test dscp field on wireshark captured traffic)

R3(config)#access-list 101 permit ip any any dcsp cs2
R3(config)#access-list 101 permit ip any any
R3(config)#int fa0/0
R3(config-if)#ip access-group 101 in
R3#sh access-list 101       !(to verify if the dcsp is happening on the receiving router)

sh class-map
sh policy-map
sh policy-map int fa0/1   !(important command to see it all)
sh ip nbar protocol-discovery stats bit-rate top-n 5  !(different types of traffic going through the
router. This uses NBAR)
sh ip nbar
sh ip rtp header-compression
sh queue fa0/0
sh mls qos
```