Communications Services and Security
Quality of Service

Cèsar Fernández

Departament d'Informàtica
Universitat de Lleida

Curs 2019 - 2020

1. **QoS Overview**
   - What is QoS ?
   - QoS architecture
   - QoS service models
2. **Classification**
   - Classification overview
   - Policy-based routing
   - Policy propagation via BGP
   - Committed Access Rate (CAR)
3. **Congestion Management**
   - Congestion management overview
   - Flow-based WFQ
   - Class-based WFQ (CBWFQ)
   - Custom queueing (CQ)
   - Priority queueing (PQ)
   - Low Latency Queueing (LLQ)
4. **Congestion Avoidance**
   - Congestion avoidance overview
   - WRED
5. **Policing and shaping**
   - Policing and shaping overview
   - Token bucket
   - Traffic policing
   - Traffic shaping
6. **Resource Reservation Protocol**
   - Overview
   - The protocol
   - Features
   - Configuration and monitoring
7. **Bibliography**

Contents

1. **QoS Overview**
   - What is QoS ?
   - QoS architecture
   - QoS service models
2. **Classification**
3. **Congestion Management**
4. **Congestion Avoidance**
5. **Policing and shaping**
6. **Resource Reservation Protocol**
7. **Bibliography**

What is QoS ?

Ability of a network to improve service to specific network traffic, providing the following services:

- Dedicated bandwidth
- Improving packet losses
- Avoiding and managing congestion
- Shaping traffic
- Setting priorities across the network

QoS architecture

Three essential components:

- QoS in a single network: queuing, scheduling and shaping
- QoS across networks: signaling
- QoS policy and management

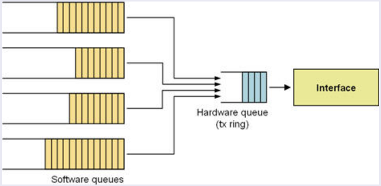
Types of routers:

- Edge routers: packet classification, admission control
- Backbone routers: congestion management and avoidance

QoS architecture

Queuing

- Soft queues only formed when incoming traffic is faster than outcoming rates
- By default (if not QoS defined), slow output i/fs (few Mbps) use *Weighted Fair Queuing*. Otherwise: FIFO applied
- Queue length may be configured
- When queues are full, traffic is dropped



Source: http://wiki.nil.com/Queuing_Principles_in_Cisco_IOS

QoS architecture

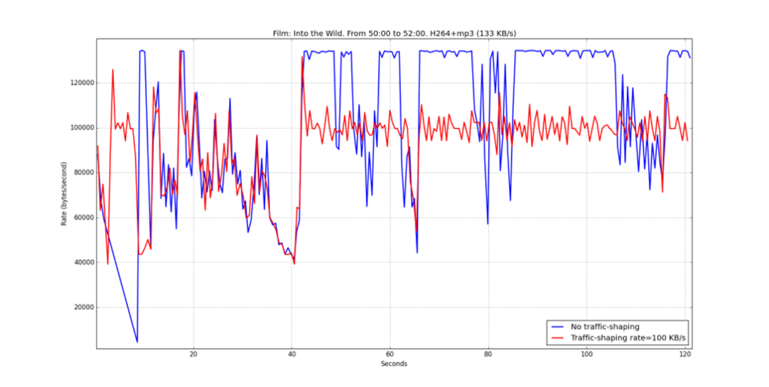
Scheduling

- How the soft queues are served:
  - WFQ (Flow based, class based)
  - Custom queuing: assigns a given bandwidth
  - Priority queuing: Assigns priority. Higher priorities are served first

QoS architecture

Shaping

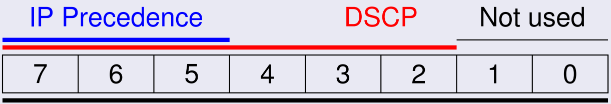
- Average rate and maximum burst size are enforced on outgoing traffic
- Token bucket mechanisms



QoS architecture

Signaling

- Field TOS (Type of Service) of IPv4 header marked to indicate priority
- 3 MSB determine **IP precedence**. 8 priority levels
- 6 MSB determine DSCP (Differentiated Services Code Point, **DiffServ**). New standard.



IP Precedence DSCP Not used

7	6	5	4	3	2	1	0

TOS field

IP Precedence values

111	Network Control	011	Flash
110	Internet Control	010	Immediate
101	Critical	001	Priority
100	Flash override	000	Routine

QoS service models

Service models (or levels of service) describe the end-to-end QoS capabilities. 3 models:

- Best effort
- Integrated services
- Differentiated services

Best effort

Network delivers data if it can, without any assurance of reliability, delay bounds, or throughput.
FIFO queuing. Suitable for most applications (email, file transfer, . . .)

QoS service models

Integrated services

- Application requests a specific service before sending data
- Requests made by signaling (e.g. RSVP (Reservation Protocol), asking for bandwidth and delay requirements)
- If possible, networks employs smart queuing mechanisms to provide service; WFQ or WRED (Weighted RED)

Differentiated services

- Not explicitly requested service
- Using IP Precedence or DCSP signaling

Contents

1 QoS Overview

2 Classification
- Classification overview
- Policy-based routing
- Policy propagation via BGP
- Committed Access Rate (CAR)

3 Congestion Management

4 Congestion Avoidance

5 Policing and shaping

6 Resource Reservation Protocol

7 Bibliography

Classification overview

To provide a preferential service to a type of traffic, it must be classified. Classification is done in 2 steps:

1. Traffic must be identified. Identification methods:
    - Use of ACLs (Access Control Lists)
    - Definition of **route maps**
2. Optionally may be marked.
    - If identified and not marked, classification is said to be on a per-hop basis. Not passed to the next router
    - When marked for network-wide use, IP Precedence bits are set

When marked, routers can use IP Precedence bits to:

- determine how WFQ and WRED methods manages the traffic
- use features such as **policy-based routing** or **committed access rate** (CAR)

Policy-based routing

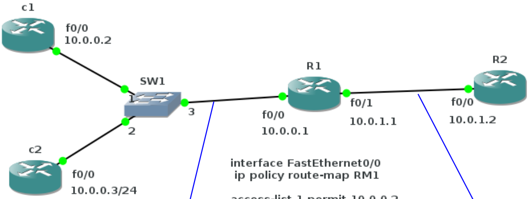
PBR. How it works ?

Traffic flows can be configured, marked and routed accordingly.

- Incoming traffic is classified using ACLs or extended ACLs. (Based on IPs, port numbers, packet length, . . .)
- IP Precedence bits are set according to classification
- Specific next-hop routers may be set

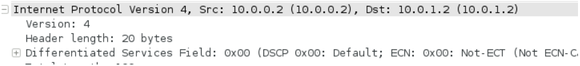
Policy-based routing

Example:

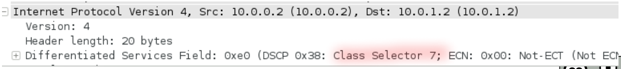


```
interface FastEthernet0/0
 ip policy route-map RM1

access-list 1 permit 10.0.0.2
access-list 2 permit 10.0.0.3
!
route-map RM1 permit 10
 match ip address 1
 set ip precedence network  (IP precedence 7)
!
route-map RM1 permit 20
 match ip address 2
 set ip precedence priority  (IP precedence 1)
```

Ping from 10.0.0.2 to 10.0.1.2



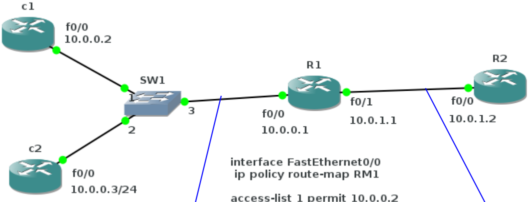
```
⊟ Internet Protocol Version 4, Src: 10.0.0.2 (10.0.0.2), Dst: 10.0.1.2 (10.0.1.2)
    Version: 4
    Header length: 20 bytes
  ⊞ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-C
```



```
⊟ Internet Protocol Version 4, Src: 10.0.0.2 (10.0.0.2), Dst: 10.0.1.2 (10.0.1.2)
    Version: 4
    Header length: 20 bytes
  ⊞ Differentiated Services Field: 0xe0 (DSCP 0x38: Class Selector 7; ECN: 0x00: Not-ECT (Not ECN
```

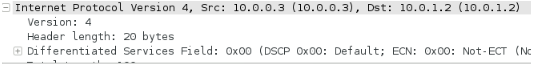
Policy-based routing

Example:

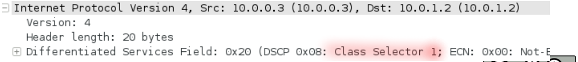


```
interface FastEthernet0/0
 ip policy route-map RM1

access-list 1 permit 10.0.0.2
access-list 2 permit 10.0.0.3
!
route-map RM1 permit 10
 match ip address 1
 set ip precedence network  (IP precedence 7)
!
route-map RM1 permit 20
 match ip address 2
 set ip precedence priority  (IP precedence 1)
```

Ping from 10.0.0.3 to 10.0.1.2



```
⊟ Internet Protocol Version 4, Src: 10.0.0.3 (10.0.0.3), Dst: 10.0.1.2 (10.0.1.2)
    Version: 4
    Header length: 20 bytes
  ⊞ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Nc
```



```
⊟ Internet Protocol Version 4, Src: 10.0.0.3 (10.0.0.3), Dst: 10.0.1.2 (10.0.1.2)
    Version: 4
    Header length: 20 bytes
  ⊞ Differentiated Services Field: 0x20 (DSCP 0x08: Class Selector 1; ECN: 0x00: Not-E
```

Policy propagation via BGP

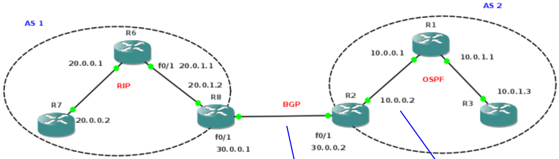
Configuration

Allows packet classification marking IP precedence based on BGP community

1. Indicate to the incoming i/f that bgp-policy IP precedence classification must be used
2. Define access list matching the required path
3. Define a route-map setting the IP precedence
4. Use the route-map defined in the BGP router instance

Policy propagation via BGP

Example:

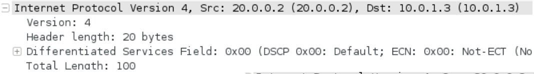


```
! R2

show ip route 20.0.0.0 255.255.255.0
Routing entry for 20.0.0.0/24
  Known via "bgp 2", distance 20, metric 1
  Tag 1, precedence flash (3), type external
  Last update from 30.0.0.1 00:00:18 ago
  Routing Descriptor Blocks:
  * 30.0.0.1, from 30.0.0.1, 00:00:18 ago
      Route metric is 1, traffic share count is 1
      AS Hops 1
      Route tag 1
```

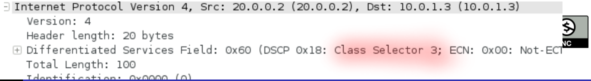
CARE: Routers must be rebooted to show precedence

```
!R2

interface FastEthernet0/1
 ip address 30.0.0.2 255.255.255.0
 bgp-policy source ip-prec-map

router bgp 2
 no synchronization
 table-map MARK-PRECEDENCE
 bgp log-neighbor-changes
 network 10.0.0.0 mask 255.255.254.0
 neighbor 30.0.0.1 remote-as 1

ip as-path access-list 1 permit ^(1_)+$

route-map MARK-PRECEDENCE permit 10
 match as-path 1
 set ip precedence flash
```

Ping from 20.0.0.2 to 10.0.1.3



```
⊟ Internet Protocol Version 4, Src: 20.0.0.2 (20.0.0.2), Dst: 10.0.1.3 (10.0.1.3)
    Version: 4
    Header length: 20 bytes
  ⊞ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (No
    Total Lenath: 100
```



```
⊟ Internet Protocol Version 4, Src: 20.0.0.2 (20.0.0.2), Dst: 10.0.1.3 (10.0.1.3)
    Version: 4
    Header length: 20 bytes
  ⊞ Differentiated Services Field: 0x60 (DSCP 0x18: Class Selector 3; ECN: 0x00: Not-ECT
    Total Length: 100
    Identification: 0x0000 (0)
```

Committed Access Rate (CAR)

CAR

CAR is a feature that implements classification and policing. Limits the input or output rate at an i/f.
Rate policies can be applied according to:

- All IP traffic
- IP precedence
- MAC address
- IP access list

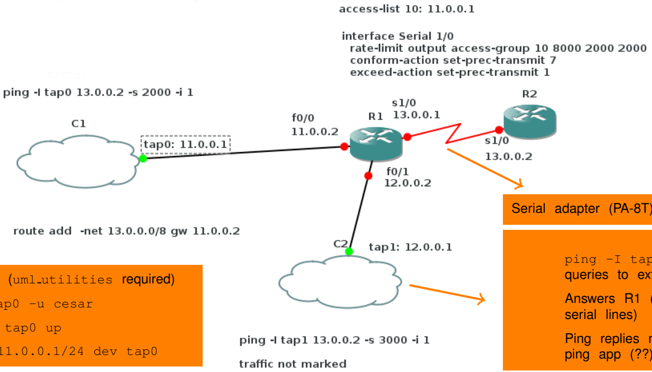
Committed Access Rate (CAR)

CAR configuration

Configuration is done in a interface:

- Set **rate-limit** for input or output traffic giving:
  - Average rate (in bps)
  - Normal burst size (in bytes)
  - Maximum burst size. Bursts between normal and maximum are considered exceeding with increasing probability
- Set the actions to be performed for conforming (**conform-action**) and exceeding (**exceed-action**) traffic. Actions can be:
  - Drop the packet
  - Transmit
  - Set precedence and transmit
  - Continue (evaluate the next rate-limit action)
  - Set the precedence and continue

Committed Access Rate (CAR)



```
access-list 10: 11.0.0.1

interface Serial 1/0
  rate-limit output access-group 10 8000 2000 2000
  conform-action set-prec-transmit 7
  exceed-action set-prec-transmit 1
```

```
ping -I tap0 13.0.0.2 -s 2000 -i 1
```

```
route add  -net 13.0.0.0/8 gw 11.0.0.2
```

```
ping -I tap1 13.0.0.2 -s 3000 -i 1
```

traffic not marked

Serial adapter (PA-8T) at 1.4 Mbps

`ping -I tap1` sends ARP queries to external IP (13.0.0.2)

Answers R1 (proxy ARP on serial lines)

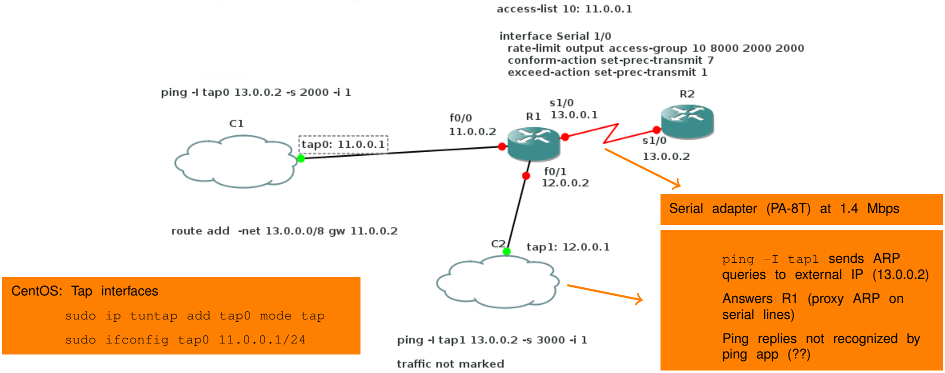
Ping replies not recognized by ping app (??)

Fedora: Tap interfaces (`uml_utilities` required)

```
tunctl -t tap0 -u cesar
ip link set tap0 up
ip add add 11.0.0.1/24 dev tap0
```

- Traffic from `tap0`. 2000 bytes every second. Rate 16 Kbps > 8 Kbps.
- Approx. half of the packets will be set to IP prec 1 outcoming `s1/0`
- No packets marked from `tap1`
- Check capture at serial line `s1/0`

Committed Access Rate (CAR)



```
access-list 10: 11.0.0.1

interface Serial 1/0
  rate-limit output access-group 10 8000 2000 2000
  conform-action set-prec-transmit 7
  exceed-action set-prec-transmit 1
```

```
ping -I tap0 13.0.0.2 -s 2000 -i 1
```

```
route add  -net 13.0.0.0/8 gw 11.0.0.2
```

```
ping -I tap1 13.0.0.2 -s 3000 -i 1
```

traffic not marked

Serial adapter (PA-8T) at 1.4 Mbps

`ping -I tap1` sends ARP queries to external IP (13.0.0.2)

Answers R1 (proxy ARP on serial lines)

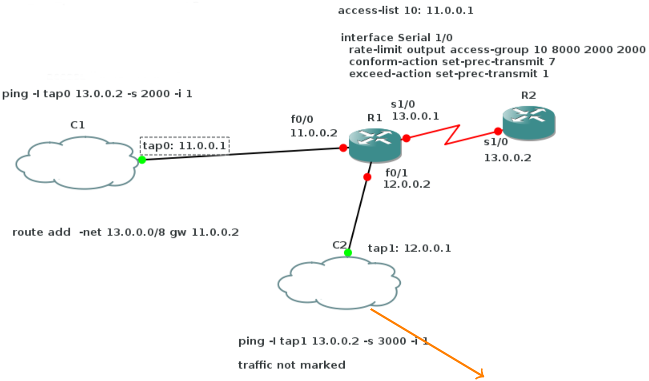
Ping replies not recognized by ping app (??)

CentOS: Tap interfaces

```
sudo ip tuntap add tap0 mode tap
sudo ifconfig tap0 11.0.0.1/24
```

- Traffic from `tap0`. 2000 bytes every second. Rate 16 Kbps $>$ 8 Kbps.
- Approx. half of the packets will be set to IP prec 1 outcoming `s1/0`
- No packets marked from `tap1`
- Check capture at serial line `s1/0`

Committed Access Rate (CAR)



```
access-list 10: 11.0.0.1

interface Serial 1/0
  rate-limit output access-group 10 8000 2000 2000
  conform-action set-prec-transmit 7
  exceed-action set-prec-transmit 1
```

```
ping -I tap0 13.0.0.2 -s 2000 -i 1
```

```
route add  -net 13.0.0.0/8 gw 11.0.0.2
```

```
ping -I tap1 13.0.0.2 -s 3000 -i 1
```

traffic not marked

To show ping responses on tap1:

```
echo "1 rt2" >> /etc/iproute2/rt_tables
ip route add 13.0.0.0/24 via 12.0.0.2 table rt2
ip rule add from 12.0.0.1/32 table rt2
```

Contents

1. QoS Overview
2. Classification
3. **Congestion Management**
   - Congestion management overview
   - Flow-based WFQ
   - Class-based WFQ (CBWFQ)
   - Custom queueing (CQ)
   - Priority queueing (PQ)
   - Low Latency Queueing (LLQ)
4. Congestion Avoidance
5. Policing and shaping
6. Resource Reservation Protocol

Congestion management overview

Congestion management tasks

1. Creation of software queues
2. Assign packets to queues based on classification
3. Schedule packets in queues for transmission

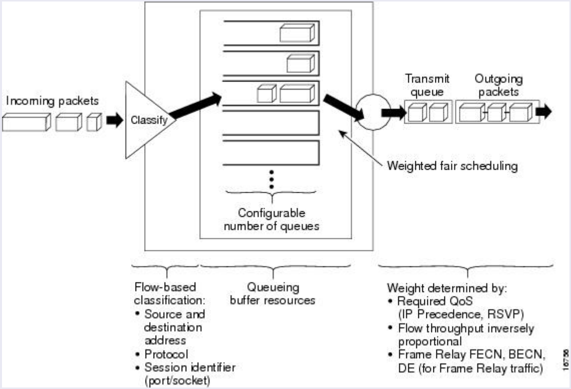
Congestion management overview

Types of queues

- FIFO. No QoS
- WFQ. Default for slow speed i/fs. 2 types:
  - Flow-based WFQ. A flow is determined by IPs, protocol and port numbers of a connection. Configurable number of queues. 256 as default. No configuration required
  - Class-based WFQ (CBWFQ). Definition of **class-maps** based on access-lists. 1 queue per class. Up to 64 classes
- Custom Queueing (CQ). Allocates bandwidth for each class of traffic. 16 queues. Round robin scheduling (Weighted round robin, WRR)
- Priority Queueing (PQ). Packets from a priority are sent before all lower priorities. Ensures low latency requirements. 4 queues
- Low Latency Queueing (LLC). Adds PQ to flow-based WFQ or CBWFQ

Flow-based WFQ

Schema


Incoming packets
Classify
Transmit queue
Outgoing packets
Weighted fair scheduling
Configurable number of queues
Flow-based classification:
• Source and destination address
• Protocol
• Session identifier (port/socket)
Queueing buffer resources
Weight determined by:
• Required QoS (IP Precedence, RSVP)
• Flow throughput inversely proportional
• Frame Relay FECN, BECN, DE (for Frame Relay traffic)
16758

Source: Cisco IOS Quality of Service Solutions Configuration Guide, Release 12.2

Flow-based WFQ

WFQ and IP precedence

Having $Nflows_j$ flows with a IP precedence value j and an assigned weight w_j, the assigned bandwidth $(1/r_i)$ to a flow of precedence i is computed as:

$$\frac{1}{r_i} = \frac{w_i}{\sum_{j=0}^{7} Nflows_j \cdot w_j}$$

As flows are added and ended, the allocated bandwidth changes continuously

Example (taking $w_i = i + 1$)

Having 5 flows; 2 with IP precedence value 0 (routine), and 3 with IP precedence 5 (critical), their assigned bandwidth results:

$$\frac{1}{r_0} = \frac{1}{2 \cdot 1 + 3 \cdot 6} = 1/20 = 0.05$$

$$\frac{1}{r_5} = \frac{6}{2 \cdot 1 + 3 \cdot 6} = 6/20 = 0.3$$

Flow-based WFQ

Configuring WFQ

- WFQ is configured as default control management for slow speed links (<2 Mbps)
- Command **fair-queue** run on i/f basis. 3 parameters:
  1. *congestive-discard-threshold*. Number of packets allowed in each queue. Default 64
  2. *dynamic-queues*. Number of WFQ queues. Power of 2. Default depends on i/f BW. 256 for links $>$ 512 Kbps
  3. *reservable-queues*. Reserved to RSVP (Integrated Services) or CBWFQ (DiffServ), ... Default 0

Flow-based WFQ

Monitoring WFQ

```
R1#show queue Serial 1/0
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops:
  Queueing strategy: weighted fair
  Output queue: 0/1000/64/0 (size/max total/threshold/drops)
     Conversations  0/1/256 (active/max active/max total)
     Reserved Conversations 0/0 (allocated/max allocated)
     Available Bandwidth 1158 kilobits/sec

R1#show queue Serial 1/0
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops:
  Queueing strategy: weighted fair
  Output queue: 63/1000/64/668800 (size/max total/threshold/drops)
     Conversations  1/2/256 (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) 63/4048/66880
  Conversation 29, linktype: ip, length: 332
  source: 12.0.0.1, destination: 13.0.0.2, id: 0x3FFA, ttl: 63, prot:
```

Flow-based WFQ

WFQ weights

Predefined weights (w_i) are the following:

IP prec.	Name	WFQ weight ($1/w_i$)
111	Network Control	4,048
110	Internet Control	4,626
101	Critical	5,397
100	Flash override	6,476
011	Flash	8,096
010	Immediate	10,794
001	Priority	16,192
000	Routine	32,384

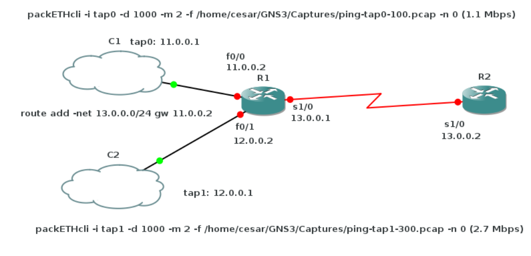
Computed as

$$\frac{1}{w_i} = \frac{32,384}{\mathrm{IP\_Prec}_i + 1}$$

Weights only can be configured as DWFQ (Distributed WFQ) that runs on advanced processors

Flow-based WFQ

Example



```
packETHcli -i tap0 -d 1000 -m 2 -f /home/cesar/GNS3/Captures/ping-tap0-100.pcap -n 0 (1.1 Mbps)
```

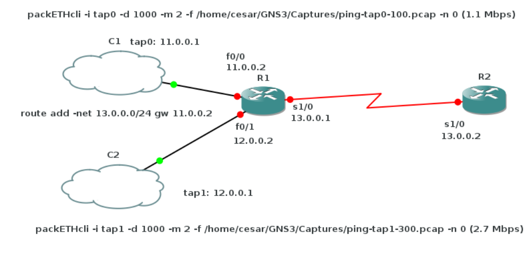
```
route add -net 13.0.0.0/24 gw 11.0.0.2
```

```
packETHcli -i tap1 -d 1000 -m 2 -f /home/cesar/GNS3/Captures/ping-tap1-300.pcap -n 0 (2.7 Mbps)
```

- No further configuration required on i/f `s1/0`
- Packets at input i/fs are IP precedence marked after classification; `f0/1 network`(7), `f0/0 routine`(0)
- `ping` from `tap0`. Data length 92 bytes. ICMP header (8 bytes). IP header (20 bytes). Ethernet header (14 bytes). Total packet length: 134 bytes. **Data rate**: $134 \cdot 8/10^{-3} = 1.07$ Mbps
- `ping` from `tap1`. Data length 292 bytes. Total packet length: 334 bytes. Data rate: 2.67 Mbps
- Find packETH here and packETHcli here

Flow-based WFQ

Example



```
packETHcli -i tap0 -d 1000 -m 2 -f /home/cesar/GNS3/Captures/ping-tap0-100.pcap -n 0 (1.1 Mbps)
```

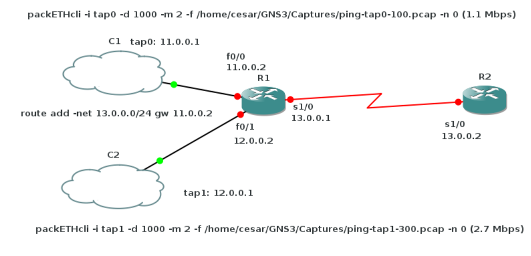
```
route add -net 13.0.0.0/24 gw 11.0.0.2
```

```
packETHcli -i tap1 -d 1000 -m 2 -f /home/cesar/GNS3/Captures/ping-tap1-300.pcap -n 0 (2.7 Mbps)
```

- Assigned bandwidths:
  - From `tap0`: $1/r_0 = \frac{1/32{,}384}{1/4{,}048+1/32{,}384} \simeq 1/9$
  - From `tap1`: $1/r_7 = \frac{1/4{,}048}{1/4{,}048+1/32{,}384} \simeq 8/9$
- Capturing at `s1/0`, taking 793 packets, we observe:
  - 189 packets from `tap0`. 128 bytes each (IP packet). 24,192 bytes
  - 604 packets from `tap1`. 198,112 bytes
  - Gives a ratio of 8.18 = 198,112 / 24,192

Flow-based WFQ

Example



```
packETHcli -i tap0 -d 1000 -m 2 -f /home/cesar/GNS3/Captures/ping-tap0-100.pcap -n 0 (1.1 Mbps)
```

```
route add -net 13.0.0.0/24 gw 11.0.0.2
```

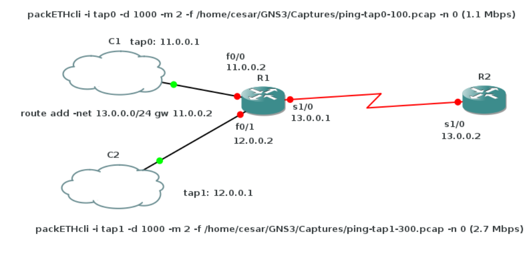
```
packETHcli -i tap1 -d 1000 -m 2 -f /home/cesar/GNS3/Captures/ping-tap1-300.pcap -n 0 (2.7 Mbps)
```

R1 configuration

```
interface FastEthernet0/0
 ip address 11.0.0.2 255.255.255.0
 ip policy route-map RM0
!
interface FastEthernet0/1
 ip address 12.0.0.2 255.255.255.0
 ip policy route-map RM1
```

Flow-based WFQ

Example



```
packETHcli -i tap0 -d 1000 -m 2 -f /home/cesar/GNS3/Captures/ping-tap0-100.pcap -n 0 (1.1 Mbps)
```

```
route add -net 13.0.0.0/24 gw 11.0.0.2
```

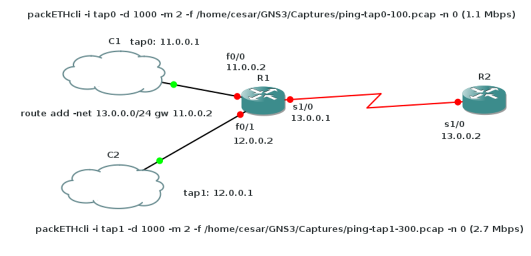
```
packETHcli -i tap1 -d 1000 -m 2 -f /home/cesar/GNS3/Captures/ping-tap1-300.pcap -n 0 (2.7 Mbps)
```

R1 configuration

```
access-list 1 permit any
!
route-map RM1 permit 10
 match ip address 1
 set ip precedence network
!
route-map RM0 permit 10
 match ip address 1
 set ip precedence routine
```

Flow-based WFQ

Example



```
packETHcli -i tap0 -d 1000 -m 2 -f /home/cesar/GNS3/Captures/ping-tap0-100.pcap -n 0 (1.1 Mbps)
```

```
route add -net 13.0.0.0/24 gw 11.0.0.2
```

```
packETHcli -i tap1 -d 1000 -m 2 -f /home/cesar/GNS3/Captures/ping-tap1-300.pcap -n 0 (2.7 Mbps)
```

Monitoring queues

```
R1#show queueing interface Serial 1/0
Interface Serial1/0 queueing strategy: fair
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 46388
  Queueing strategy: weighted fair
  Output queue: 64/1000/64/46388 (size/max total/threshold/drops)
     Conversations  2/3/256 (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) 48/4048/22662/0/0
  Conversation 29, linktype: ip, length: 332
  source: 12.0.0.1, destination: 13.0.0.2, id: 0x3FFA, ttl: 63, prot: 1

  (depth/weight/total drops/no-buffer drops/interleaves) 16/32384/23727/0/0
  Conversation 28, linktype: ip, length: 132
  source: 11.0.0.1, destination: 13.0.0.2, id: 0x3F9E, ttl: 63, prot: 1
```

Class-based WFQ (CBWFQ)

Characteristics

- Classes defined according to matching criteria, access-lists and input i/fs
- A single queue (from the WFQ) is reserved for each class
- Parameters to assign at each class-queue:
  - **Bandwidth**. In bps or a % of the total. A **max-reserved-bandwidth** is set as default to 75%
  - **Weight** for WFQ is automatically derived from the assigned bandwidth
  - **Queue limit**
- Packet drop. Once the queue limit is reached, **tail-drop** applies. WRED can be also configured
- Up to 64 class-queues per i/f

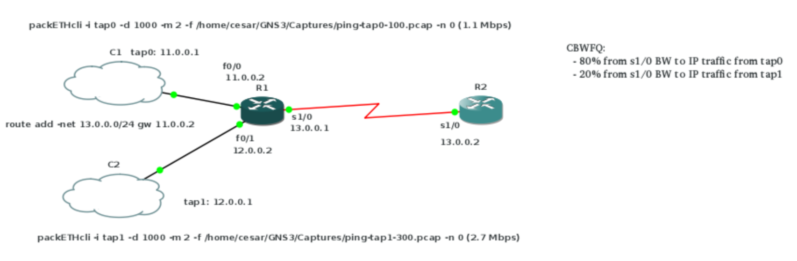
Class-based WFQ (CBWFQ)

Configuration steps

- Define **class-maps** specifying how traffic is classified
- Define **policy-maps** indicating what to do with defined classes
- Apply policies to i/fs

Class-based WFQ (CBWFQ)

Example



```
packETHcli -i tap0 -d 1000 -m 2 -f /home/cesar/GNS3/Captures/ping-tap0-100.pcap -n 0 (1.1 Mbps)
```

```
route add -net 13.0.0.0/24 gw 11.0.0.2
```

```
packETHcli -i tap1 -d 1000 -m 2 -f /home/cesar/GNS3/Captures/ping-tap1-300.pcap -n 0 (2.7 Mbps)
```

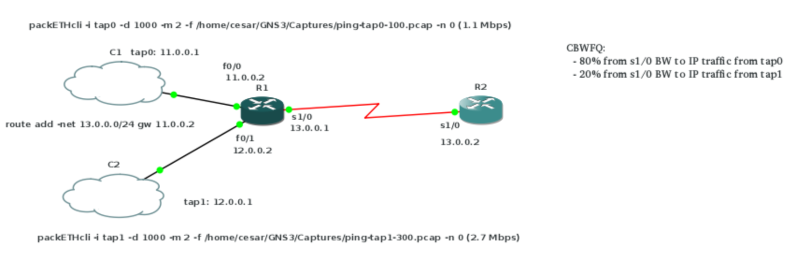
CBWFQ:
- 80% from s1/0 BW to IP traffic from tap0
- 20% from s1/0 BW to IP traffic from tap1

Requirements

- 80% bandwidth of `s1/0` reserved to IP traffic from `tap0`
- 20% bandwidth of `s1/0` reserved to IP traffic from `tap1`
- Being so, `max-reserved-bandwidth` must be set to 100%

Class-based WFQ (CBWFQ)

Example



```
packETHcli -i tap0 -d 1000 -m 2 -f /home/cesar/GNS3/Captures/ping-tap0-100.pcap -n 0 (1.1 Mbps)
```

```
route add -net 13.0.0.0/24 gw 11.0.0.2
```

```
packETHcli -i tap1 -d 1000 -m 2 -f /home/cesar/GNS3/Captures/ping-tap1-300.pcap -n 0 (2.7 Mbps)
```

CBWFQ:
- 80% from s1/0 BW to IP traffic from tap0
- 20% from s1/0 BW to IP traffic from tap1

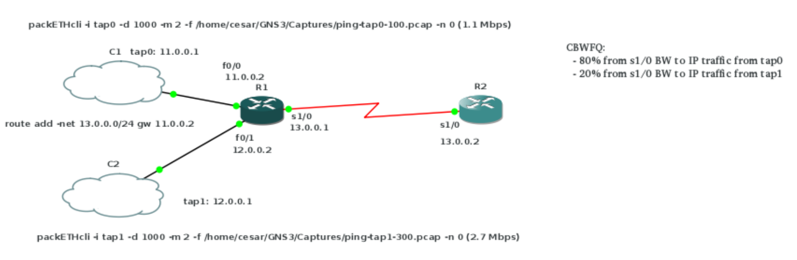
R1 configuration

```
access-list 101 permit ip 11.0.0.0 0.0.0.255 any
access-list 102 permit ip 12.0.0.0 0.0.0.255 any
!
class-map match-all class1
   match access-group 101
class-map match-all class2
   match access-group 102
```

```
policy-map policy1
   class class1
     bandwidth percent 80
   class class2
     bandwidth percent 20
!
interface Serial1/0
   ip address 13.0.0.1 255.255.255.0
   max-reserved-bandwidth 100
   service-policy output policy1
```

Class-based WFQ (CBWFQ)

Example



```
packETHcli -i tap0 -d 1000 -m 2 -f /home/cesar/GNS3/Captures/ping-tap0-100.pcap -n 0 (1.1 Mbps)
```

```
route add -net 13.0.0.0/24 gw 11.0.0.2
```

```
packETHcli -i tap1 -d 1000 -m 2 -f /home/cesar/GNS3/Captures/ping-tap1-300.pcap -n 0 (2.7 Mbps)
```

CBWFQ:
- 80% from s1/0 BW to IP traffic from tap0
- 20% from s1/0 BW to IP traffic from tap1

Run test

i/f	# Packets	IP length (bytes)	Traffic volume (bytes)
tap0	1,917	128	245,376
tap1	195	328	63,960

$$\text{Traffic ratio} = \frac{245{,}376}{63{,}960} = 3.83 \simeq \frac{80}{20}$$

Class-based WFQ (CBWFQ)

Queue monitoring

```
R1# show queueing interface Serial 1/0
Interface Serial1/0 queueing strategy: fair
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 14010
  Queueing strategy: Class-based queueing
  Output queue: 129/1000/64/14010 (size/max total/threshold/drops)
     Conversations  3/3/256 (active/max active/max total)
     Reserved Conversations 2/2 (allocated/max allocated)
     Available Bandwidth 1 kilobits/sec

  (depth/weight/total drops/no-buffer drops/interleaves) 64/20/5634/0/0
  Conversation 265, linktype: ip, length: 132
  source: 11.0.0.1, destination: 13.0.0.2, id: 0x3F9E, ttl: 63, prot: 1

  (depth/weight/total drops/no-buffer drops/interleaves) 64/78/8380/0/0
  Conversation 266, linktype: ip, length: 332
  source: 12.0.0.1, destination: 13.0.0.2, id: 0x3FFA, ttl: 63, prot: 1

  (depth/weight/total drops/no-buffer drops/interleaves) 1/32384/0/0/0
  Conversation 257, linktype: cdp, length: 333
```

Custom queueing (CQ)

Characteristics

- Up to 16 configurable queues
- Configurable parameters:
  - **limit**: max number of packet per queue (default 20)
  - **byte-count**: counts the number of bytes per queue served at each round. If limit is reached while transmitting a packet, the remaining packet is transmitted
- Queues served in a **round-robin** fashion
- Guaranteed bandwidth can be easily derived
- Packet **classification**: based on **protocol** type or **interfaces**
- A **default queue** can be assigned to non-matching traffic

Custom queueing (CQ)

Assigning the byte-count. An example

Assume we want to allocate 3 traffic flows as follows (IP lengths are known and supposed fixed):

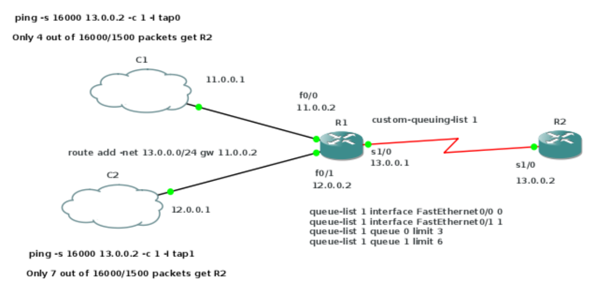
Traffic	IP length (L) (bytes)	BW reserved (B) (%)
A	200	30
B	450	50
C	1,500	20

We proceed:

Traffic	B/L	Normalized B/L (N)	byte-count = N·L
A	0.150	11.2	2,240
B	0.111	8.3	3,735
C	0.013	1.0	1,500

Custom queueing (CQ)

Example



```
ping -s 16000 13.0.0.2 -c 1 -I tap0
```

Only 4 out of 16000/1500 packets get R2

```
route add -net 13.0.0.0/24 gw 11.0.0.2
```

```
custom-queuing-list 1
```

```
queue-list 1 interface FastEthernet0/0 0
queue-list 1 interface FastEthernet0/1 1
queue-list 1 queue 0 limit 3
queue-list 1 queue 1 limit 6
```

```
ping -s 16000 13.0.0.2 -c 1 -I tap1
```

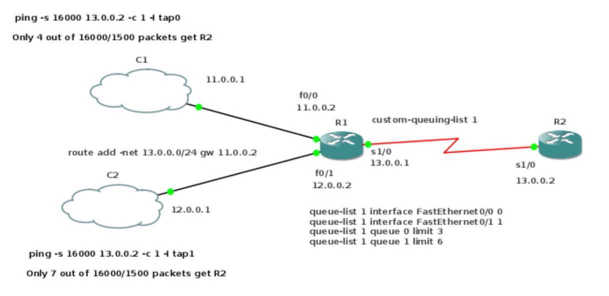
Only 7 out of 16000/1500 packets get R2

Queue monitoring

```
R1#show interfaces Serial 1/0
Serial1/0 is up, line protocol is up
....
  Output queues: (queue #: size/max/drops)
     0: 0/3/7 1: 0/6/3 2: 0/20/0 3: 0/20/0 4: 0/20/0
     5: 0/20/0 6: 0/20/0 7: 0/20/0 8: 0/20/0 9: 0/20/0
     10: 0/20/0 11: 0/20/0 12: 0/20/0 13: 0/20/0 14: 0/20/0
     15: 0/20/0 16: 0/20/0
```

Custom queueing (CQ)

Example



```
ping -s 16000 13.0.0.2 -c 1 -I tap0
```

Only 4 out of 16000/1500 packets get R2

```
route add -net 13.0.0.0/24 gw 11.0.0.2
```

```
custom-queuing-list 1
```

```
queue-list 1 interface FastEthernet0/0 0
queue-list 1 interface FastEthernet0/1 1
queue-list 1 queue 0 limit 3
queue-list 1 queue 1 limit 6
```

```
ping -s 16000 13.0.0.2 -c 1 -I tap1
```

Only 7 out of 16000/1500 packets get R2

Queue monitoring

```
R1# clear counters Serial 1/0

R1# show queueing interface Serial 1/0
Interface Serial1/0 queueing strategy: custom

Output queue utilization (queue/count)
    0/4 1/7 2/0 3/0 4/0 5/0 6/0 7/0 8/0
    9/0 10/0 11/0 12/0 13/0 14/0 15/0 16/0
```

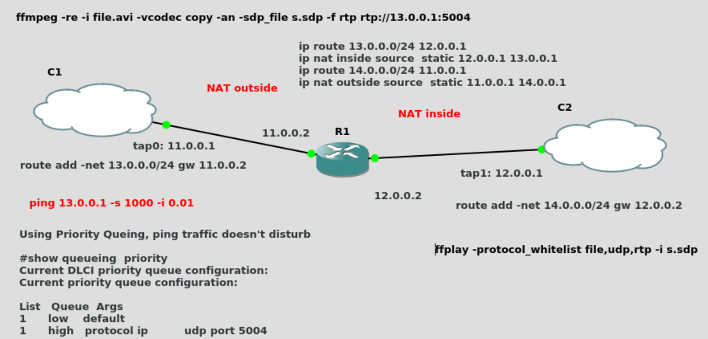
Priority queueing (PQ)

Characteristics

- Up to 4 queues. **High**, **medium**, **normal** and **low**
- PQ gives **absolute** priority. Highest priority queues are first processed until being emptied
  - Some lowest priority packets may be never sent
  - Use traffic shaping or CAR to avoid previous issue on higher priorities
- Packets classified as usual
- Not classified packets ingress **normal** priority queue
- PQ adds extra processing. Not acceptable for high speed i/fs

Priority queueing (PQ)

Example. Video streaming



```
ffmpeg -re -i file.avi -vcodec copy -an -sdp_file s.sdp -f rtp rtp://13.0.0.1:5004
```

```
ip route 13.0.0.0/24 12.0.0.1
ip nat inside source  static 12.0.0.1 13.0.0.1
ip route 14.0.0.0/24 11.0.0.1
ip nat outside source  static 11.0.0.1 14.0.0.1
```

```
route add -net 13.0.0.0/24 gw 11.0.0.2
```

```
ping 13.0.0.1 -s 1000 -i 0.01
```

```
route add -net 14.0.0.0/24 gw 12.0.0.2
```

Using Priority Queing, ping traffic doesn't disturb

```
ffplay -protocol_whitelist file,udp,rtp -i s.sdp
```

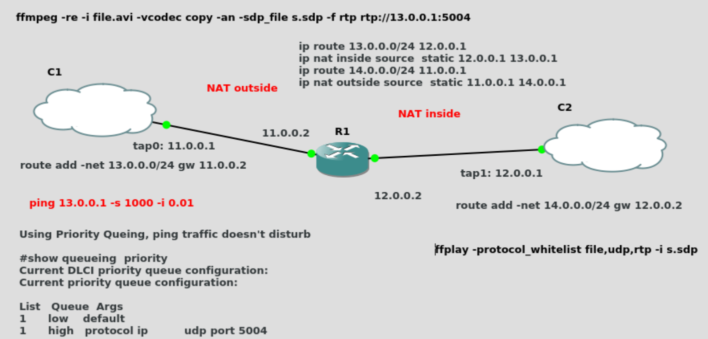
```
#show queueing  priority
Current DLCI priority queue configuration:
Current priority queue configuration:

List   Queue  Args
1      low    default
1      high   protocol ip         udp port 5004
```

NAT configuration

- Video streaming between 2 tap i/fs through R1
- `tap1` seen as 13.0.0.1 from `tap0`
- `tap0` seen as 14.0.0.1 from `tap1`

Priority queueing (PQ)

Example. Video streaming



```
ffmpeg -re -i file.avi -vcodec copy -an -sdp_file s.sdp -f rtp rtp://13.0.0.1:5004
```

```
ip route 13.0.0.0/24 12.0.0.1
ip nat inside source  static 12.0.0.1 13.0.0.1
ip route 14.0.0.0/24 11.0.0.1
ip nat outside source  static 11.0.0.1 14.0.0.1
```

```
route add -net 13.0.0.0/24 gw 11.0.0.2
```

```
route add -net 14.0.0.0/24 gw 12.0.0.2
```

```
ping 13.0.0.1 -s 1000 -i 0.01
```

Using Priority Queing, ping traffic doesn't disturb

```
ffplay -protocol_whitelist file,udp,rtp -i s.sdp
```

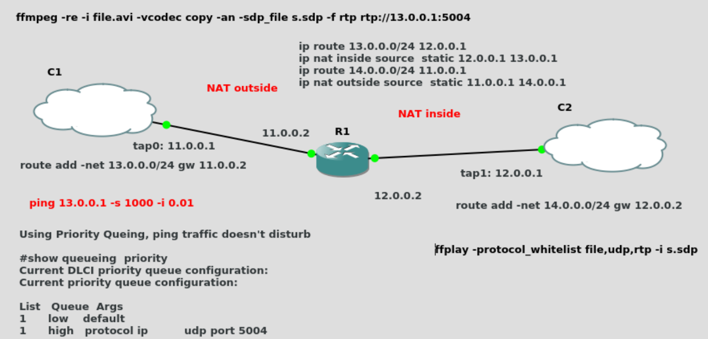
```
#show queueing  priority
Current DLCI priority queue configuration:
Current priority queue configuration:

List   Queue  Args
1      low    default
1      high   protocol ip        udp port 5004
```

Traffic configuration

- Video stream from `tap0` to `tap1` at 800 Kbps
- `ping` traffic would cause stream losses without PQ

Priority queueing (PQ)

Example. Video streaming



```
ffmpeg -re -i file.avi -vcodec copy -an -sdp_file s.sdp -f rtp rtp://13.0.0.1:5004
```

```
ip route 13.0.0.0/24 12.0.0.1
ip nat inside source  static 12.0.0.1 13.0.0.1
ip route 14.0.0.0/24 11.0.0.1
ip nat outside source  static 11.0.0.1 14.0.0.1
```

```
route add -net 13.0.0.0/24 gw 11.0.0.2
```

```
ping 13.0.0.1 -s 1000 -i 0.01
```

```
route add -net 14.0.0.0/24 gw 12.0.0.2
```

Using Priority Queing, ping traffic doesn't disturb

```
ffplay -protocol_whitelist file,udp,rtp -i s.sdp
```

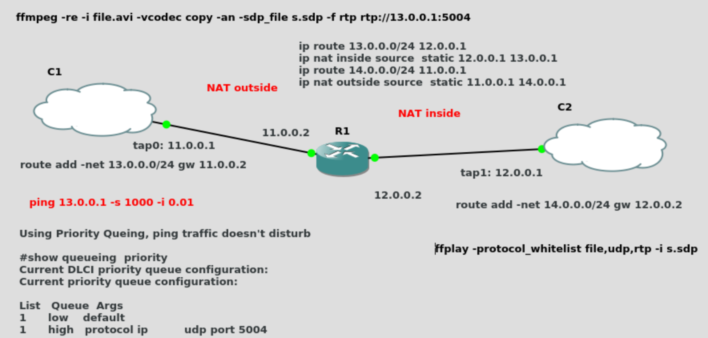
```
#show queueing  priority
Current DLCI priority queue configuration:
Current priority queue configuration:

List   Queue  Args
1      low    default
1      high   protocol ip         udp port 5004
```

R1 PQ configuration

```
interface FastEthernet0/1
 ip address 12.0.0.2 255.255.255.0
 ip nat inside
 priority-group 1
!
priority-list 1 protocol ip high udp 5004
priority-list 1 default low
```

Priority queueing (PQ)

Example. Video streaming



```
ffmpeg -re -i file.avi -vcodec copy -an -sdp_file s.sdp -f rtp rtp://13.0.0.1:5004
```

```
ip route 13.0.0.0/24 12.0.0.1
ip nat inside source  static 12.0.0.1 13.0.0.1
ip route 14.0.0.0/24 11.0.0.1
ip nat outside source  static 11.0.0.1 14.0.0.1
```

```
route add -net 13.0.0.0/24 gw 11.0.0.2
```

```
ping 13.0.0.1 -s 1000 -i 0.01
```

```
route add -net 14.0.0.0/24 gw 12.0.0.2
```

Using Priority Queing, ping traffic doesn't disturb

```
ffplay -protocol_whitelist file,udp,rtp -i s.sdp
```

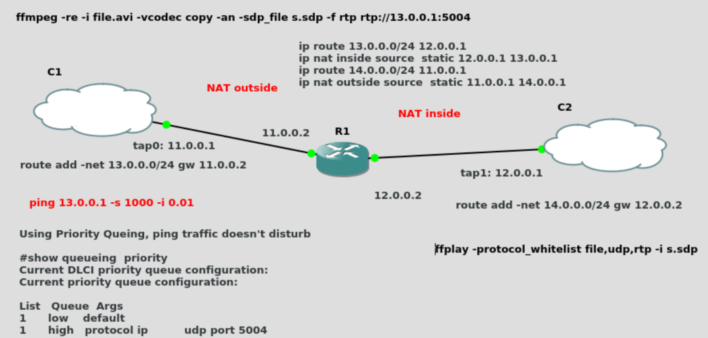
```
#show queueing  priority
Current DLCI priority queue configuration:
Current priority queue configuration:

List   Queue  Args
1      low    default
1      high   protocol ip          udp port 5004
```

PQ monitoring

```
R1# show queueing interface FastEthernet 0/1
Interface FastEthernet0/1 queueing strategy: priority

Output queue utilization (queue/count)
    high/2772 medium/0 normal/16254 low/862
```

Priority queueing (PQ)

Example. Video streaming



```
ffmpeg -re -i file.avi -vcodec copy -an -sdp_file s.sdp -f rtp rtp://13.0.0.1:5004
```

```
ip route 13.0.0.0/24 12.0.0.1
ip nat inside source  static 12.0.0.1 13.0.0.1
ip route 14.0.0.0/24 11.0.0.1
ip nat outside source  static 11.0.0.1 14.0.0.1
```

```
route add -net 13.0.0.0/24 gw 11.0.0.2
```

```
ping 13.0.0.1 -s 1000 -i 0.01
```

```
route add -net 14.0.0.0/24 gw 12.0.0.2
```

Using Priority Queing, ping traffic doesn't disturb

```
ffplay -protocol_whitelist file,udp,rtp -i s.sdp
```

```
#show queueing  priority
Current DLCI priority queue configuration:
Current priority queue configuration:

List   Queue  Args
1      low    default
1      high   protocol ip         udp port 5004
```

PQ monitoring

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

List    Queue   Args
1       low     default
1       high    protocol ip          udp port 5004
```

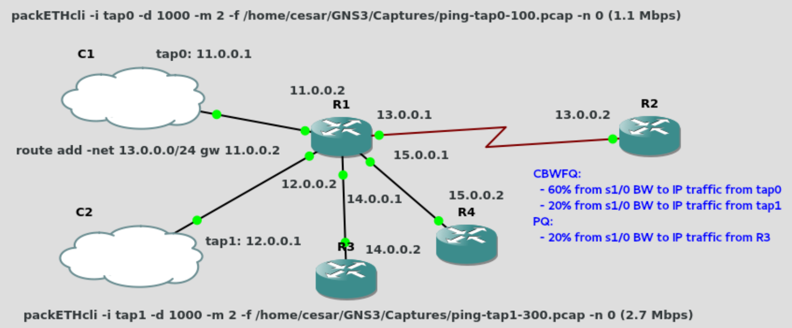
Low Latency Queueing (LLQ)

Overview

- LLQ adds PQ to CBWFQ
- Useful for real-time applications such as audio calls. Reduce jitter in voice conversations
- Voice comms uses UDP, not suitable to WRED congestion avoidance
- Packets in PQ are dequeued before those in WFQ queues
- LLQ uses a single priority queue within the CBWFQ classes

Low Latency Queueing (LLQ)

Example



```
packETHcli -i tap0 -d 1000 -m 2 -f /home/cesar/GNS3/Captures/ping-tap0-100.pcap -n 0 (1.1 Mbps)
```

```
route add -net 13.0.0.0/24 gw 11.0.0.2
```

CBWFQ:
- 60% from s1/0 BW to IP traffic from tap0
- 20% from s1/0 BW to IP traffic from tap1

PQ:
- 20% from s1/0 BW to IP traffic from R3

```
packETHcli -i tap1 -d 1000 -m 2 -f /home/cesar/GNS3/Captures/ping-tap1-300.pcap -n 0 (2.7 Mbps)
```

Low Latency Queueing (LLQ)

R1 configuration

```
access-list 101 permit ip 11.0.0.0 0.0.0.255 any
access-list 102 permit ip 12.0.0.0 0.0.0.255 any
access-list 103 permit ip 14.0.0.0 0.0.0.255 any

class-map match-all class1
 match access-group 101
class-map match-all class2
 match access-group 102
class-map match-all class3
 match access-group 103

policy-map policy1
 class class1
  bandwidth percent 60
 class class2
  bandwidth percent 20
 class class3
  priority percent 20  ← PQ
```

Low Latency Queueing (LLQ)

Queue monitoring

```
R1#show queueing interface Serial 1/0
Interface Serial1/0 queueing strategy: fair
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 215341
  Queueing strategy: Class-based queueing
  Output queue: 133/1000/64/215341 (size/max total/threshold/drops)
     Conversations  4/4/256 (active/max active/max total)
     Reserved Conversations 2/2 (allocated/max allocated)
     Available Bandwidth 2 kilobits/sec

  (depth/weight/total drops/no-buffer drops/interleaves) 5/0/106/0/0
  Conversation 264, linktype: ip, length: 1504
  source: 14.0.0.2, destination: 13.0.0.2, id: 0x0115, ttl: 254, prot: 1

  (depth/weight/total drops/no-buffer drops/interleaves) 64/26/103921/0/0
  Conversation 265, linktype: ip, length: 132
  source: 11.0.0.1, destination: 13.0.0.2, id: 0x3F9E, ttl: 63, prot: 1

  (depth/weight/total drops/no-buffer drops/interleaves) 64/78/111317/0/0
  Conversation 266, linktype: ip, length: 332
  source: 12.0.0.1, destination: 13.0.0.2, id: 0x3FFA, ttl: 63, prot: 1

  (depth/weight/total drops/no-buffer drops/interleaves) 3/32384/0/0/0
  Conversation 33, linktype: ip, length: 104
  source: 15.0.0.2, destination: 13.0.0.2, id: 0x001F, ttl: 254, prot: 1
```

PQ weight

Low Latency Queueing (LLQ)

ping from R3

```
R3# ping 13.0.0.2 size 100 repeat 10

Type escape sequence to abort.
Sending 10, 100-byte ICMP Echos to 13.0.0.2, timeout is 2 seconds:
!!!!!!!!!!
Success rate is 100 percent (10/10), round-trip min/avg/max = 52/60/68
```

ping from R4

```
R4 #ping 13.0.0.2 size 100 repeat 10

Type escape sequence to abort.
Sending 10, 100-byte ICMP Echos to 13.0.0.2, timeout is 2 seconds:
..........
Success rate is 0 percent (0/10)
```

Contents

1. QoS Overview
2. Classification
3. Congestion Management
4. Congestion Avoidance
   - Congestion avoidance overview
   - WRED
5. Policing and shaping
6. Resource Reservation Protocol
7. Bibliography

Congestion avoidance overview

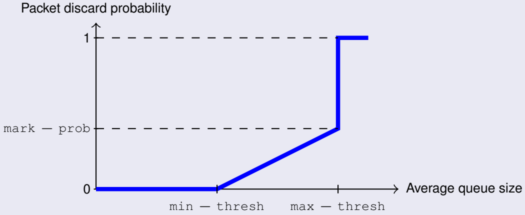
Congestion management tasks

- **RED** is used to prevent congestion
- **Tail-drop** as **default** if no RED configured
- CISCO implements a **weighted** version of RED (**WRED**), combining RED and IP Precedence. Weighted can be disabled, turning into a simple RED mechanism
- WRED additional features:
  - **Flow-based WRED**. More fairness to all flows
  - **Diffserv WRED**. Drop probabilities based on differentiated service code points (DCSP)

Congestion avoidance overview

RED fundamentals

- See chapter 1 (TCP congestion) to revisit RED mechanics
- Only effective in TCP flows
- Drops cause TCP not increasing advertise windows. Too much drops can put TCP into slow start
- Parameter names in CISCO configuration:



Computing the average queue length:

$$\texttt{AvgLen} = (1 - 2^{-n}) \cdot \texttt{AvgLen} + 2^{-n} \cdot \texttt{SampleLen}$$

n: `exponential-weighting-constant`

WRED

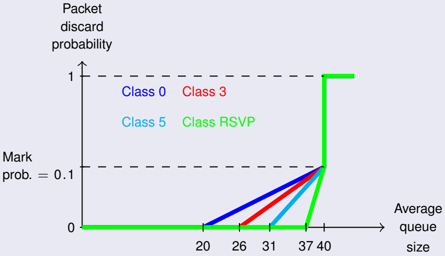
WRED basics

- A different probability profile applied to each IP precedence
- To turn WRED into RED, put the same values to all IP precedences

WRED default values

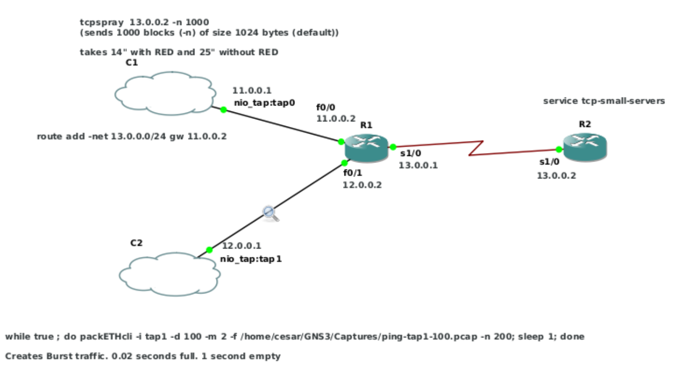
Exponential weighting constant (*n*): 9

Class	Min. Thresh.
0	20
1	22
2	24
3	26
4	28
5	31
6	33
7	35
RSVP	37



WRED

Example



tcpspray 13.0.0.2 -n 1000
(sends 1000 blocks (-n) of size 1024 bytes (default))

takes 14" with RED and 25" without RED

route add -net 13.0.0.0/24 gw 11.0.0.2

service tcp-small-servers

```
while true ; do packETHcli -i tap1 -d 100 -m 2 -f /home/cesar/GNS3/Captures/ping-tap1-100.pcap -n 200; sleep 1; done
```

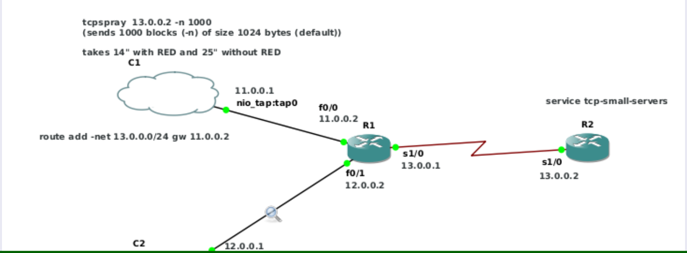
Creates Burst traffic. 0.02 seconds full. 1 second empty

R1 configuration

```
interface Serial1/0
  ip address 13.0.0.1 255.255.255.0
  random-detect
```

WRED

Example



tcpspray 13.0.0.2 -n 1000
(sends 1000 blocks (-n) of size 1024 bytes (default))

takes 14" with RED and 25" without RED

route add -net 13.0.0.0/24 gw 11.0.0.2

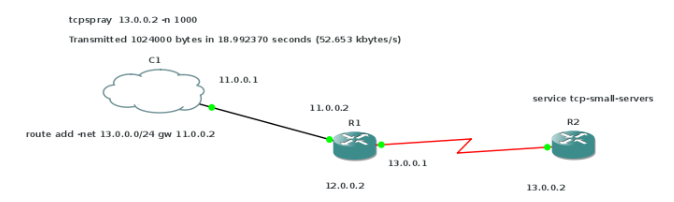
service tcp-small-servers

RED monitoring

```
R1#show queueing interface Serial 1/0
Interface Serial1/0 queueing strategy: random early detection (WRED)
    Random-detect not active on the dialer
    Exp-weight-constant: 9 (1/512)
    Mean queue depth: 0

  class    Random drop      Tail drop    Minimum Maximum  Mark
           pkts/bytes       pkts/bytes    thresh  thresh  prob
      0     104/16864          0/0          20      40   1/10
      1       0/0              0/0          22      40   1/10
      2       0/0              0/0          24      40   1/10
      3       0/0              0/0          26      40   1/10
      4       0/0              0/0          28      40   1/10
      5       0/0              0/0          31      40   1/10
      6       0/0              0/0          33      40   1/10
      7       0/0              0/0          35      40   1/10
   rsvp       0/0              0/0          37      40   1/10
```

WRED

Testing WRED



```
tcpspray 13.0.0.2 -n 1000
Transmitted 1024000 bytes in 18.992370 seconds (52.653 kbytes/s)
```

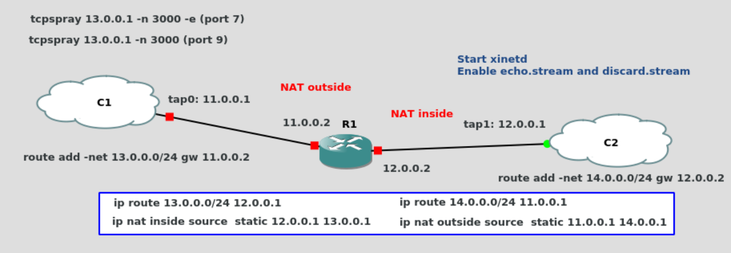
```
route add -net 13.0.0.0/24 gw 11.0.0.2
```

```
service tcp-small-servers
```

- CISCO `tcp-small-servers` not able to push tcspray to the limit
- Only 52 Kbytes/s out of 187 Kbytes/s (serial line)
- One should test between `tap` i/fs

WRED

Example 2



```
tcpspray 13.0.0.1 -n 3000 -e (port 7)
tcpspray 13.0.0.1 -n 3000 (port 9)
```

Start xinetd
Enable echo.stream and discard.stream

```
route add -net 13.0.0.0/24 gw 11.0.0.2
```

```
route add -net 14.0.0.0/24 gw 12.0.0.2
```

```
ip route 13.0.0.0/24 12.0.0.1
ip nat inside source  static 12.0.0.1 13.0.0.1
```

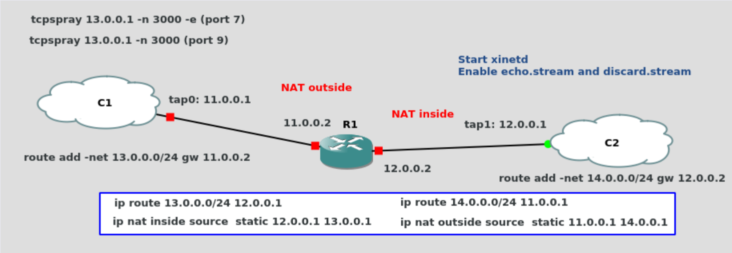
```
ip route 14.0.0.0/24 11.0.0.1
ip nat outside source  static 11.0.0.1 14.0.0.1
```

R1 config

```
interface FastEthernet0/0
 ip address 11.0.0.2 255.255.255.0
 ip nat outside
 ip policy route-map RM0

access-list 100 permit tcp any any eq discard
access-list 101 permit tcp any any eq echo
!
route-map RM0 permit 10
 match ip address 100
 set ip precedence priority
!
route-map RM0 permit 20
 match ip address 101
 set ip precedence network
```

WRED

Example 2



tcpspray 13.0.0.1 -n 3000 -e (port 7)

tcpspray 13.0.0.1 -n 3000 (port 9)

Start xinetd
Enable echo.stream and discard.stream

route add -net 13.0.0.0/24 gw 11.0.0.2

route add -net 14.0.0.0/24 gw 12.0.0.2

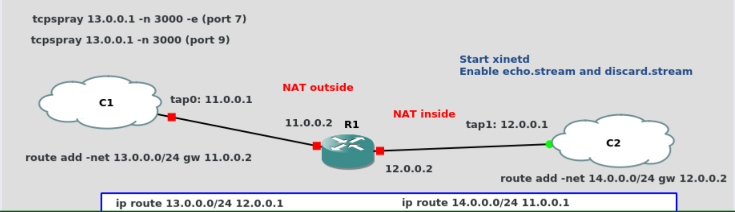
```
ip route 13.0.0.0/24 12.0.0.1                          ip route 14.0.0.0/24 11.0.0.1
ip nat inside source  static 12.0.0.1 13.0.0.1         ip nat outside source  static 11.0.0.1 14.0.0.1
```

TCP tests

```
$ tcpspray 13.0.0.1 -e  -n 3000
Received 3072000 bytes in 1.919753 second (1562.701 kbytes/s)
Transmitted 3072000 bytes in 1.453911 second (2063.400 kbytes/s)

$ tcpspray 13.0.0.1 -n 3000
Transmitted 3072000 bytes in 6.201178 seconds (483.779 kbytes/s)
```

WRED

Example 2



tcpspray 13.0.0.1 -n 3000 -e (port 7)

tcpspray 13.0.0.1 -n 3000 (port 9)

Start xinetd
Enable echo.stream and discard.stream

route add -net 13.0.0.0/24 gw 11.0.0.2

route add -net 14.0.0.0/24 gw 12.0.0.2

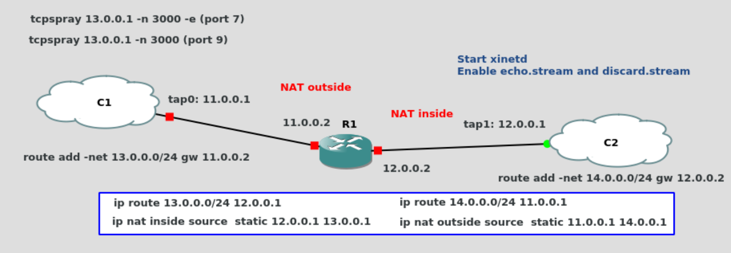
ip route 13.0.0.0/24 12.0.0.1 ip route 14.0.0.0/24 11.0.0.1

WRED monitoring

```
R1#show  queueing random-detect
Current random-detect configuration:
  FastEthernet0/1
    Queueing strategy: random early detection (WRED)
    Random-detect not active on the dialer
    Exp-weight-constant: 9 (1/512)
    Mean queue depth: 0

  class    Random drop       Tail drop    Minimum Maximum  Mark
           pkts/bytes        pkts/bytes    thresh  thresh  prob
      0      0/0               0/0          20      40   1/10
      1     146/221044        45/68130      22      40   1/10
      2      0/0               0/0          24      40   1/10
      3      0/0               0/0          26      40   1/10
      4      0/0               0/0          28      40   1/10
      5      0/0               0/0          31      40   1/10
      6      0/0               0/0          33      40   1/10
      7     47/55230          51/61286      35      40   1/10
   rsvp      0/0               0/0          37      40   1/10
```

WRED

Example 2



```
tcpspray 13.0.0.1 -n 3000 -e (port 7)
tcpspray 13.0.0.1 -n 3000 (port 9)
```

Start xinetd
Enable echo.stream and discard.stream

```
route add -net 13.0.0.0/24 gw 11.0.0.2
```

```
route add -net 14.0.0.0/24 gw 12.0.0.2
```

```
ip route 13.0.0.0/24 12.0.0.1                          ip route 14.0.0.0/24 11.0.0.1
ip nat inside source  static 12.0.0.1 13.0.0.1         ip nat outside source  static 11.0.0.1 14.0.0.1
```

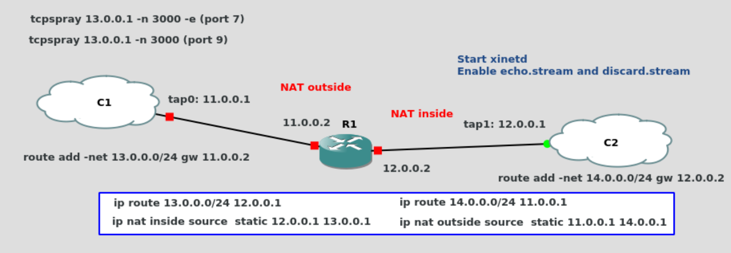
Modifying WRED parameters

```
interface FastEthernet0/1
 ip address 12.0.0.2 255.255.255.0
 ip nat inside
 random-detect
 random-detect precedence 7 23 40 10
```

IP prec   min-thresh   max-thresh   mark-prob denominator

WRED

Example 2



```
tcpspray 13.0.0.1 -n 3000 -e (port 7)
tcpspray 13.0.0.1 -n 3000 (port 9)
```

Start xinetd
Enable echo.stream and discard.stream

```
route add -net 13.0.0.0/24 gw 11.0.0.2
route add -net 14.0.0.0/24 gw 12.0.0.2
```

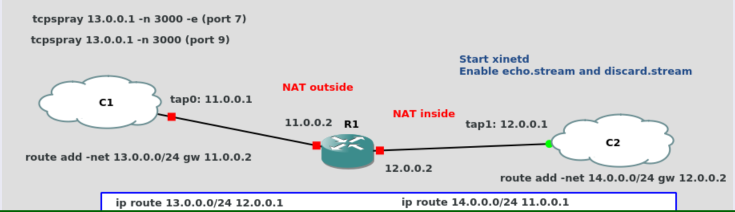
```
ip route 13.0.0.0/24 12.0.0.1                    ip route 14.0.0.0/24 11.0.0.1
ip nat inside source  static 12.0.0.1 13.0.0.1   ip nat outside source  static 11.0.0.1 14.0.0.1
```

TCP tests

```
$ tcpspray 13.0.0.1 -e  -n 3000
Received 3072000 bytes in 2.778369 seconds (1079.770 kbytes/s)
Transmitted 3072000 bytes in 2.549548 seconds (1176.679 kbytes/s)

$ tcpspray 13.0.0.1 -n 3000
Transmitted 3072000 bytes in 2.209408 seconds (1357.830 kbytes/s)
```

WRED

Example 2



tcpspray 13.0.0.1 -n 3000 -e (port 7)

tcpspray 13.0.0.1 -n 3000 (port 9)

Start xinetd
Enable echo.stream and discard.stream

route add -net 13.0.0.0/24 gw 11.0.0.2

route add -net 14.0.0.0/24 gw 12.0.0.2

ip route 13.0.0.0/24 12.0.0.1

ip route 14.0.0.0/24 11.0.0.1

WRED monitoring

```
R1#show  queueing random-detect
Current random-detect configuration:
  FastEthernet0/1
    Queueing strategy: random early detection (WRED)
    Random-detect not active on the dialer
    Exp-weight-constant: 9 (1/512)
    Mean queue depth: 0
  class          Random drop      Tail drop    Minimum Maximum  Mark
                  pkts/bytes       pkts/bytes    thresh  thresh   prob
      0      0/0              0/0              20      40  1/10
      1     43/65102          0/0              22      40  1/10
      2      0/0              0/0              24      40  1/10
      3      0/0              0/0              26      40  1/10
      4      0/0              0/0              28      40  1/10
      5      0/0              0/0              31      40  1/10
      6      0/0              0/0              33      40  1/10
      7     65/72358         28/22120          23      40  1/10
   rsvp      0/0              0/0              37      40  1/10
```

WRED

Througput results

Average throughput over 3 measures

Prec (-e)	Prec (-d)	Throughput (-e) (kB/s)	Throughput (-d) (kB/s)
7	0	1,480	955
6	0	1,496	990
5	0	1,512	821
4	0	1,390	896
3	0	1,360	931
2	0	1,357	1,062
1	0	1,177	1,182
0	0	1,010	1,730

Contents

1 QoS Overview

2 Classification

3 Congestion Management

4 Congestion Avoidance

5 **Policing and shaping**
- Policing and shaping overview
- Token bucket
- Traffic policing
- Traffic shaping

6 Resource Reservation Protocol

7 Bibliography

Policing and shaping overview

Policing and shaping are **traffic regulation** mechanisms

- **Policing**: non-compliant traffic is discarded. (ex. CAR policy seen before)
- **Shaping**: non-compliant traffic is shaped and transmitted

How compliance is determined ? : **Token bucket**

Token bucket

Definition

Token bucket is a formal definition of **data transfer rate**. 3 components:

- **Mean rate** (r): amount of data to be transferred per unit time **on average**. Also called *Committed Information Rate* (CIR)
- **Burst size** (b): amount of data that **can be** transferred in a given **time interval**
- **Time interval** (t): Burst size expressed in time units. Derived from mean rate and burst size

The following relation holds: $r = \frac{b}{t}$

Over any integral part of t, transmit rate must not exceed r. Inside t, may be arbitrarily fast.

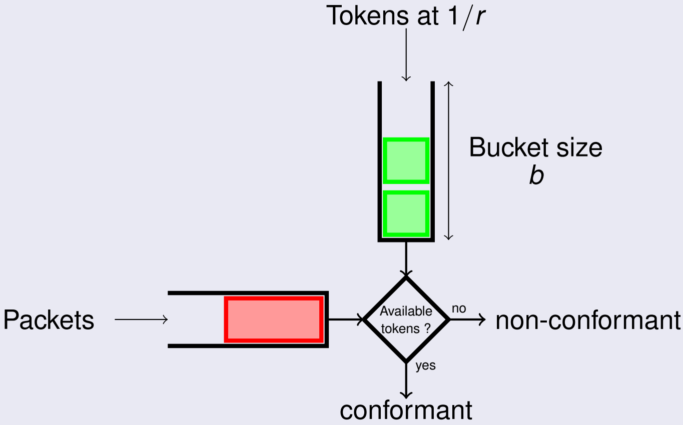
Token bucket

Algorithm

- A token is added to the bucket every $\frac{1}{r}$ seconds
- The bucket size is b. Tokens arriving when the bucket is full are discarded
- An arriving packet of size d is determined as:
  - **conformant**, if d is smaller than the number of tokens in the bucket
  - **non-conformant**, otherwise
- Conformant packets are transmitted. Bucket is decremented in d
- Non-conformant packets are discarded (policing) or delayed until enough tokens (shaping)

Token bucket

Algorithm



Token bucket

Token bucket with 2 buckets

- Exceeding bursts may be allowed
- An **exceed bucket** is added to the already existing **conform bucket**
- Overflowing tokens from conform bucket drops into the exceed bucket
- Tokens can be borrowed from exceed bucket if conform bucket is not enough
- In this case, 3 actions must be observed:
  - **Conformant**
  - **Exceeded**
  - **Violated**
- Conform bucket size (b_c). Exceed bucket size (b_e)
- $b_e = 0$ is a token bucket with 1 bucket

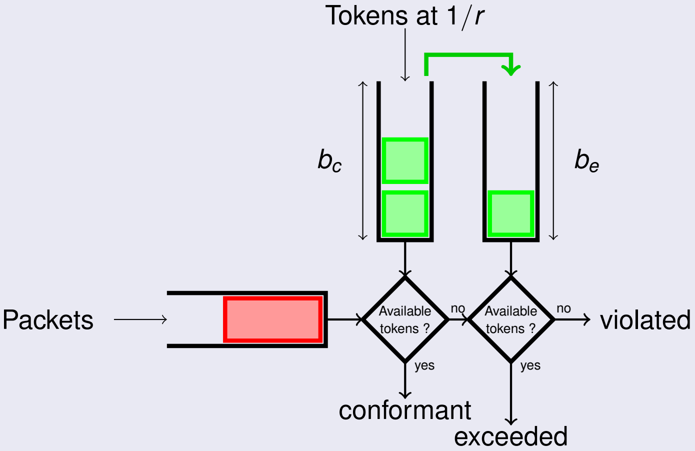
Token bucket

Algorithm with two buckets

- A token is added to the conform bucket every $\frac{1}{r}$ seconds
- Tokens arriving when the conform bucket is full fills into the exceed bucket
- An arriving packet of size d is determined as:
  - **conformant**, if d is smaller than the number of tokens in the conform bucket
  - **exceed**, if d is greater than the number of tokens in the conform bucket and smaller than the number of tokens in the conform and exceed bucket
  - **violated**, otherwise
- Conformant packets are transmitted. Conform bucket is decremented in d
- Exceeded packets are treated according to exceeding policy. Conform bucket is emptied and exceed bucket is decremented in $(d - b)$
- Violated packets are treated according to violating policy

Token bucket

Algorithm with 2 buckets



Token bucket

A numerical example

Tokens rate = 1. $b_c = 4$. $b_e = 6$

time	Packet Length	Conform bucket	Exceed bucket	Action
0	-	4	6	-
1	2	3	6	conform
2	-	4	6	-
3	5	1	5	exceed
4	-	2	5	-
5	5	1	2	exceed
6	4	2	2	violated
7	-	3	2	-
8	-	4	2	-
9	-	4	3	-

Traffic policing

Overview

- Traffic policing allows control of maximum **incoming or leaving** rate using token bucket
- Traffic can be partitioned into several classes
- Several actions on conforming, exceeding and violating traffic:
  - Drop
  - Transmit
  - Set IP precedence and transmit
  - Set DCSP value and transmit

Traffic policing

Configuration steps

- Configure a **class map**
- Configure a **police map**
- Configure token bucket parameters inside a policy map:
  - Average rate (in bps or as a fraction of the bandwidth)
  - Conformant bucket size (b_c) (in bytes)
  - Excess burst parameter ($b_c + b_e$) (in bytes). If excess burst parameter equals b_c, then $b_e = 0$
  - Conform, exceed and violate actions

Traffic policing

Configuration example

CAR example (here) can be also configured as follows:

```
access-list 1 permit any
class-map match-all CMa
 match access-group 1

policy-map PMa
 class CMa
   police 8000 2000 2000 conform-action set-prec-transmit 7
            exceed-action set-prec-transmit 1

interface Serial1/0
 ip address 13.0.0.1 255.255.255.0
 service-policy output PMa
```

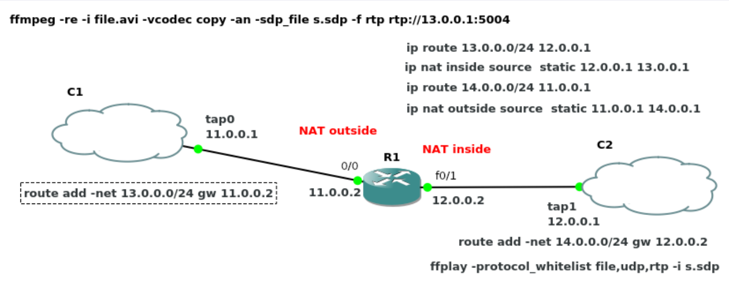
Traffic shaping

Overview

- Traffic shaping allows modify the **leaving** traffic profile to commit a given rate using token bucket
- Being so, we ensure traffic conforms certain policies
- As a result, traffic may suffers delays
- Shaping may be done based on ACLs or traffic classes

Traffic shaping

Shaping example. Video streaming



```
ffmpeg -re -i file.avi -vcodec copy -an -sdp_file s.sdp -f rtp rtp://13.0.0.1:5004
```

```
ip route 13.0.0.0/24 12.0.0.1
ip nat inside source  static 12.0.0.1 13.0.0.1
ip route 14.0.0.0/24 11.0.0.1
ip nat outside source  static 11.0.0.1 14.0.0.1
```

```
route add -net 13.0.0.0/24 gw 11.0.0.2
```

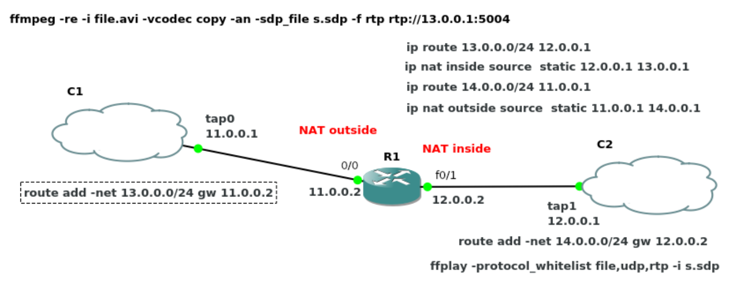
```
route add -net 14.0.0.0/24 gw 12.0.0.2
```

```
ffplay -protocol_whitelist file,udp,rtp -i s.sdp
```

- Video stream at a 800 Kbps rate
- Shaping configured at i/f `f0/1` at `R1`. 800 Kbps mean rate. No exceed bucket

Traffic shaping

Shaping example. Video streaming



```
ffmpeg -re -i file.avi -vcodec copy -an -sdp_file s.sdp -f rtp rtp://13.0.0.1:5004
```

```
ip route 13.0.0.0/24 12.0.0.1
ip nat inside source  static 12.0.0.1 13.0.0.1
ip route 14.0.0.0/24 11.0.0.1
ip nat outside source  static 11.0.0.1 14.0.0.1
```

```
route add -net 13.0.0.0/24 gw 11.0.0.2
```

```
route add -net 14.0.0.0/24 gw 12.0.0.2
```

```
ffplay -protocol_whitelist file,udp,rtp -i s.sdp
```

R1 config

```
interface FastEthernet0/1
  ip address 12.0.0.2 255.255.255.0
  ip nat inside
  traffic-shape rate 800000 100000 100000
```

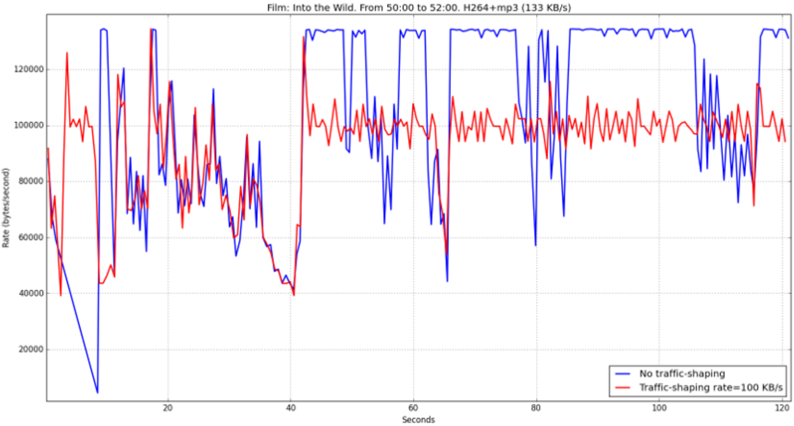
Average rate (bps)

b_c (bits)

$b_c + b_e$ (bits)

Traffic shaping

Traffic at `R1 f0/1`



Traffic shaping

Traffic shaping monitoring

```
R1#show traffic-shape

Interface   Fa0/1
       Access Target    Byte   Sustain   Excess    Interval  Increment
VC     List   Rate      Limit  bits/int  bits/int  (ms)      (bytes)
-             800000    25000  100000    100000    125       12500
```

r: average rate (bps)

b_c (bits)

$b_c + b_e$ (bits)

t: time interval (ms)

$$r = \frac{b_c}{t}$$

Traffic shaping

Traffic shaping monitoring

```
R1# show traffic-shape statistics
                  Acc. Queue Packets   Bytes     Packets   Bytes     Shaping
I/F               List Depth                     Delayed   Delayed   Active
Fa0/1
                   0     3246      4342043   592       805842    no
R1# show traffic-shape statistics
                  Acc. Queue Packets   Bytes     Packets   Bytes     Shaping
I/F               List Depth                     Delayed   Delayed   Active
Fa0/1
                   0     3268      4372183   592       805842    no
R1# show traffic-shape statistics
                  Acc. Queue Packets   Bytes     Packets   Bytes     Shaping
I/F               List Depth                     Delayed   Delayed   Active
Fa0/1                   0     3308      4426983   595       809952    yes
```

Contents

1. QoS Overview
2. Classification
3. Congestion Management
4. Congestion Avoidance
5. Policing and shaping
6. **Resource Reservation Protocol**
   - Overview
   - The protocol
   - Features
   - Configuration and monitoring
7. Bibliography

Overview

What is it ?

- RSVP is an implementation of **Integrated Services**
- **Signaling protocol**. By itself doesn't provide QoS
- Clients use RSVP to apply for QoS for a **session**
- A session consists of:
  - Destination address (unicast or multicast)
  - IP protocol
  - Destination port
- Layer 4 protocol on top IP
- Routers must implement QoS through WFQ, WRED, LLQ, . . .

Overview

how does it work ? (I)

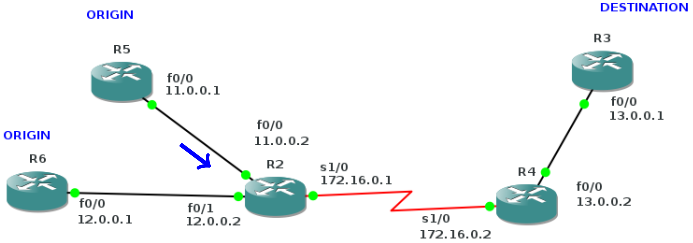
- RSVP reserves in a direction only (origin →destination)
- A route between **origin** and **destination** is established by routing protocols
- All the traversed routers must be informed about reservations
- Basic steps:
  1. **PATH** message initiated by origin to destination. This message includes:
     - Session parameters
     - QoS requirements (required rate, burst, delay, . . .)
     - Nodes IP traversed
  2. **RESV** message from destination to origin following the same route (reverse) that PATH.
     - At each node (router), resource availability is determined
     - If enough resources, RESV message is forwarded to next node
     - If reservation is declined (not enough resources), an error message is sent to origin

Overview

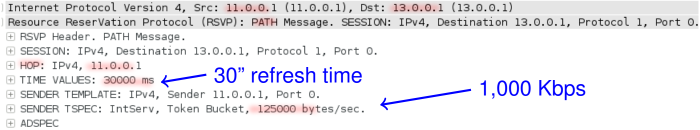
how does it work ? (II)

- **Session maintenance**. PATH and RESV messages are periodically refreshed. 30” by default
- **Confirmation**. At each node, after a RESV, a CONFIRM message is sent to destination
- **Finishing reservations**. Can be finished by origin, destination or intermediate nodes
  - **PathTear** messages sent in PATH direction
  - **ResvTear** messages sent in RESV direction
- Tear messages **free** resources

The protocol



PATH message. Asks for a reservation: 1,000 Kbps (average rate) and 100 KBytes (Burst size). Protocol 1 (ICMP), any port (0)

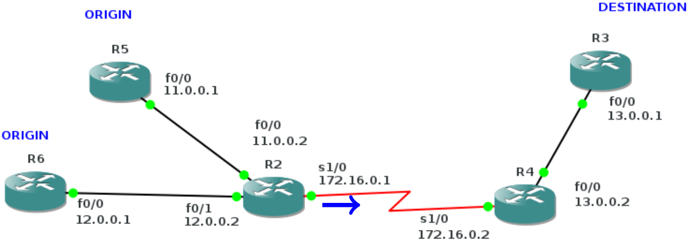


```
Internet Protocol Version 4, Src: 11.0.0.1 (11.0.0.1), Dst: 13.0.0.1 (13.0.0.1)
Resource ReserVation Protocol (RSVP): PATH Message. SESSION: IPv4, Destination 13.0.0.1, Protocol 1, Port 0.
 ⊞ RSVP Header. PATH Message.
 ⊞ SESSION: IPv4, Destination 13.0.0.1, Protocol 1, Port 0.
 ⊞ HOP: IPv4, 11.0.0.1
 ⊞ TIME VALUES: 30000 ms
 ⊞ SENDER TEMPLATE: IPv4, Sender 11.0.0.1, Port 0.
 ⊞ SENDER TSPEC: IntServ, Token Bucket, 125000 bytes/sec.
 ⊞ ADSPEC
```

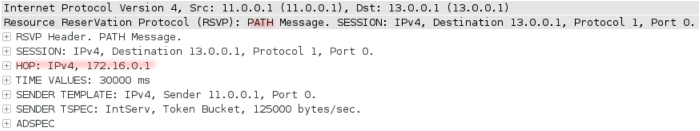
30” refresh time

1,000 Kbps

The protocol

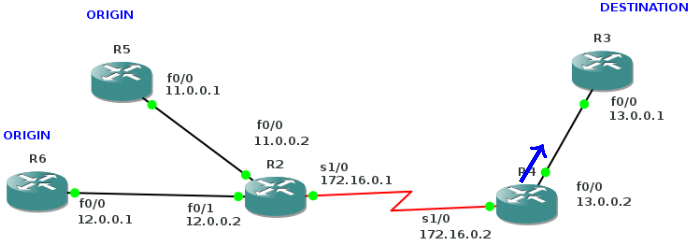


PATH forwarded to next node. Hop field changed

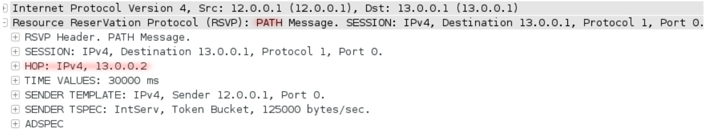


```
Internet Protocol Version 4, Src: 11.0.0.1 (11.0.0.1), Dst: 13.0.0.1 (13.0.0.1)
Resource ReserVation Protocol (RSVP): PATH Message. SESSION: IPv4, Destination 13.0.0.1, Protocol 1, Port 0.
⊞ RSVP Header. PATH Message.
⊞ SESSION: IPv4, Destination 13.0.0.1, Protocol 1, Port 0.
⊞ HOP: IPv4, 172.16.0.1
⊞ TIME VALUES: 30000 ms
⊞ SENDER TEMPLATE: IPv4, Sender 11.0.0.1, Port 0.
⊞ SENDER TSPEC: IntServ, Token Bucket, 125000 bytes/sec.
⊞ ADSPEC
```

The protocol

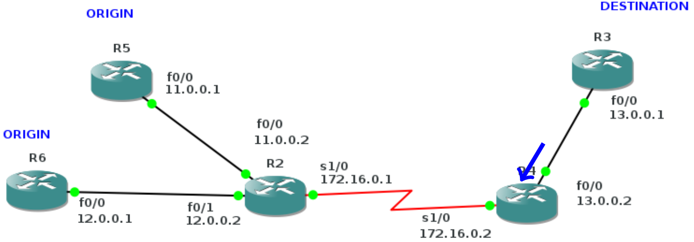


PATH forwarded to destination node

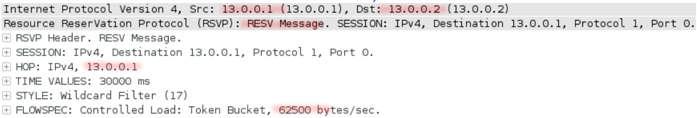


```
⊞ Internet Protocol Version 4, Src: 12.0.0.1 (12.0.0.1), Dst: 13.0.0.1 (13.0.0.1)
⊟ Resource ReserVation Protocol (RSVP): PATH Message. SESSION: IPv4, Destination 13.0.0.1, Protocol 1, Port 0.
  ⊞ RSVP Header. PATH Message.
  ⊞ SESSION: IPv4, Destination 13.0.0.1, Protocol 1, Port 0.
  ⊞ HOP: IPv4, 13.0.0.2
  ⊞ TIME VALUES: 30000 ms
  ⊞ SENDER TEMPLATE: IPv4, Sender 12.0.0.1, Port 0.
  ⊞ SENDER TSPEC: IntServ, Token Bucket, 125000 bytes/sec.
  ⊞ ADSPEC
```

The protocol

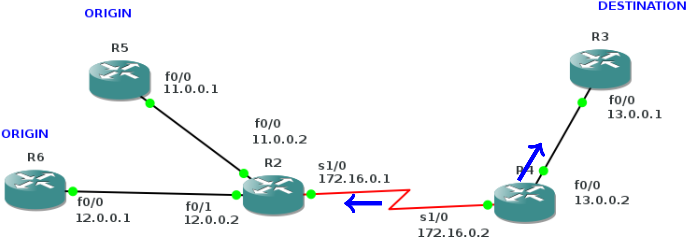


RESV sent to **next hop**. Only 500 Kbps reserved

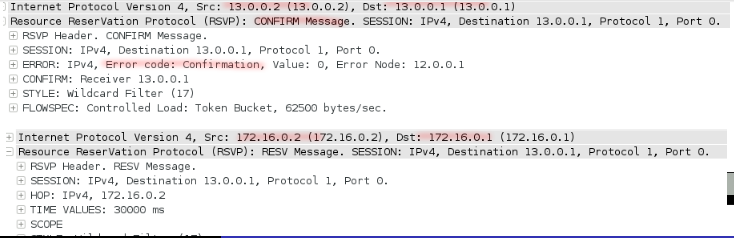


```
Internet Protocol Version 4, Src: 13.0.0.1 (13.0.0.1), Dst: 13.0.0.2 (13.0.0.2)
Resource ReserVation Protocol (RSVP): RESV Message. SESSION: IPv4, Destination 13.0.0.1, Protocol 1, Port 0.
⊞ RSVP Header. RESV Message.
⊞ SESSION: IPv4, Destination 13.0.0.1, Protocol 1, Port 0.
⊞ HOP: IPv4, 13.0.0.1
⊞ TIME VALUES: 30000 ms
⊞ STYLE: Wildcard Filter (17)
⊞ FLOWSPEC: Controlled Load: Token Bucket, 62500 bytes/sec.
```

The protocol

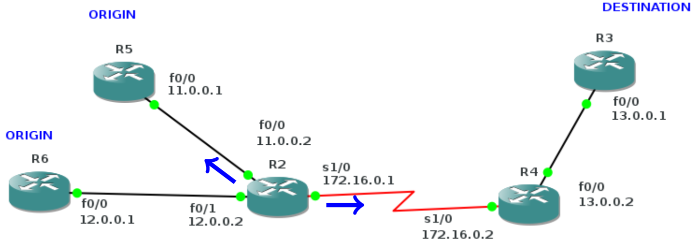


CONFIRM to **destination** if enough resources on R4. RESV sent to **next hop**

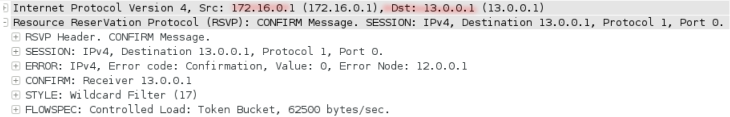


```
] Internet Protocol Version 4, Src: 13.0.0.2 (13.0.0.2), Dst: 13.0.0.1 (13.0.0.1)
] Resource ReserVation Protocol (RSVP): CONFIRM Message. SESSION: IPv4, Destination 13.0.0.1, Protocol 1, Port 0.
  ⊞ RSVP Header. CONFIRM Message.
  ⊞ SESSION: IPv4, Destination 13.0.0.1, Protocol 1, Port 0.
  ⊞ ERROR: IPv4, Error code: Confirmation, Value: 0, Error Node: 12.0.0.1
  ⊞ CONFIRM: Receiver 13.0.0.1
  ⊞ STYLE: Wildcard Filter (17)
  ⊞ FLOWSPEC: Controlled Load: Token Bucket, 62500 bytes/sec.

⊞ Internet Protocol Version 4, Src: 172.16.0.2 (172.16.0.2), Dst: 172.16.0.1 (172.16.0.1)
⊟ Resource ReserVation Protocol (RSVP): RESV Message. SESSION: IPv4, Destination 13.0.0.1, Protocol 1, Port 0.
  ⊞ RSVP Header. RESV Message.
  ⊞ SESSION: IPv4, Destination 13.0.0.1, Protocol 1, Port 0.
  ⊞ HOP: IPv4, 172.16.0.2
  ⊞ TIME VALUES: 30000 ms
  ⊞ SCOPE
  ⊞ STYLE: Wildcard Filter (17)
```

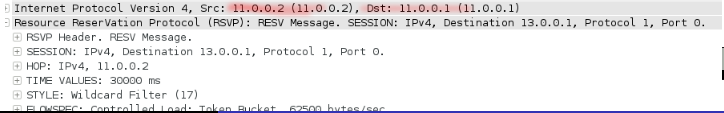
The protocol



CONFIRM to **destination** if enough resources on R2. RESV sent to **next hop**

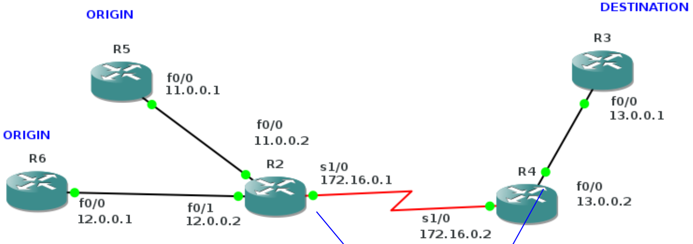


```
⊞ Internet Protocol Version 4, Src: 172.16.0.1 (172.16.0.1), Dst: 13.0.0.1 (13.0.0.1)
⊟ Resource ReserVation Protocol (RSVP): CONFIRM Message. SESSION: IPv4, Destination 13.0.0.1, Protocol 1, Port 0.
  ⊞ RSVP Header. CONFIRM Message.
  ⊞ SESSION: IPv4, Destination 13.0.0.1, Protocol 1, Port 0.
  ⊞ ERROR: IPv4, Error code: Confirmation, Value: 0, Error Node: 12.0.0.1
  ⊞ CONFIRM: Receiver 13.0.0.1
  ⊞ STYLE: Wildcard Filter (17)
  ⊞ FLOWSPEC: Controlled Load: Token Bucket, 62500 bytes/sec.
```



```
⊞ Internet Protocol Version 4, Src: 11.0.0.2 (11.0.0.2), Dst: 11.0.0.1 (11.0.0.1)
⊟ Resource ReserVation Protocol (RSVP): RESV Message. SESSION: IPv4, Destination 13.0.0.1, Protocol 1, Port 0.
  ⊞ RSVP Header. RESV Message.
  ⊞ SESSION: IPv4, Destination 13.0.0.1, Protocol 1, Port 0.
  ⊞ HOP: IPv4, 11.0.0.2
  ⊞ TIME VALUES: 30000 ms
  ⊞ STYLE: Wildcard Filter (17)
  ⊞ FLOWSPEC: Controlled Load: Token Bucket, 62500 bytes/sec.
```

The protocol



Reservation at R2

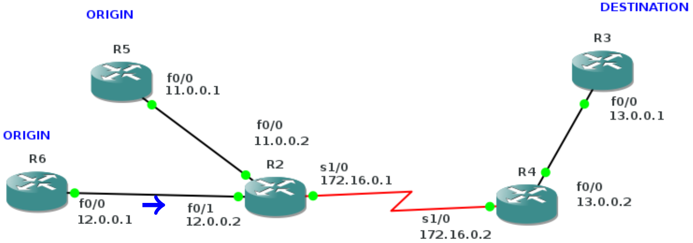
```
To            From          Pro DPort Sport Next Hop      I/F      Fi Serv BPS
13.0.0.1      11.0.0.1       1    0     0    172.16.0.2     Se1/0    WF LOAD 500K
```

Reservation at R4

```
To            From          Pro DPort Sport Next Hop      I/F      Fi Serv BPS
13.0.0.1      11.0.0.1       1    0     0    13.0.0.1       Fa0/0    WF LOAD 500K
```

Reserved resources at output interfaces from origin to destination

The protocol



After a second reservation accepted from R6 results:

Reservation at R2

```
To            From          Pro DPort Sport Next Hop      I/F      Fi Serv BPS
13.0.0.1      11.0.0.1       1   0     0     172.16.0.2    Se1/0    WF LOAD 500K
13.0.0.1      12.0.0.1       1   0     0     172.16.0.2    Se1/0    WF LOAD 500K
```

Reservation at R4

```
To            From          Pro DPort Sport Next Hop      I/F      Fi Serv BPS
13.0.0.1      11.0.0.1       1   0     0     13.0.0.1     Fa0/0    WF LOAD 500K
13.0.0.1      12.0.0.1       1   0     0     13.0.0.1     Fa0/0    WF LOAD 500K
```

Features

Integrated services

- 2 type of services can be reserved:
  - Guaranteed-rate
  - Controlled-load
- **Guaranteed-rate**. Offered service as an unloaded network according to bandwidth requirements. Delay tolerant services. CISCO implements it using WFQ with weights proportional to bandwidth
- **Controlled-load**. Delivers assured bandwidth with constant delay. Implemented with WRED (not confirmed by experimentation)
- Both types of service may use LLQ. Reservations with rate and burst size below some threshold are considered priority and put into **priority queue** (Assigned weight is 0)

Features

Reservation styles

- A reservation belongs to a **class** and a **scope**
- Two classes:
  1. **Shared**. A single reservation is made for **multiple upstream senders**
  2. **Distinct**. A reservation established for each sender
- Two scopes:
  1. **Explicit**. The reservation is defined by a explicit **list of senders**
  2. **Wildcard**. Some wildcard (0) used to define multiple senders
- A **sender** consists of an origin IP and origin port

Features

Reservation styles

Such a combination of classes and scopes leads to **three** reservation styles

	Classes	
Scope	Distinct	Shared
Explicit	fixed-filter (**FF**)	shared-explicit (**SE**)
Wildcard	-	Wildcard-filter (**WF**)

Features

Reservation styles

Such a combination of classes and scopes leads to **three** reservation styles

	Classes	
Scope	Distinct	Shared
Explicit	fixed-filter (**FF**)	shared-explicit (**SE**)
Wildcard	-	Wildcard-filter (**WF**)

FF:

- Reservation not shared by any other senders
- If another **receiver** is added for the same sender, reservations are **merged**
- Example: video broadcast

Features

Reservation styles

Such a combination of classes and scopes leads to **three** reservation styles

Scope	Classes	
	Distinct	Shared
Explicit	fixed-filter (**FF**)	shared-explicit (**SE**)
Wildcard	-	Wildcard-filter (**WF**)

SE:

- Reservation shared by other senders
- Senders explicitly specified by the receiver

Features

Reservation styles

Such a combination of classes and scopes leads to **three** reservation styles

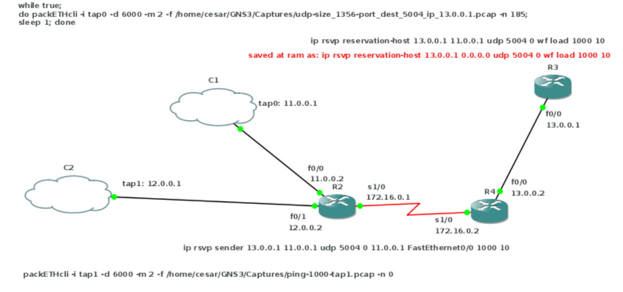
Scope	Classes	
	Distinct	Shared
Explicit	fixed-filter (**FF**)	shared-explicit (**SE**)
Wildcard	-	Wildcard-filter (**WF**)

WF:

- Reservation shared by other senders
- Senders specified by a wildcard
- WF and SE reservations useful for audio conference multicast. No more than one link active at the same time

Configuration and monitoring

Example 1



```
while true;
do packETHcli -i tap0 -d 6000 -m 2 -f /home/cesar/GNS3/Captures/udp-size_1356-port_dest_5004_ip_13.0.0.1.pcap -n 185;
sleep 1; done
```

```
ip rsvp reservation-host 13.0.0.1 11.0.0.1 udp 5004 0 wf load 1000 10
```

saved at ram as: ip rsvp reservation-host 13.0.0.1 0.0.0.0 udp 5004 0 wf load 1000 10

```
ip rsvp sender 13.0.0.1 11.0.0.1 udp 5004 0 11.0.0.1 FastEthernet0/0 1000 10
```

```
packETHcli -i tap1 -d 6000 -m 2 -f /home/cesar/GNS3/Captures/ping-1000-tap1.pcap -n 0
```

Enabling RSVP at interfaces (R2)

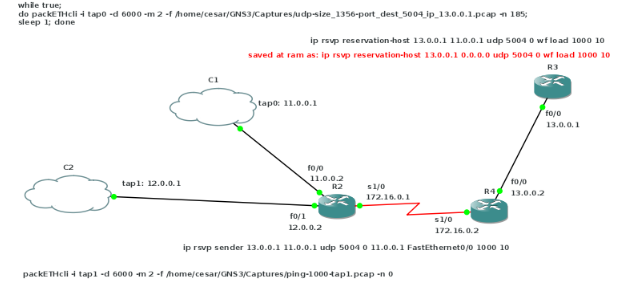
```
interface FastEthernet0/0
 ip address 11.0.0.2 255.255.255.0
 ip rsvp bandwidth 1200 1200
!
interface Serial1/0
 ip address 172.16.0.1 255.255.255.0
 ip rsvp bandwidth 1150 1150
```

Maximum amount of reservable BW per flow (Kbps) Default: entire BW

Maximum amount of reservable BW per i/f. 75% as default (Kbps)

Configuration and monitoring

Example 1



```
while true;
do packETHcli -i tap0 -d 6000 -m 2 -f /home/cesar/GNS3/Captures/udp-size_1356-port_dest_5004_ip_13.0.0.1.pcap -n 185;
sleep 1; done
```

```
ip rsvp reservation-host 13.0.0.1 11.0.0.1 udp 5004 0 wf load 1000 10
```

saved at ram as: ip rsvp reservation-host 13.0.0.1 0.0.0.0 udp 5004 0 wf load 1000 10

```
ip rsvp sender 13.0.0.1 11.0.0.1 udp 5004 0 11.0.0.1 FastEthernet0/0 1000 10
```

```
packETHcli -i tap1 -d 6000 -m 2 -f /home/cesar/GNS3/Captures/ping-1000-tap1.pcap -n 0
```

Enabling RSVP at interfaces (R4)

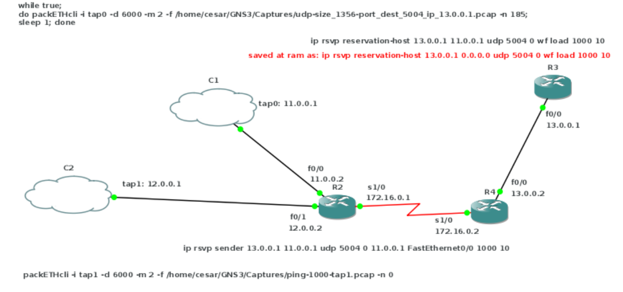
```
interface FastEthernet0/0
 ip address 13.0.0.2 255.255.255.0
 ip rsvp bandwidth 1200 1200
!
interface Serial1/0
 ip address 172.16.0.2 255.255.255.0
 ip rsvp bandwidth 1150 1150
```

Enabling RSVP at interfaces (R3)

```
interface FastEthernet0/0
 ip address 13.0.0.1 255.255.255.0
 ip rsvp bandwidth 1200 1200
```

Configuration and monitoring

Example 1



```
while true;
do packETHcli -i tap0 -d 6000 -m 2 -f /home/cesar/GNS3/Captures/udp-size_1356-port_dest_5004_ip_13.0.0.1.pcap -n 185;
sleep 1; done
```

```
ip rsvp reservation-host 13.0.0.1 11.0.0.1 udp 5004 0 wf load 1000 10
saved at ram as: ip rsvp reservation-host 13.0.0.1 0.0.0.0 udp 5004 0 wf load 1000 10
```

```
ip rsvp sender 13.0.0.1 11.0.0.1 udp 5004 0 11.0.0.1 FastEthernet0/0 1000 10
```

```
packETHcli -i tap1 -d 6000 -m 2 -f /home/cesar/GNS3/Captures/ping-1000-tap1.pcap -n 0
```

RSVP PATH proxy (R2)

For clients without RSVP capabilities

```
ip rsvp sender 13.0.0.1 11.0.0.1 UDP 5004 0 11.0.0.1 FastEthernet0/0 1000 10
```

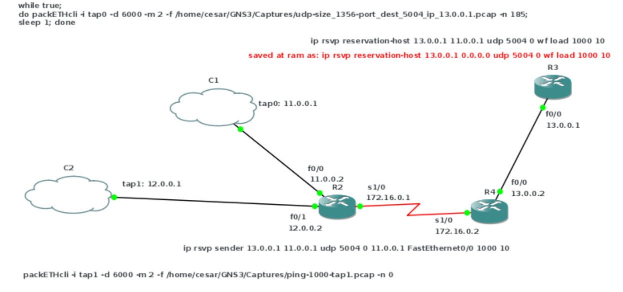
Prev. Hop

BW asked (kbps)

Max burst asked (KB)

Configuration and monitoring

Example 1



```
while true;
do packETHcli -i tap0 -d 6000 -m 2 -f /home/cesar/GNS3/Captures/udp-size_1356-port_dest_5004_ip_13.0.0.1.pcap -n 185;
sleep 1; done
```

```
ip rsvp reservation-host 13.0.0.1 11.0.0.1 udp 5004 0 wf load 1000 10
```

saved at ram as: ip rsvp reservation-host 13.0.0.1 0.0.0.0 udp 5004 0 wf load 1000 10

```
ip rsvp sender 13.0.0.1 11.0.0.1 udp 5004 0 11.0.0.1 FastEthernet0/0 1000 10
```

```
packETHcli -i tap1 -d 6000 -m 2 -f /home/cesar/GNS3/Captures/ping-1000-tap1.pcap -n 0
```

RSVP CONFIRM (R3)

```
ip rsvp reservation-host 13.0.0.1 11.0.0.1 UDP 5004 0 WF LOAD 1000 10
```

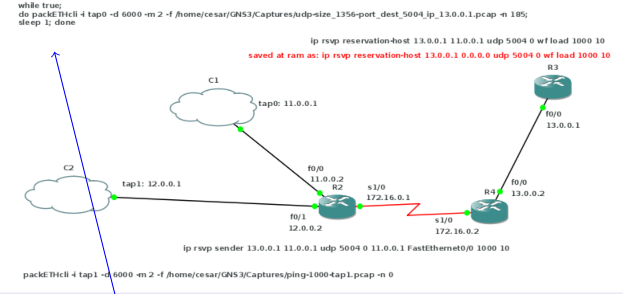
Reserv. style

BW reserved (kbps)

Max burst reserved (KB)

Configuration and monitoring

Example 1



```
while true;
do packETHcli -i tap0 -d 6000 -m 2 -f /home/cesar/GNS3/Captures/udp-size_1356-port_dest_5004_ip_13.0.0.1.pcap -n 185;
sleep 1; done
```

```
ip rsvp reservation-host 13.0.0.1 11.0.0.1 udp 5004 0 wf load 1000 10
```

saved at ram as: ip rsvp reservation-host 13.0.0.1 0.0.0.0 udp 5004 0 wf load 1000 10

```
ip rsvp sender 13.0.0.1 11.0.0.1 udp 5004 0 11.0.0.1 FastEthernet0/0 1000 10
```

```
packETHcli -i tap1 -d 6000 -m 2 -f /home/cesar/GNS3/Captures/ping-1000-tap1.pcap -n 0
```

Generate reserved traffic (`tap0`)

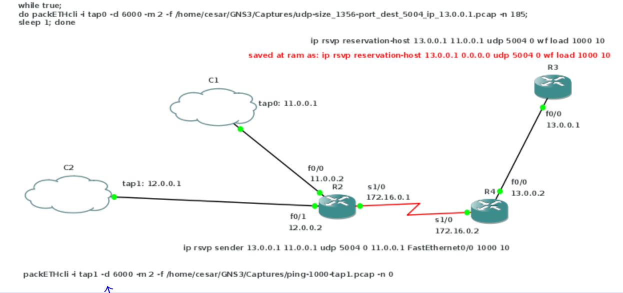
Rate: $\frac{1356 \cdot 8}{6000 \cdot 10^{-6}} = 1.8$ Mbps

Burst size: $185 \cdot 1356 = 250$ KB

Burst duration: $\frac{185 \cdot 1356 \cdot 8}{10^8} = 14.6$ ms

Configuration and monitoring

Example 1



```
while true;
do packETHcli -i tap0 -d 6000 -m 2 -f /home/cesar/GNS3/Captures/udp-size_1356-port_dest_5004_ip_13.0.0.1.pcap -n 185;
sleep 1; done
```

```
ip rsvp reservation-host 13.0.0.1 11.0.0.1 udp 5004 0 wf load 1000 10
```

saved at ram as: ip rsvp reservation-host 13.0.0.1 0.0.0.0 udp 5004 0 wf load 1000 10

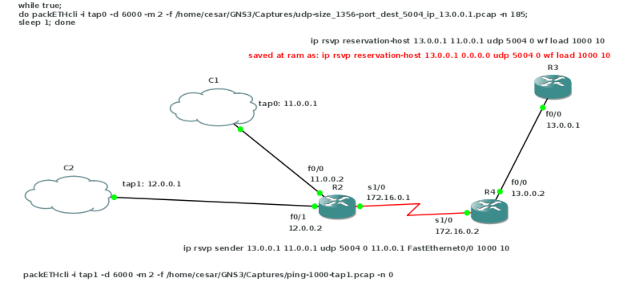
```
ip rsvp sender 13.0.0.1 11.0.0.1 udp 5004 0 11.0.0.1 FastEthernet0/0 1000 10
```

```
packETHcli -i tap1 -d 6000 -m 2 -f /home/cesar/GNS3/Captures/ping-1000-tap1.pcap -n 0
```

Generate non reserved traffic (`tap1`)
≃ 1.3 Mbps

Configuration and monitoring

Example 1



```
while true;
do packETHcli -i tap0 -d 6000 -m 2 -f /home/cesar/GNS3/Captures/udp-size_1356-port_dest_5004_ip_13.0.0.1.pcap -n 185;
sleep 1; done
```

```
ip rsvp reservation-host 13.0.0.1 11.0.0.1 udp 5004 0 wf load 1000 10
```

saved at ram as: ip rsvp reservation-host 13.0.0.1 0.0.0.0 udp 5004 0 wf load 1000 10

```
ip rsvp sender 13.0.0.1 11.0.0.1 udp 5004 0 11.0.0.1 FastEthernet0/0 1000 10
```

```
packETHcli -i tap1 -d 6000 -m 2 -f /home/cesar/GNS3/Captures/ping-1000-tap1.pcap -n 0
```

Monitoring reservations (R2 and R4)

```
R2# show ip rsvp reservation
To            From          Pro DPort Sport Next Hop      I/F      Fi Serv BPS
13.0.0.1      0.0.0.0       UDP 5004  0     172.16.0.2    Se1/0    WF LOAD 1M
```

Configuration and monitoring

Example 1. Monitoring queues (R2)

```
R2#show queueing interface Serial 1/0
Interface Serial1/0 queueing strategy: fair
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 18256
  Queueing strategy: weighted fair
  Output queue: 106/1000/64/18256 (size/max total/threshold/drops)
     Conversations  3/4/256 (active/max active/max total)
     Reserved Conversations 1/1 (allocated/max allocated)
     Available Bandwidth 158 kilobits/sec

  (depth/weight/total drops/no-buffer drops/interleaves) 9/6/0/0/0
  Conversation 265, linktype: ip, length: 1360
  source: 11.0.0.1, destination: 13.0.0.1, id: 0xEF7D, ttl: 63,
  TOS: 0 prot: 17, source port 48823, destination port 5004

  (depth/weight/total drops/no-buffer drops/interleaves) 42/32384/4304/0/0
  Conversation 63, linktype: ip, length: 1360
  source: 11.0.0.1, destination: 13.0.0.1, id: 0xEF7D, ttl: 63,
  TOS: 0 prot: 17, source port 48823, destination port 5004

  (depth/weight/total drops/no-buffer drops/interleaves) 55/32384/7187/0/0
  Conversation 28, linktype: ip, length: 1032
  source: 12.0.0.1, destination: 13.0.0.1, id: 0x39A8, ttl: 61, prot: 1
```

Reserved flow. Weight 6 in WFQ

Configuration and monitoring

Example 1. Monitoring queues (R2)

```
R2#show queueing interface Serial 1/0
Interface Serial1/0 queueing strategy: fair
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 18256
  Queueing strategy: weighted fair
  Output queue: 106/1000/64/18256 (size/max total/threshold/drops)
     Conversations  3/4/256 (active/max active/max total)
     Reserved Conversations 1/1 (allocated/max allocated)
     Available Bandwidth 158 kilobits/sec

  (depth/weight/total drops/no-buffer drops/interleaves) 9/6/0/0/0
  Conversation 265, linktype: ip, length: 1360
  source: 11.0.0.1, destination: 13.0.0.1, id: 0xEF7D, ttl: 63,
  TOS: 0 prot: 17, source port 48823, destination port 5004

  (depth/weight/total drops/no-buffer drops/interleaves) 42/32384/4304/0/0
  Conversation 63, linktype: ip, length: 1360
  source: 11.0.0.1, destination: 13.0.0.1, id: 0xEF7D, ttl: 63,
  TOS: 0 prot: 17, source port 48823, destination port 5004

  (depth/weight/total drops/no-buffer drops/interleaves) 55/32384/7187/0/0
  Conversation 28, linktype: ip, length: 1032
  source: 12.0.0.1, destination: 13.0.0.1, id: 0x39A8, ttl: 61, prot: 1
```

Not reserved flow. Weight 32,384 in WFQ. Best-effort

Configuration and monitoring

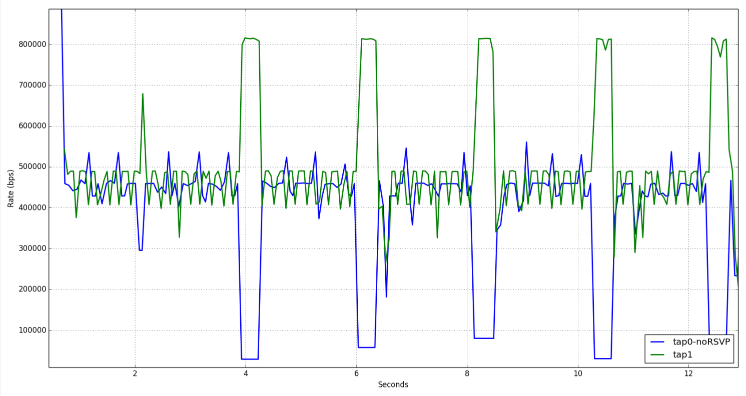
Example 1. Monitoring queues (R2)

```
R2#show queueing interface Serial 1/0
Interface Serial1/0 queueing strategy: fair
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 18256
  Queueing strategy: weighted fair
  Output queue: 106/1000/64/18256 (size/max total/threshold/drops)
     Conversations  3/4/256 (active/max active/max total)
     Reserved Conversations 1/1 (allocated/max allocated)
     Available Bandwidth 158 kilobits/sec

  (depth/weight/total drops/no-buffer drops/interleaves) 9/6/0/0/0
  Conversation 265, linktype: ip, length: 1360
  source: 11.0.0.1, destination: 13.0.0.1, id: 0xEF7D, ttl: 63,
  TOS: 0 prot: 17, source port 48823, destination port 5004

  (depth/weight/total drops/no-buffer drops/interleaves) 42/32384/4304/0/0
  Conversation 63, linktype: ip, length: 1360
  source: 11.0.0.1, destination: 13.0.0.1, id: 0xEF7D, ttl: 63,
  TOS: 0 prot: 17, source port 48823, destination port 5004

  (depth/weight/total drops/no-buffer drops/interleaves) 55/32384/7187/0/0
  Conversation 28, linktype: ip, length: 1032
  source: 12.0.0.1, destination: 13.0.0.1, id: 0x39A8, ttl: 61, prot: 1
```

Non compliant (token bucket) part from reserved flow. Weight 32,384 in WFQ. Best-effort

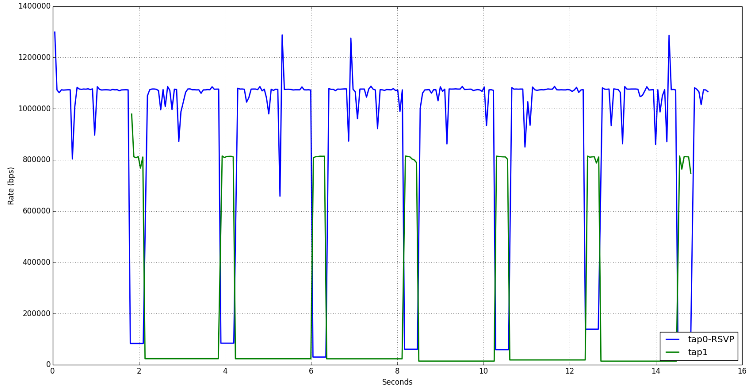
Configuration and monitoring

Example 1. Throughput measurements without RSVP



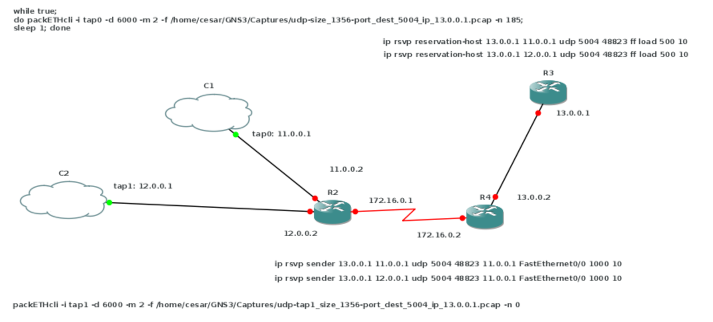
Configuration and monitoring

Example 1. Throughput measurements with RSVP



Configuration and monitoring

Example 2. FF style reservations



```
while true;
do packETHcli -i tap0 -d 6000 -m 2 -f /home/cesar/GNS3/Captures/udp-size_1356-port_dest_5004_ip_13.0.0.1.pcap -n 185;
sleep 1; done
```

```
ip rsvp reservation-host 13.0.0.1 11.0.0.1 udp 5004 48823 ff load 500 10
ip rsvp reservation-host 13.0.0.1 12.0.0.1 udp 5004 48823 ff load 500 10
```

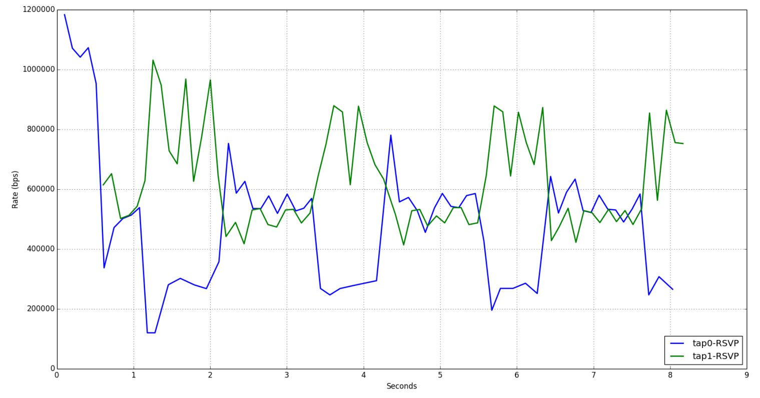
```
ip rsvp sender 13.0.0.1 11.0.0.1 udp 5004 48823 11.0.0.1 FastEthernet0/0 1000 10
ip rsvp sender 13.0.0.1 12.0.0.1 udp 5004 48823 11.0.0.1 FastEthernet0/0 1000 10
```

```
packETHcli -i tap1 -d 6000 -m 2 -f /home/cesar/GNS3/Captures/udp-tap1_size_1356-port_dest_5004_ip_13.0.0.1.pcap -n 0
```

- Two FF reservations (source port must be included)
- 1000 Kbps asked for and 500 Kbps allowed

Configuration and monitoring

Example 2. FF style reservations. Throughput measurements



Configuration and monitoring

Reservation styles by example

Ask for BW from R2

```
R2(config)# ip rsvp sender 13.0.0.1 11.0.0.1 UDP 5004 48823 11.0.0.1 FastEthernet0/0 1000 10
R2(config)# ip rsvp sender 13.0.0.1 12.0.0.1 UDP 5004 48823 12.0.0.1 FastEthernet0/1 1000 10
```

FF BW allocation from R3

```
R3(config)# ip rsvp reservation-host 13.0.0.1 12.0.0.1 UDP 5004 48823 FF LOAD 500 10
R3(config)# ip rsvp reservation-host 13.0.0.1 11.0.0.1 UDP 5004 48823 FF LOAD 500 10
```

Reserved resources at R2

```
R2# show ip rsvp reservation
To            From          Pro DPort Sport Next Hop      I/F       Fi Serv BPS
13.0.0.1      11.0.0.1      UDP 5004  48823 172.16.0.2    Se1/0     FF LOAD 500K
13.0.0.1      12.0.0.1      UDP 5004  48823 172.16.0.2    Se1/0     FF LOAD 500K
```

Configuration and monitoring

Reservation styles by example

Reserve more than available

```
R3(config)# no ip rsvp reservation-host 13.0.0.1 12.0.0.1 UDP 5004 48823 FF LOAD 500 10
R3(config)# no ip rsvp reservation-host 13.0.0.1 11.0.0.1 UDP 5004 48823 FF LOAD 500 10

R3(config)# ip rsvp reservation-host 13.0.0.1 11.0.0.1 UDP 5004 48823 FF LOAD 800 10

R2# show ip rsvp reservation
To            From          Pro DPort Sport Next Hop      I/F      Fi Serv BPS
13.0.0.1      11.0.0.1      UDP 5004  48823 172.16.0.2    Se1/0    FF LOAD 800K

R3(config)# ip rsvp reservation-host 13.0.0.1 12.0.0.1 UDP 5004 48823 FF LOAD 800 10

R2# show ip rsvp reservation
To            From          Pro DPort Sport Next Hop      I/F      Fi Serv BPS
13.0.0.1      11.0.0.1      UDP 5004  48823 172.16.0.2    Se1/0    FF LOAD 800K
```

An error message returned from R4 telling that not enough BW

Configuration and monitoring

Reservation styles by example

SE style reservations

```
R3(config)# no ip rsvp reservation-host 13.0.0.1 11.0.0.1 UDP 5004 48823 FF LOAD 800 10
R3(config)# no ip rsvp reservation-host 13.0.0.1 12.0.0.1 UDP 5004 48823 FF LOAD 800 10

R3(config)# ip reservation-host 13.0.0.1 11.0.0.1 UDP 5004 48823 SE LOAD 500 10
R3(config)# ip reservation-host 13.0.0.1 12.0.0.1 UDP 5004 48823 SE LOAD 800 10

R2# show ip rsvp reservation
To            From          Pro DPort Sport Next Hop      I/F      Fi Serv BPS
13.0.0.1      11.0.0.1      UDP 5004  48823 172.16.0.2    Se1/0    SE LOAD 500K
13.0.0.1      12.0.0.1      UDP 5004  48823 172.16.0.2    Se1/0    SE LOAD 800K
```

Maximum shared (800 Kbps) for both sessions

Configuration and monitoring

Reservation styles by example

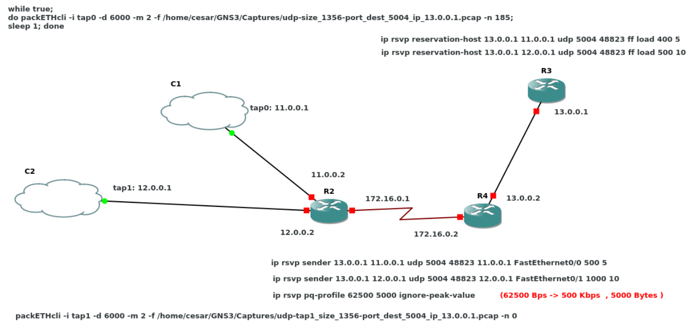
WF style reservations

```
R3(config)# no ip reservation-host 13.0.0.1 11.0.0.1 UDP 5004 48823 SE LOAD 500 10
R3(config)# no ip reservation-host 13.0.0.1 12.0.0.1 UDP 5004 48823 SE LOAD 800 10

R3(config)# ip reservation-host 13.0.0.1 11.0.0.1 UDP 5004 48823 WF LOAD 500 10
R3(config)# ip reservation-host 13.0.0.1 12.0.0.1 UDP 5004 48823 WF LOAD 800 10

R2# show ip rsvp reservation
To            From          Pro DPort Sport Next Hop      I/F      Fi Serv BPS
13.0.0.1      0.0.0.0       UDP 5004  0     172.16.0.2    Se1/0    WF LOAD 800K
```

Wildcards (IP 0.0.0.0 and port 0) for any source. Maximum reservation used (800 Kbps)

Configuration and monitoring

Example 3. Guaranteed-rate. Using PQ



```
while true;
do packETHcli -i tap0 -d 6000 -m 2 -f /home/cesar/GNS3/Captures/udp-size_1356-port_dest_5004_ip_13.0.0.1.pcap -n 185;
sleep 1; done
```

```
ip rsvp reservation-host 13.0.0.1 11.0.0.1 udp 5004 48823 ff load 400 5
ip rsvp reservation-host 13.0.0.1 12.0.0.1 udp 5004 48823 ff load 500 10
```

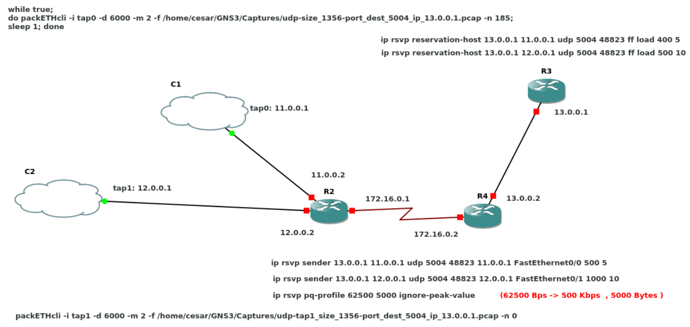
```
ip rsvp sender 13.0.0.1 11.0.0.1 udp 5004 48823 11.0.0.1 FastEthernet0/0 500 5
ip rsvp sender 13.0.0.1 12.0.0.1 udp 5004 48823 12.0.0.1 FastEthernet0/1 1000 10
ip rsvp pq-profile 62500 5000 ignore-peak-value        (62500 Bps -> 500 Kbps  , 5000 Bytes )
```

```
packETHcli -i tap1 -d 6000 -m 2 -f /home/cesar/GNS3/Captures/udp-tap1_size_1356-port_dest_5004_ip_13.0.0.1.pcap -n 0
```

- Traffic from `tap0` considered priority. Reserved rate (*r*) 400 Kbps, burst size (*b*) 5 KB
- Traffic from `tap1`. Reserved 500 Kbps, burst size 10 KB

Configuration and monitoring

Example 3. Guaranteed-rate. Using PQ



```
while true;
do packETHcli -i tap0 -d 6000 -m 2 -f /home/cesar/GNS3/Captures/udp-size_1356-port_dest_5004_ip_13.0.0.1.pcap -n 185;
sleep 1; done
```

```
ip rsvp reservation-host 13.0.0.1 11.0.0.1 udp 5004 48823 ff load 400 5
ip rsvp reservation-host 13.0.0.1 12.0.0.1 udp 5004 48823 ff load 500 10
```

```
ip rsvp sender 13.0.0.1 11.0.0.1 udp 5004 48823 11.0.0.1 FastEthernet0/0 500 5
ip rsvp sender 13.0.0.1 12.0.0.1 udp 5004 48823 12.0.0.1 FastEthernet0/1 1000 10
ip rsvp pq-profile 62500 5000 ignore-peak-value
```

(62500 Bps -> 500 Kbps , 5000 Bytes)

```
packETHcli -i tap1 -d 6000 -m 2 -f /home/cesar/GNS3/Captures/udp-tap1_size_1356-port_dest_5004_ip_13.0.0.1.pcap -n 0
```

R2 configuration for PQ

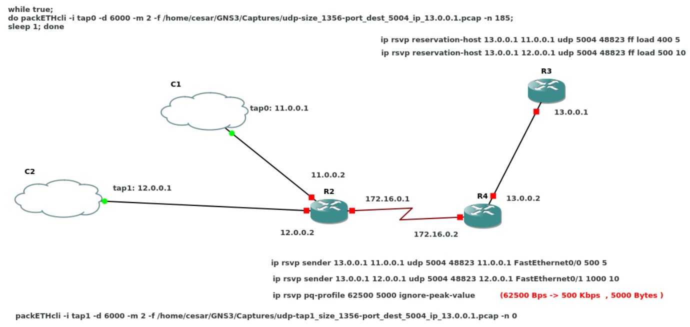
```
ip rsvp pq-profile 62500 5000 ignore-peak-value
```

Max. rate (r') in Bps
62500 Bps = 500 Kbps

Max. burst (b') in Bytes
5000 B = 5 KB

Configuration and monitoring

Example 3. Guaranteed-rate. Using PQ



```
while true;
do packETHcli -i tap0 -d 6000 -m 2 -f /home/cesar/GNS3/Captures/udp-size_1356-port_dest_5004_ip_13.0.0.1.pcap -n 185;
sleep 1; done
```

```
ip rsvp reservation-host 13.0.0.1 11.0.0.1 udp 5004 48823 ff load 400 5
ip rsvp reservation-host 13.0.0.1 12.0.0.1 udp 5004 48823 ff load 500 10
```

```
ip rsvp sender 13.0.0.1 11.0.0.1 udp 5004 48823 11.0.0.1 FastEthernet0/0 500 5
ip rsvp sender 13.0.0.1 12.0.0.1 udp 5004 48823 12.0.0.1 FastEthernet0/1 1000 10
ip rsvp pq-profile 62500 5000 ignore-peak-value
```

(62500 Bps -> 500 Kbps , 5000 Bytes)

```
packETHcli -i tap1 -d 6000 -m 2 -f /home/cesar/GNS3/Captures/udp-tap1_size_1356-port_dest_5004_ip_13.0.0.1.pcap -n 0
```

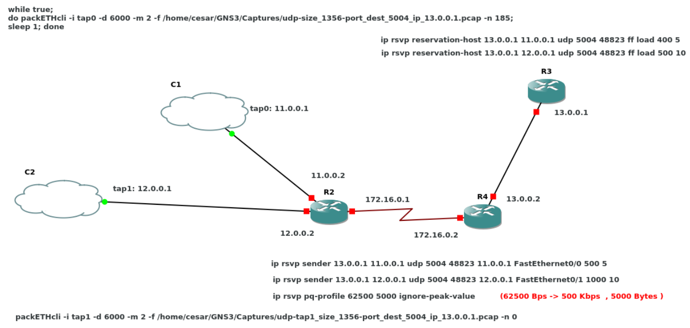
Every reservation such that:

$$r \leq r' \text{ and } b \leq b'$$

will be considered into PQ

Configuration and monitoring

Example 3. Guaranteed-rate. Using PQ



```
while true;
do packETHcli -i tap0 -d 6000 -m 2 -f /home/cesar/GNS3/Captures/udp-size_1356-port_dest_5004_ip_13.0.0.1.pcap -n 185;
sleep 1; done
```

```
ip rsvp reservation-host 13.0.0.1 11.0.0.1 udp 5004 48823 ff load 400 5
ip rsvp reservation-host 13.0.0.1 12.0.0.1 udp 5004 48823 ff load 500 10
```

```
ip rsvp sender 13.0.0.1 11.0.0.1 udp 5004 48823 11.0.0.1 FastEthernet0/0 500 5
ip rsvp sender 13.0.0.1 12.0.0.1 udp 5004 48823 12.0.0.1 FastEthernet0/1 1000 10
ip rsvp pq-profile 62500 5000 ignore-peak-value
```

(62500 Bps -> 500 Kbps , 5000 Bytes)

```
packETHcli -i tap1 -d 6000 -m 2 -f /home/cesar/GNS3/Captures/udp-tap1_size_1356-port_dest_5004_ip_13.0.0.1.pcap -n 0
```

Monitoring reservations

```
R2#show ip rsvp  installed
RSVP: FastEthernet0/0 has no installed reservations
RSVP: Serial1/0
BPS    To              From            Protoc DPort  Sport  Weight Conversation
400K   13.0.0.1        11.0.0.1        UDP    5004   48823  0      264
500K   13.0.0.1        12.0.0.1        UDP    5004   48823  6      265
RSVP: FastEthernet0/1 has no installed reservations
```

Contents

Bibliography

- *Network Warrior 2nd Ed.* Gary A. Donahue. O'Reilly, 2011
- Cisco IOS Quality of Service Solutions Configuration Guide, Release 12.2SR
- Cisco. Quality of Service Networking
- Cisco IOS Quality of Service Solutions Command Reference
- *Administering CISCO IP QoS in IP Networks* . Syngress, 2001
- *Internet QoS: Architectures and Mechanisms for Quality of Service*. Zheng Wang. Morgan Kaufmann Publishers, 2001