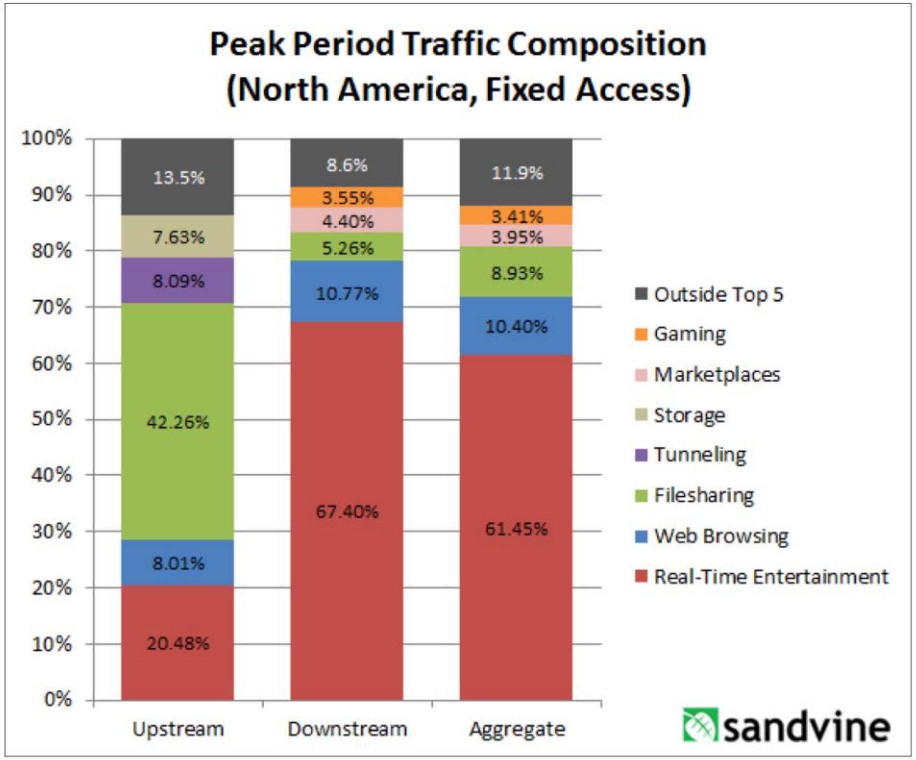
# Infrastructure QoS support

Carlo Vallati
Assistant Professor@ University of Pisa
c.vallati@iet.unipi.it

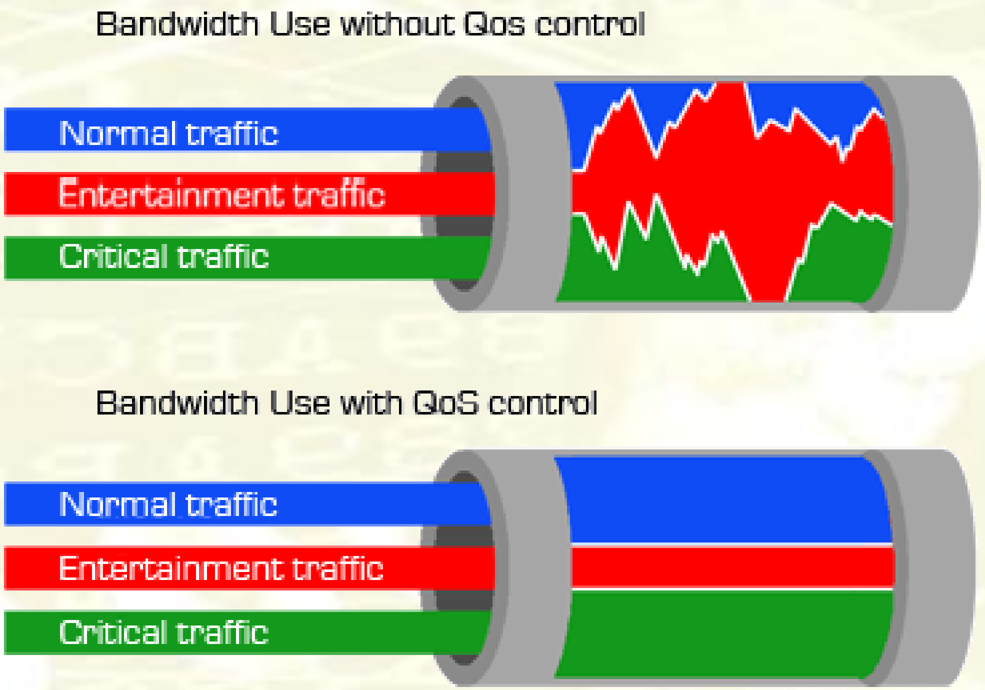



| | Upstream | | Downstream | | Aggregate | |
|---|---|---|---|---|---|---|
| **Rank** | **Application** | **Share** | **Application** | **Share** | **Application** | **Share** |
| 1 | BitTorrent | 36.35% | Netflix | 31.62% | Netflix | 28.18% |
| 2 | HTTP | 6.03% | YouTube | 18.69% | YouTube | 16.78% |
| 3 | SSL | 5.87% | HTTP | 9.74% | HTTP | 9.26% |
| 4 | Netflix | 4.44% | BitTorrent | 4.05% | BitTorrent | 7.39% |
| 5 | YouTube | 3.63% | iTunes | 3.27% | iTunes | 2.91% |
| 6 | Skype | 2.76% | MPEG - Other | 2.60% | SSL | 2.54% |
| 7 | QVoD | 2.55% | SSL | 2.05% | MPEG - Other | 2.32% |
| 8 | Facebook | 1.54% | Amazon Video | 1.61% | Amazon Video | 1.48% |
| 9 | FaceTime | 1.44% | Facebook | 1.31% | Facebook | 1.34% |
| 10 | Dropbox | 1.39% | Hulu | 1.29% | Hulu | 1.15% |
| | **Top 10** | **66.00%** | **Top 10** | **76.23%** | **Top 10** | **73.35%** |

SOURCE: SANDVINE NETWORK DEMOGRAPHICS

sandvine

# QoS in the Internet

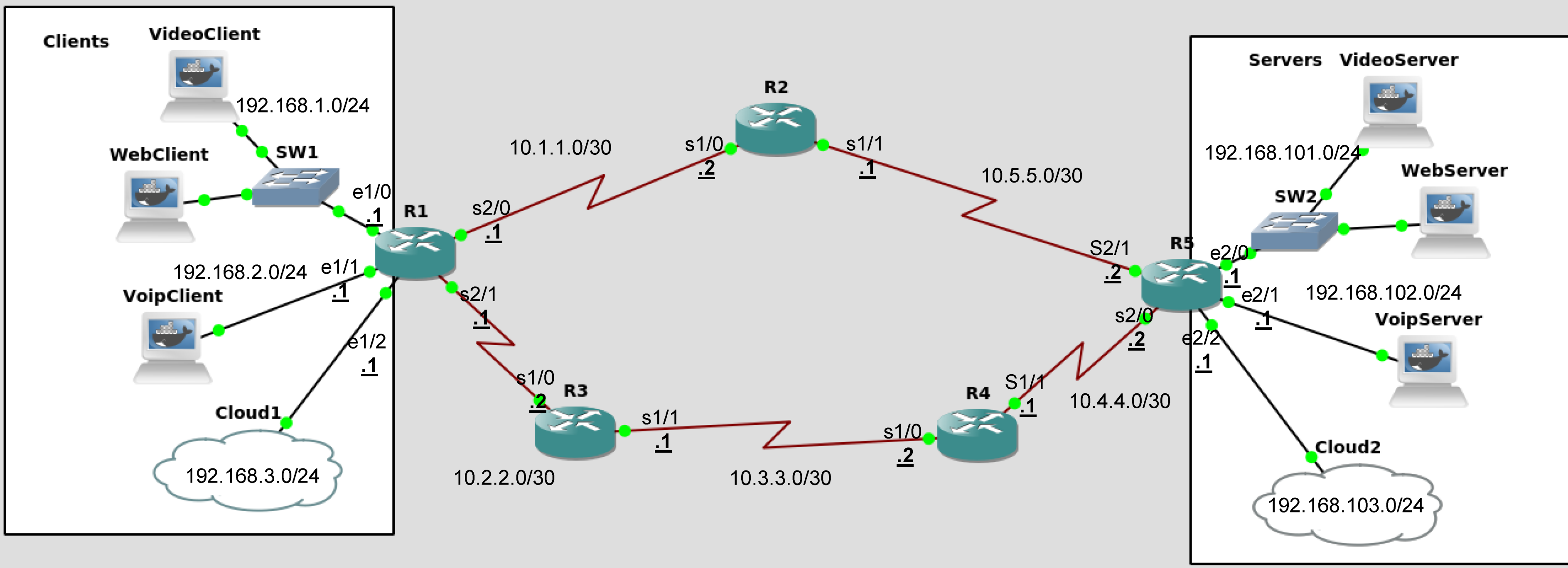



E.g. VoIP requires packets to be delivered with a delay which is in the worst case in the order of 150ms one-way

# Basic Network

Carlo Vallati
Assistant Professor@ University of Pisa
c.vallati@iet.unipi.it

# Network Architecture

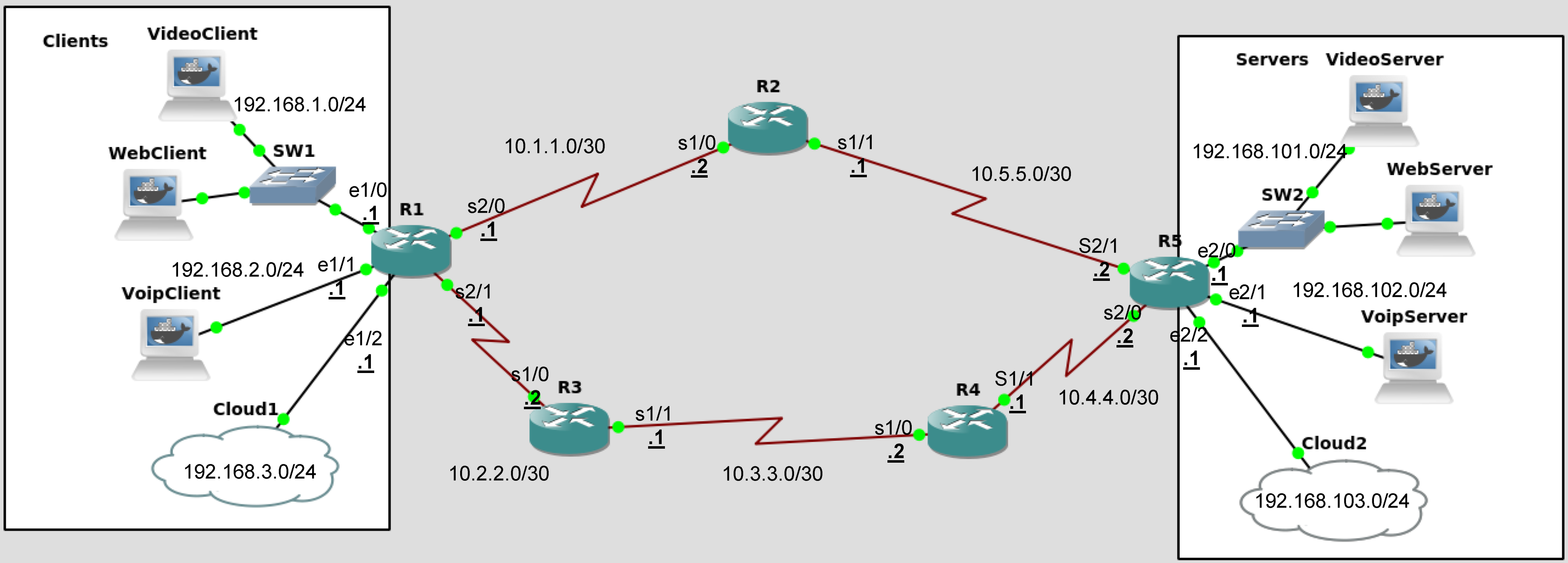



Linux OS
tap0

***tapX*** is a virtual network interface exposed on the Linux OS to provide a point of access to the emulated network

Linux OS
tap1

# Network Architecture




Linux OS tap0

Each tap is attached to an ***isolated Network Namespace*** with its own network stack and routing table. Applications running on each namespace are forced to ***communicate through*** the emulated network

Linux OS tap1

# Differentiated Services Architectures

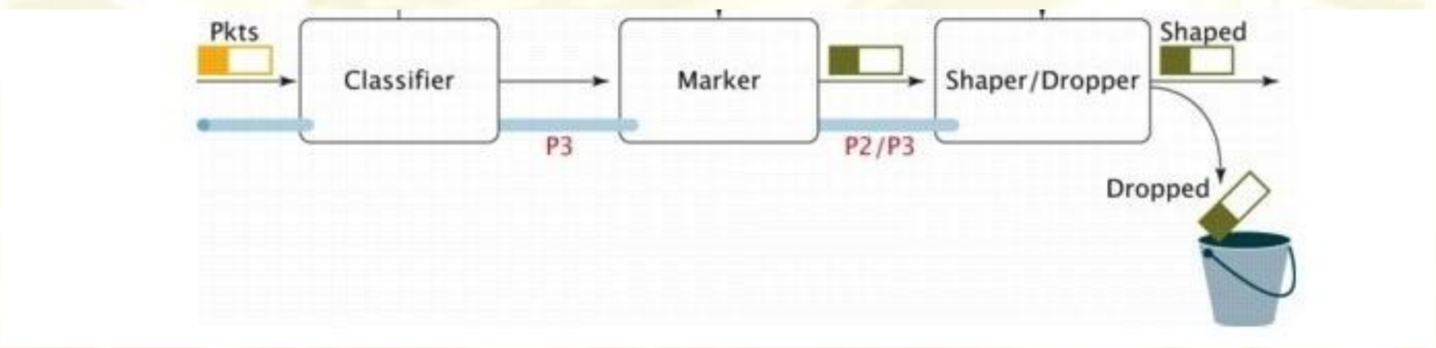
Carlo Vallati
Assistant Professor@ University of Pisa
c.vallati@iet.unipi.it

# Intro

- Network Boundary
  - Classification & Marking
  - Shaping and Policing
- Per-Hop Behavior
  - Scheduling and Resource Allocation
  - Congestion Avoidance and Packet Drop Policy

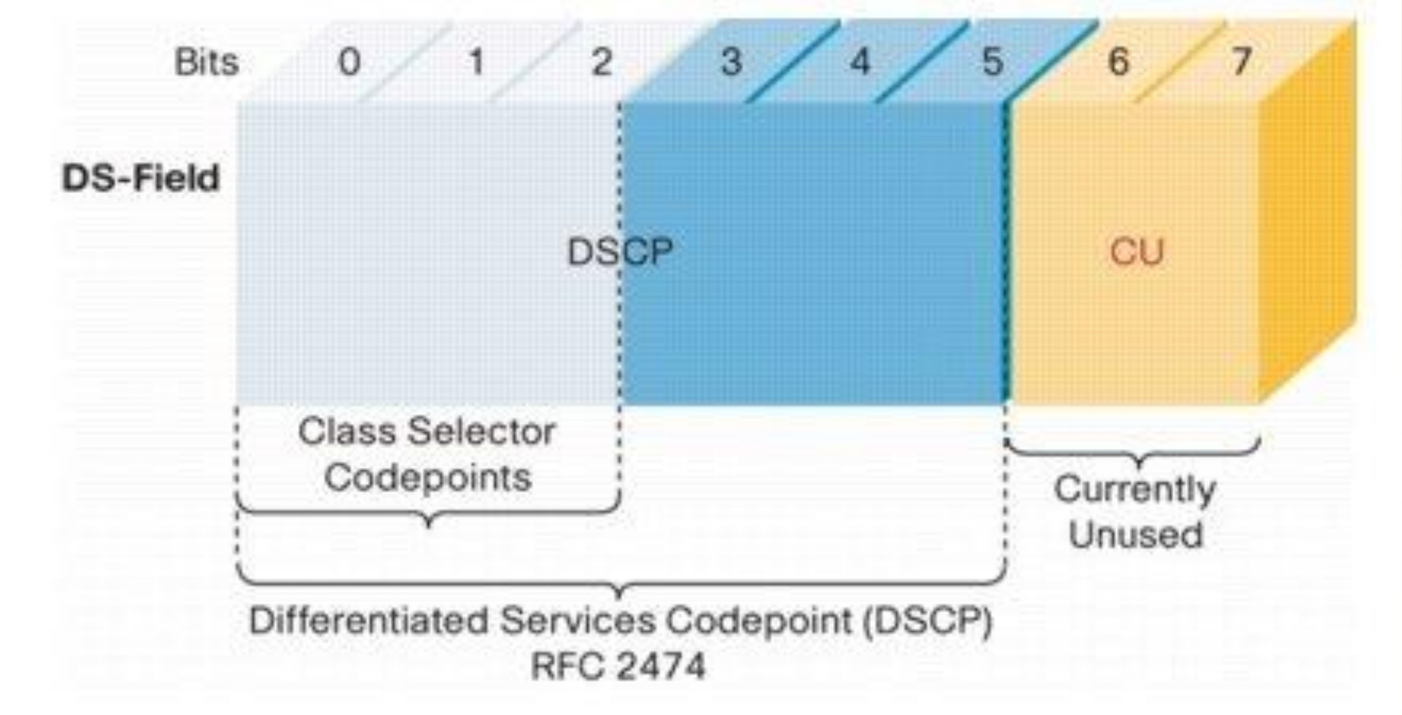
# Network Ingress

- Routers at the network boundary perform ***Classifier*** functions to identify packets belonging to a certain traffic class based on one or more TCP/IP header fields
- A ***Marker*** is used to color the classified traffic
- A ***Shaper*** or ***Traffic Policing*** is used to regulate ingress rate



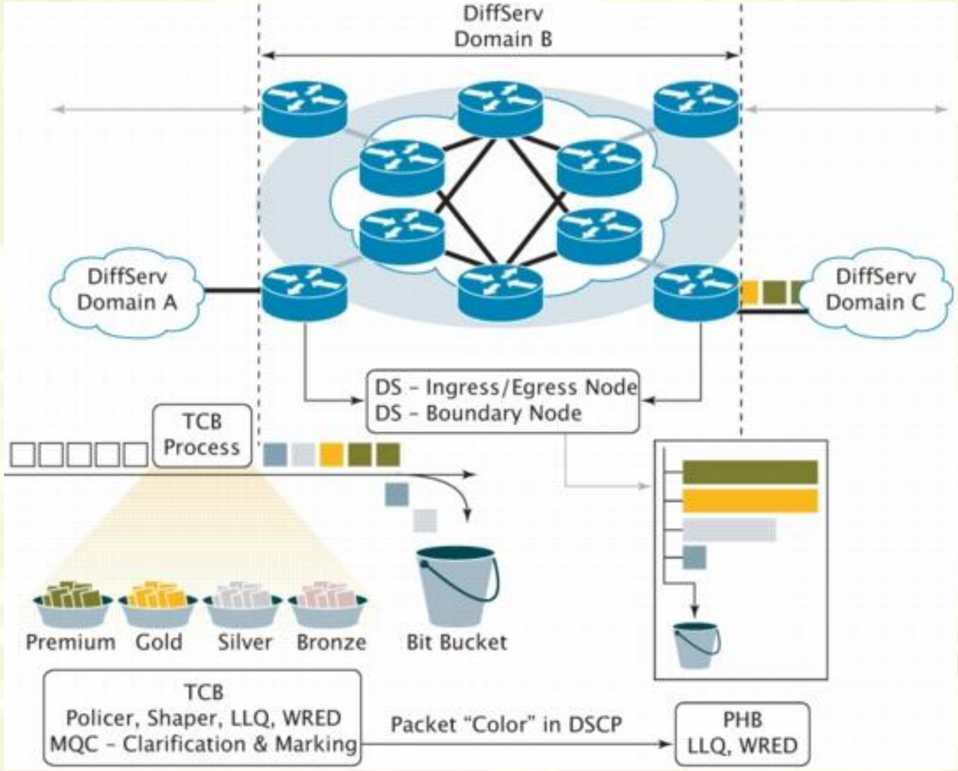
# Packet Marker

- The marker set the Differentiated Services Code Point (DSCP) field



# Per-Hop Behavior

- Within the network core, a per-hop behavior (PHB) is applied to the packets based on either the IP Precedence or the DSCP field marked in the packet header



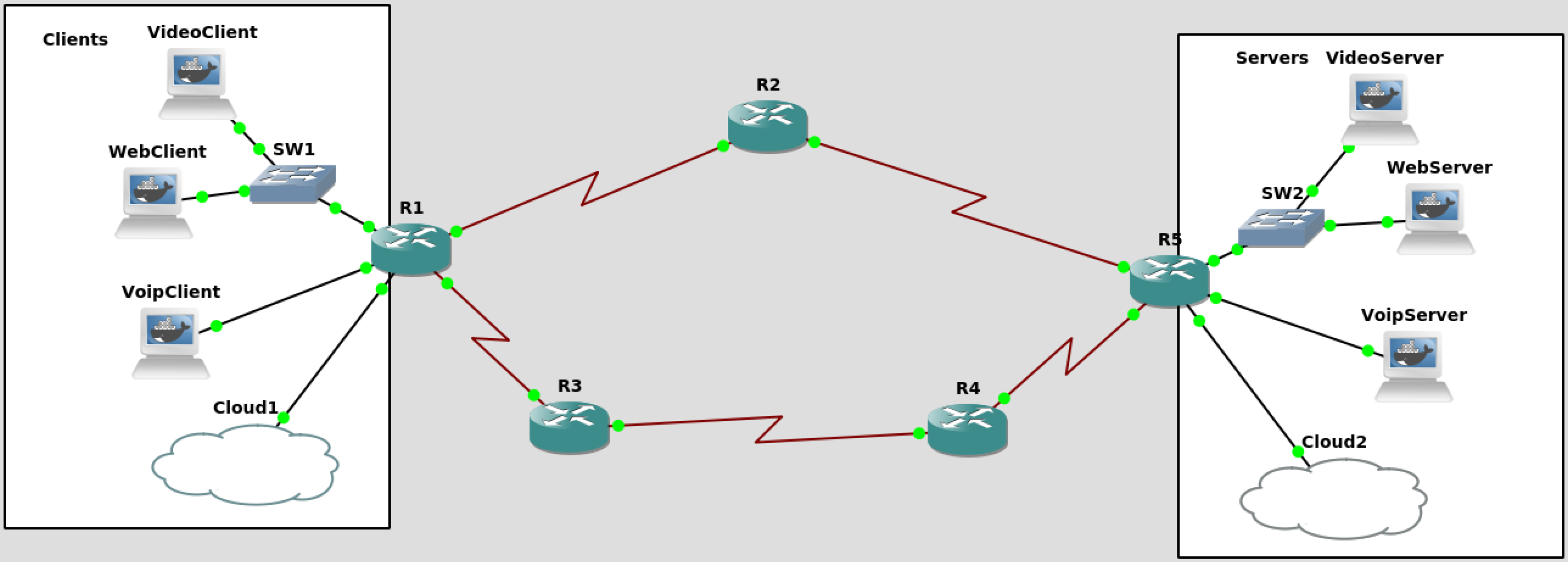
# Resulting Service

| QoS Class Names | Layer 3 QoS Markings | | IPP / CoS Markings |
|---|---|---|---|
| | PHB | DSCP | |
| Network Control | CS6 | 48 | 6 |
| Voice Real-Time Transport | EF | 46 | 5 |
| Clinical Life Critical | CS5 | 40 | 5 |
| Multimedia Conferencing | AF41 | 34 | 4 |
| Real-Time Interactive | CS4 | 32 | 4 |
| Multimedia Streaming | AF31 | 26 | 3 |
| Call Signaling | CS3 | 24 | 3 |
| Low-Latency Data | AF21 | 18 | 2 |
| OAM (Net Mgmt) | CS2 | 16 | 2 |
| High-Throughput Data | AF11 | 10 | 1 |
| Low-Priority Data | CS1 | 8 | 1 |
| Best Effort | 0 | 0 | 0 |

# Network Boundary

Carlo Vallati
Assistant Professor@ University of Pisa
c.vallati@iet.unipi.it

# Application Scenario




3 Flows by priority:

1. VoIP Flow (UDP): LOW Latency, LOW Loss
2. Video Flow (UDP): MEDIUM Latency, LOW Loss
3. Web Flow (TCP): Delay and Loss tolerant

# Packet Classification and Marking using Class Maps

- Define Classification
  - `access-list 1 permit 192.168.102.0 0.0.0.255`
  - `class-map match-all VOIP`
  - ` match access-group 1`

Classify all the traffic from the network 192.168.102.0/24

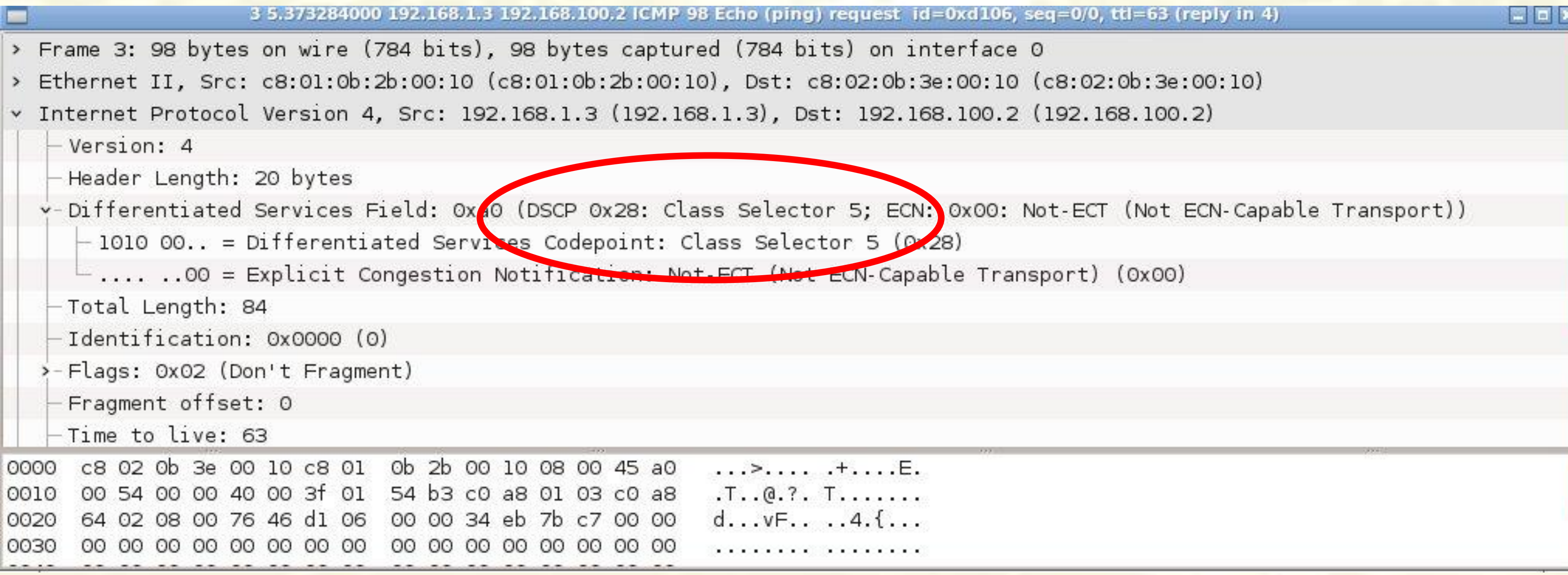
- Define Marking
  - `policy-map E11`
  - ` class VOIP`
  - `  set ip dscp ef`
  - `interface Ethernet 1/1`
  - ` service-policy input E11`

Set DSCP field for this traffic as expedited forwarding

- To disable
  - `no service-policy input E11`

# Test

- Test with ping and check that the precedence bit on the IP header is set correctly using wireshark!



# Classification based on protocol

- Classify traffic based on transport protocol or application (e.g. telemetry traffic on UDP):
  - `access-list 101 permit udp any any`
  - `class-map match-all VIDEO`
  - `  match access-group 101`
- Define Marking:
  - `policy-map E10`
  - `  class VIDEO`
  - `   set ip dscp af13`

This add to the class VIDEO

Set DSCP field for this traffic as assured forwarding

# Classification based on protocol

- Apply marking (marking is already applied in Ethernet 1/0):
  - `interface Ethernet 1/0`
  - `service-policy input E10`

# Test with iperf

- Iperf is a tool to generate traffic
- Generate TCP flow (data is generated from client to server)
  - `iperf -s (to create the server)`
  - `iperf -c 192.168.1.2 (to start the flow)`
- Generate UDP flow (data is generated from client to server)
  - `iperf -s -u (to create the server)`
  - `iperf -u -c 192.168.2.2 -b 2M (to start 2Mbps flow, by default the flow is 1Mbps)`

# Test!

- Start two flows, one UDP and one TCP, the first between the VideoServer and the VideoClient, the other between the WebServer and the WebClient
  - Check with wireshark the DSCP value

# Classification based on port

- Classify traffic based on the specific service (e.g. web traffic on TCP port 80):
  - `access-list 102 permit tcp any any range www 81`
  - `class-map match-all WEB`
  - ` match access-group 102`
- Define Marking:
  - `policy-map E10`
  - ` class WEB`
  - `  set ip dscp af33`

This add to the class WEB

Set DSCP field for this traffic as assured forwarding

# Test!

- Test again with two flows, one UDP and one TCP, the first between the VideoServer and the VideoClient, the other between the WebServer and the WebClient
  - Check with wireshark the DSCP value

# Policing/Shaping

| Table 3-3. Comparison Between Policing and Shaping Functions | |
|---|---|
| **Policing Function (CAR)** | **Shaping Function (TS)** |
| Sends conforming traffic up to the line rate and allows bursts. | Smoothes traffic and sends it out at a constant rate. |
| When tokens are exhausted, it can drop packets. | When tokens are exhausted, it buffers packets and sends them out later, when tokens are available. |
| Works for both input and output traffic. | Implemented for output traffic only. |
| Transmission Control Protocol (TCP) detects the line at line speed but adapts to the configured rate when a packet drop occurs by lowering its window size. | TCP can detect that it has a lower speed line and adapt its retransmission timer accordingly. This results in less scope of retransmissions and is TCP-friendly. |

To limit ingress traffic rate

Smooth traffic flow on an interface to avoid link congestion

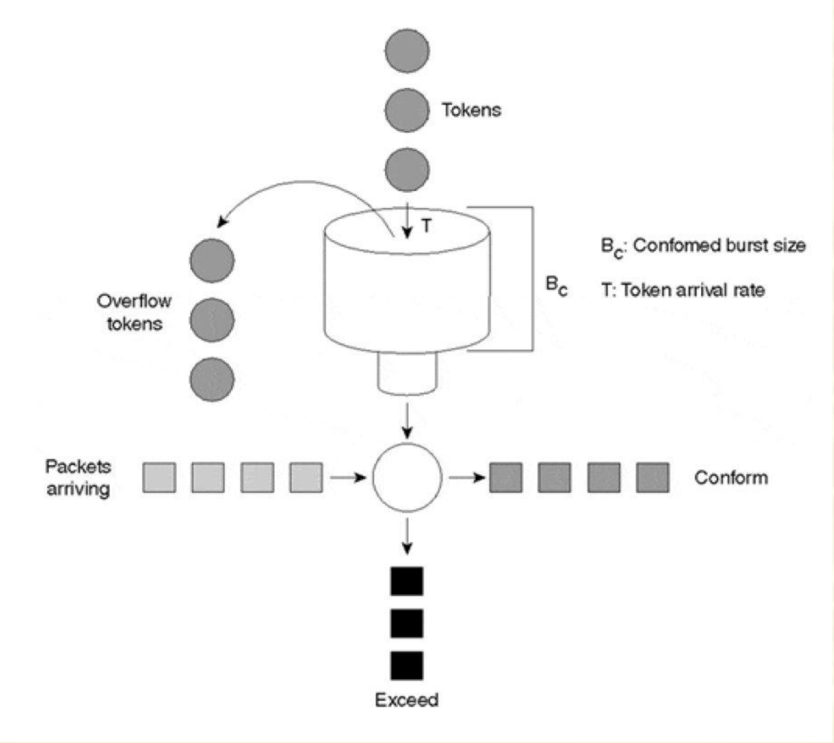
# CAR (Committed Access Rate)

- CAR is a traffic Classifier/Marker/Policing
- Usually adopted at the ingress router to limit ingress traffic of a flow
- It performs Traffic Policing with additional Classification & Marking, if needed

# CAR (Committed Access Rate)

- Rate limit statement:
  - `rate-limit {access-group num} <input/output> "CIR" "conformed burst" "extended burst" "conformed-action" "action desired" exceed-action "action desired"`

<u>CIR, Committed Information Rate</u>, *bit per second*, average traffic rate
<u>Conformed burst size</u>, *bytes*, amount of traffic allowed to exceed the bucket on an instantaneous basis
<u>Extended burst size</u>, *bytes*, bonus instantaneous rate based on token borrowing mechanism, if set equal to conformed burst size is disabled



# Packet C/M/P using CAR

- Define policing function:
  - `interface Ethernet 1/1`
  - `rate-limit input 10000000 5000 5000 conform-action continue exceed-action drop`
    - *Limit all the traffic to 10Mpbs (continue -> check other rules)*

# Packet C/M/P using CAR

- Use a classifier
  - `access-list 101 permit udp any any`
    - *Policing only udp traffic to limit the VIDEO traffic*
- Define policing function:
  - `interface Ethernet 1/0`
  - `rate-limit input access-group 101 1000000 2000 2000 conform-action transmit exceed-action set-dscp-transmit 38`
    - *Limit the VIDEO traffic 1Mbps, remark exceeded traffic*

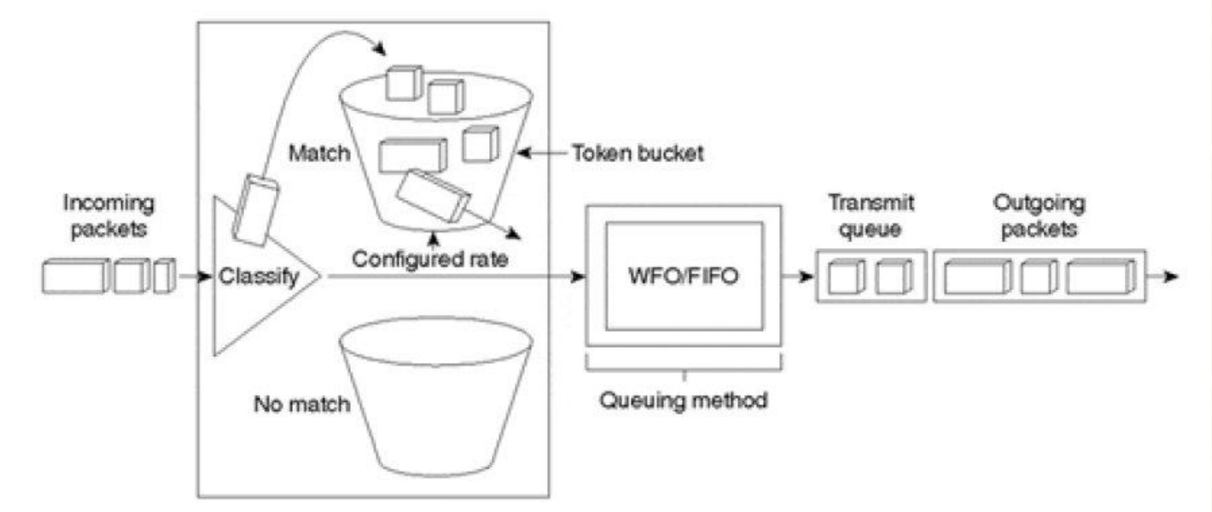
# Packet C/M/P using CAR

- Show status
  - `show interfaces Ethernet 1/1 rate`

```
Ethernet0/0
  Input
    matches: access-group 101
      params:  1000000 bps, 200000 limit, 200000 extended limit
      conformed 2039 packets, 2885550 bytes; action: set-prec-transmit 2
      exceeded 393 packets, 518162 bytes; action: drop
      last packet: 358299ms ago, current burst: 199897 bytes
      last cleared 00:06:39 ago, conformed 57000 bps, exceeded 10000 bps
R1#
```

# Traffic Shaping (TS)

- TS smoothes bursty traffic to meet the configured CIR by queuing or buffering packets exceeding the mean rate (*outbound traffic only*)
- Queued packets are transmitted as tokens become available (*Packets are not discarded*)
- Shaper is usually used at the core to avoid link congestion



# Generic Traffic Shaping

- I want to shape outgoing traffic on a certain interface (e.g. to avoid link saturation)
- Traffic class for shaping
  - `class-map match-all OUT_E00`
  - `match any`
- Add shaper as policy-map
  - `policy-map SHAPER`
  - `class OUT_E00`
  - `shape average 2000000`

# Generic Traffic Shaping

- Apply the policy-map to Ethernet 0/0
  - `interface Ethernet0/0`
  - `service-policy output SHAPER`

- Check the status of the policy-map
  - `show policy-map interface e0/0`

# Test!

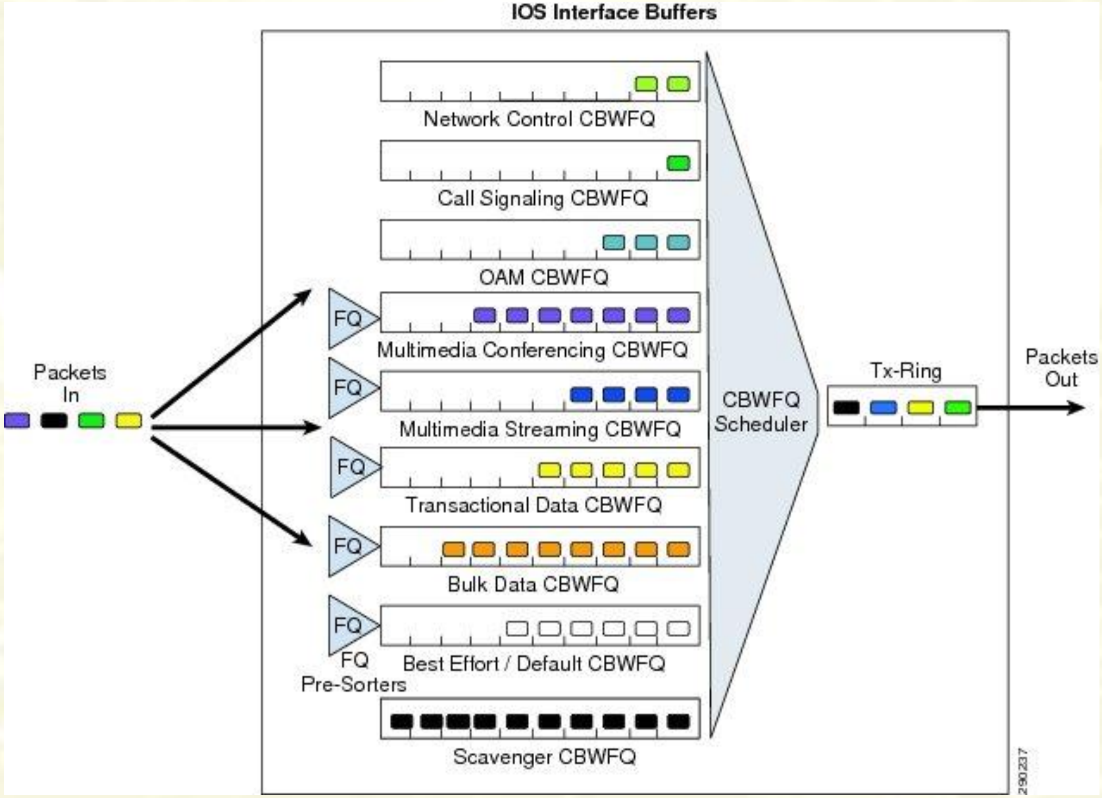
- Add a shaper to the outgoing interface e0/0 of R5 and test it with iperf using TCP traffic

# Per Hop Behavior

Carlo Vallati
Assistant Professor@ University of Pisa
c.vallati@iet.unipi.it

# Class-Based WFQ

- Traffic Class are defined
- CBWFQ allocates a different subqueue for each traffic class



# CBWFQ

- Classification based on DSCP
  - `class-map match-all VOIP`
  - `match dscp ef`
  - `class-map match-all EXCESS`
  - `match dscp af43`
  - `class-map match-all WEB`
  - `match dscp af33`
  - `class-map match-all VIDEO`
  - `match dscp af13`

Classify traffic based on the DSCP value

# CBWFQ

- Allocate bandwidth
  - policy-map OUT
  - class VOIP
  - bandwidth percent 30
  - queue-limit 100 packets
  - class VIDEO
  - bandwidth percent 60
  - class WEB
  - bandwidth percent 5
  - queue-limit 20 packets
  - class EXCESS
  - bandwidth percent 4

Set the bandwidth share for this traffic

Limit the number of packets that can be enqueued for that class

# CBWFQ

- Apply the configuration
  - `interface Serial1/0`
  - `service-policy output OUT`

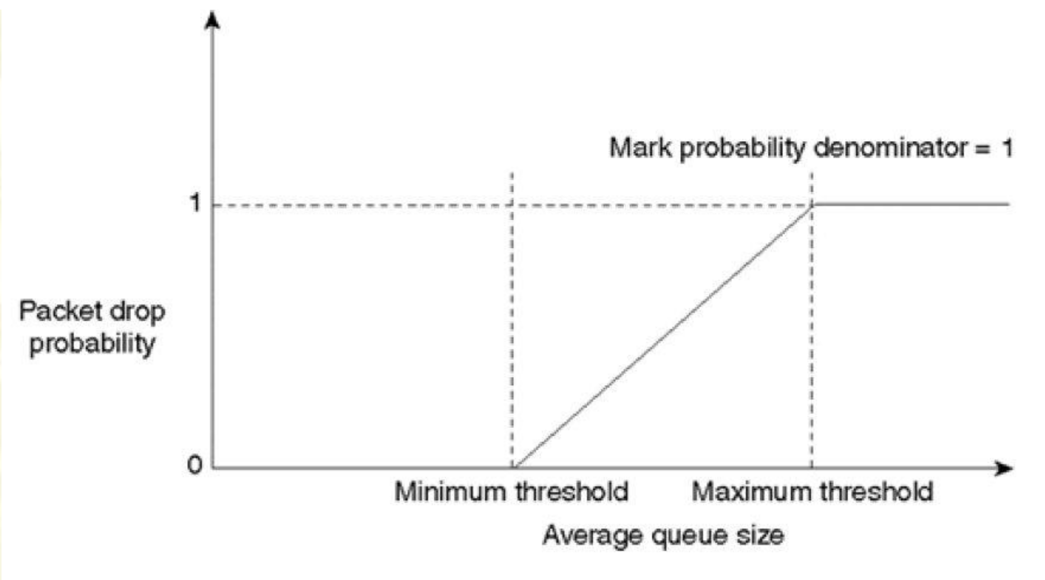
# CBWFQ - Test

- Test! Activate all the traffic at once
- VoIP
  - `iperf -u -c 192.168.2.2 -i 1 -b 100K`
  - `iperf -u -s`
- VIDEO
  - `iperf -u -c 192.168.1.3 -i 1 -b 500K`
  - `iperf -u -s`
- WEB
  - `iperf -c 192.168.1.2 -i 1 -p 80 -t 30`
  - `iperf -s`
- Get statistics
  - `show policy-map interface`

# Proactive Queue Management for Congestion Avoidance - RED

- RED is a congestion avoidance mechanism
- RED takes a ***proactive approach***, instead of waiting until the queue is full, it starts dropping packets with ***a non-zero drop probability*** when the average size is above a threshold

$$\text{packet drop probability} = \left(\frac{\text{(average queue length} - \text{minimum}}{\text{threshold)}}\right) \times \frac{\text{mark probability}}{\text{denominator}}$$




***Mark probability denominator*** is the fraction of packets dropped when the average queue depth is at the maximum threshold. If the mark probability denominator is 10, for example, 1 out of every 10 packets is dropped

# Configuring RED

- Enabling RED
  - `policy-map OUT`
  - `class VOIP`
  - `random-detect`
  - `no random-detect precedence`
- Tuning RED based on IP precedence
  - `random-detect dscp <dscp-value> <minimum-threshold> <maximum-threshold> <mark-prob-den>`
  - `random-detect dscp 46 40 60 10`
- Check RED status
  - `show policy-map interface`

# References

- Committed Access Rate: http://www.cisco.com/c/en/us/td/docs/ios/12_2/qos/configuration/guide/fqos_c/qcfcar.html
- Generic Traffic Shaping: http://www.cisco.com/c/en/us/td/docs/ios/12_2/qos/configuration/guide/fqos_c/qcfgts.html
- DSCP values: http://www.cisco.com/c/en/us/support/docs/quality-of-service-qos/qos-packet-marking/10103-dscpvalues.html
- http://www.cisco.com/en/US/technologies/tk543/tk766/technologies_white_paper09186a00800a3e2f.html
- Policy Based Routing: http://www.cisco.com/c/en/us/td/docs/ios/12_2/qos/configuration/guide/fqos_c/qcfpbr.html
- Class Based WFQ: http://www.cisco.com/c/en/us/td/docs/ios/12_0s/feature/guide/fswfq26.html
- RED: http://www.cisco.com/c/en/us/td/docs/ios/12_2/qos/configuration/guide/fqos_c/qcfwred.html