

# Queuing Methods

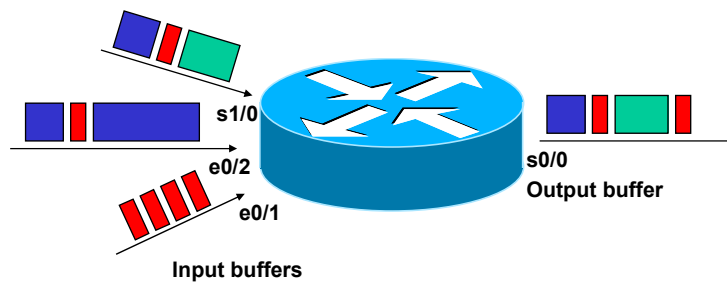
## From FIFO to CB-WFQ/PQ

(C) Herbert Haas 2009/05/02

### Need for Queuing



- Packet delivery and switching processes work at different (and varying) rates
- Buffers are needed to interface between those asynchronous processes
  - ♦ Too large buffers: Introduce more delay
  - ♦ Too small buffers: Packets might get lost during bursts



(C) Herbert Haas 2009/05/02 www.perihel.at

2

## How many queues?



- How many queues do we need practically?

- ♦ Citing Rich Seifert:

"You need at least two levels of priority. No one knows what they would do with more than eight levels of priority."

## Scheduling



- Queue = buffer
- Queuing means following a specific **scheduling algorithm** when taking packets out of the buffers
- Different queuing methods can be combined or cascaded
- **Queuing theory** models and analyses behavior of sophisticated scheduling algorithms applied on stochastic packet processes

## Note



- The current number of packets (or bytes) stored within a queue is called **queue depth**
- Under normal conditions the queue depth is typically approximating zero
- If the queue depth is greater zero for a particular (short) duration this typically leads to congestion quickly

## Congestion Management

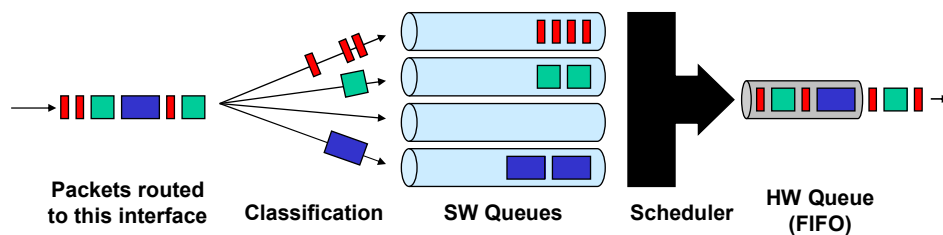


- If input and output rates mismatch for a specific amount of time
  - ◆ Buffer overrun vs. underrun
  - ◆ Persistent situation on weakest link
- Also **different dropping algorithms** may be applied upon congestion
  - ◆ Tail drop (default)
  - ◆ Class-based drop probability
  - ◆ Random early discard

## SW versus HW queues



- Queuing actually encompasses two parts: SW and HW queues!
- SW queuing is typically more sophisticated
  - ♦ WRR, CBWFQ, LLQ, etc
  - ♦ Implemented e. g. as doubly linked lists
- HW queuing is typically only FIFO
  - ♦ Called "TX-ring"
- SW queue only needed if HW-queue full
  - ♦ Otherwise packet bypasses SW-queue



(C) Herbert Haas 2009/05/02 www.perihel.at

7

## TX Ring Limit



- The size of the HW-queue
  - ♦ Is automatically derived from the interface bandwidth
  - ♦ Can be configured directly
- The HW-queue should be "appropriately" small
  - ♦ So that most of the packets experience the more sophisticated SW-queuing
- If HW-queue is too small: **Latency!**
  - ♦ Then almost all packets are SW-queued which increases the total delay (more CPU interrupts)
- If HW-queue is too large: **Jitter!**
  - ♦ Then almost no packet is SW-queued and the jitter increases

(C) Herbert Haas 2009/05/02 www.perihel.at

8

## TX Ring Limit - Configuration



```
(config-if)# tx-ring-limit <ring-limit>
```

- **Configure the maximum number of allowable packets that can be placed on the transmission ring**
  - ◆ Valid entries can be numbers from 1 to 32767
  - ◆ The default value is 60
- **Note:**
  - ◆ On Cisco 1700 series routers, possible values are 2 through 60
  - ◆ On Cisco 2600 and 3600 series routers, possible values are 3 through 60

## Note



- **Each interface has actually two rings**
  - ◆ One for RX, one for TX
- **SW-interfaces (loopback, tunnels, dialer, and virtual) don't have their own tx-ring**
  - ◆ Fully rely on the tx-ring of the associated HW interface

## Queuing Methods



- FIFO
- Priority Queuing
- Round Robin
- Weighted Round Robin
- Deficit Round Robin

Basic Queuing

- Class-based Queuing

Little Extension

- Fair Queuing
- Weighted Fair Queuing
- Class-based Weighted Fair Queuing
- LLQ

Sophisticated

## FIFO Queuing



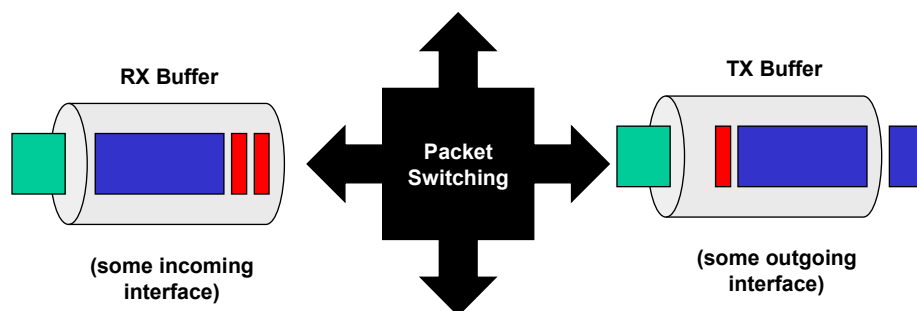
- **Fastest, simplest, and most common solution**
  - ◆ Typical implementation of HW buffers
  - ◆ Default on Cisco interfaces faster than E1 (2.048 Mbit/s)
- **Only one buffer for all type of packets available**
  - ◆ No classification
  - ◆ Order maintained

## FIFO Queuing (cont.)



- In case of congestion: **Tail drop**
  - ◆ Session starvation
  - ◆ Long delays
- Obviously each single queue of the more sophisticated queuing mechanisms are FIFO queues
- Preferred queuing mechanism
  - ◆ If low latency is required and no traffic classes should be identified
  - ◆ Available on all Cisco platforms and switching modes

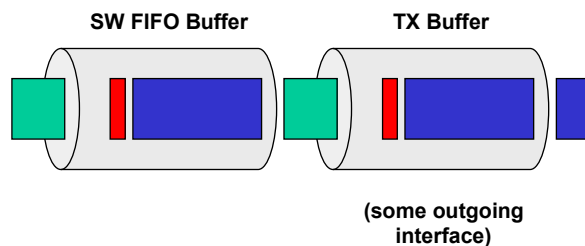
## FIFO Queuing (cont.)



## FIFO Queuing (cont.)



- Actually the SW FIFO buffer is only an extension to the HW FIFO buffer
  - ◆ Only used when HW buffer is congested



## Configuring FIFO Queuing



- FIFO is enabled by default on all interfaces with a bandwidth higher than 2 Mbit
- Default queuing method for interfaces below 2 Mbit is WFQ

```
interface serial0/0
  clockrate 64000
  no fair-queue
  hold-queue 80 out
  !
```

disables WFQ and enables FIFO

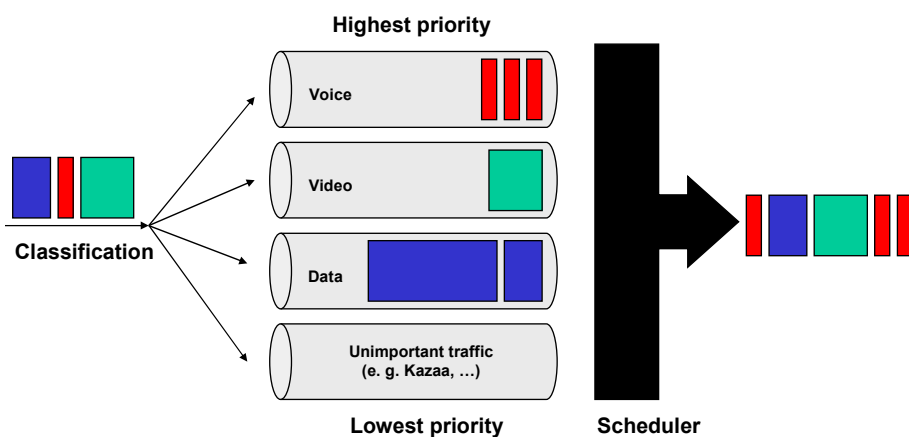
Queue depth is 80 packets – if queue is full, packet No. 81 will be dropped (tail-drop)

## Priority Queuing



- Different queues for different levels of priorities
- **Lower priority queue is only served if higher priority queue is empty !**
  - ◆ Low priority data may suffer from starvation if high priority queue depth does not vanish

## Priority Queuing (cont.)



## Configuring Priority Queuing



- **Four priority queues** can be defined on a Cisco Router
  - ◆ Hi, Medium, Normal, Low
  - ◆ As long as traffic is in a higher priority queue – lower priority traffic will not be enqueued
- FIFO queuing within priority queues
- **Queue-limit** (max packets) can be defined for each queue

## Configuring Priority Queuing (cont.)



- Traffic Classification Based On
  - ◆ Source Interface
  - ◆ Access-List definition
  - ◆ TCP/UDP Source/Destination port
  - ◆ Packet size
  - ◆ Fragments

```
interface serial1/0
  description Link to branch office intranet www-server
  priority-group 3

priority-list 3 protocol ip high list 101
priority-list 3 interface ethernet 0/0 medium
priority-list 3 interface ethernet 0/1 normal
priority-list 3 default low
priority-list 3 queue-limit 30 40 50 50

access-list 101 permit tcp any 10.1.1.1 eq 80
```

Activate PQ3 definition on Serial1/0

Hi: Traffic that matches ACL 101

Med: Incoming traffic from E0/0

Norm: Incoming traffic from E0/1

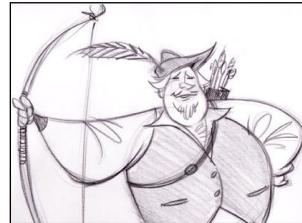
Low: any other traffic = "default"

Queue Limits Hi/Med/Norm/Low

## Round Robin Queuing



- Deterministic scheduling mechanism
- No prioritization
- Takes one packet from each queue in each round
- **Provides BW reservations**
- Also known as "Custom Queuing"

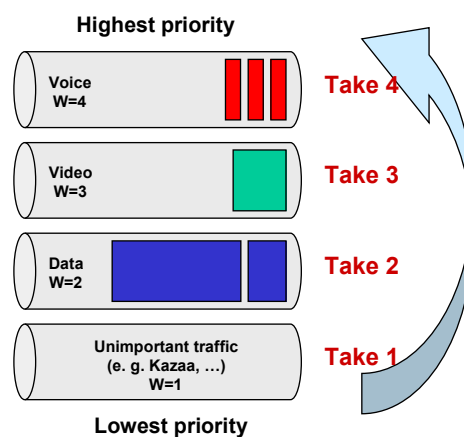


Round Robin

## Weighted Round Robin (WRR)



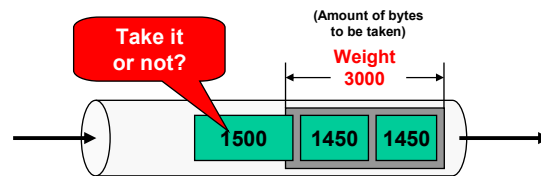
- Each queue has assigned a numeric weight
- Take up to  $\langle \text{weight} \rangle$  packets or bytes from each queue in each round
- **Byte-based weight is more fair** because packets have different sizes



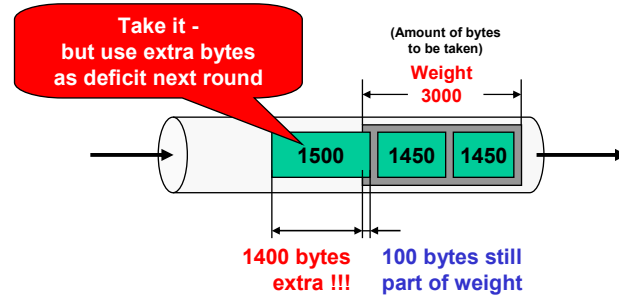
## Byte-based weight



- Weight (byte count) might extend over multiple packets
- But incomplete packets cannot be sent.  
Two strategies:
  - ♦ Either omit next packet (which would be incomplete) and send it next round
    - Costly for long packets, therefore short packets will experience higher rate on average
  - ♦ Or take the whole next packet and send it this round
    - Long packets gain more advantage and will experience higher rate on average



## Deficit Round Robin



- Solution:
  - ♦ Track extra bytes and subtract it from weight (of this queue) for the next round
  - ♦  $\text{Weight} = \text{Weight} - \text{"Deficit"}$

## Configuring WRR



- Custom Queuing (=WRR)
  - ♦ Up to 16 queues
  - ♦ Byte-based weight
  - ♦ Some queues can be configured as PQs
  - ♦ No deficit tracking(!)
- Modified Deficit Round Robin
  - ♦ Deficit Round Robin with a single PQ
  - ♦ Only on 12000 series routers

```
interface serial 0/1
  custom-queue-list 1
!
queue-list 1 protocol ip 1 list 101
queue-list 1 queue 1 byte-count 6000
queue-list 1 interface ethernet 1/1 2
queue-list 1 queue 2 byte-count 3000
queue-list 1 protocol ip 3
queue-list 1 queue 3 byte-count 5000
queue-list 1 default 4
!
access-list 101 permit ip any any precedence 5
```

Activate CQ1 definition on Serial0/1

4 Queue definitions:  
Q1: ACL definition / bc=6000  
Q2: Incoming traffic from e1/1 / bc=3000  
Q3: Any other IP traffic / bc=5000  
Q4: any other traffic

IP precedence 5 eg. used for VoIP

## "Class-based Queuing"



- If packets are classified without session context then this is generally called **class-based queuing**
- All discussed queuing methods can be implemented "class-based"
- Problem:
  - ♦ High-rate sessions dominate each queue
  - ♦ And low-rate sessions suffer!

# Fair Queuing (FQ) & Weighted Fair Queuing (WFQ)

(C) Herbert Haas 2009/05/02

## Fair Queuing



- Separates incoming traffic into "flows"
  - ◆ Flow = Packets belonging to the same session (=conversation, identified via socket information)
- **Round-robin over flows**
- Guarantees each flow an equal share of the transmission capacity

(C) Herbert Haas 2009/05/02 [www.perihel.at](http://www.perihel.at)

28

## Fair Queuing (cont.)



- **Ideally:**
  - ◆ Determine the number of active flows
  - ◆ Store packets of each conversation in a separate queue
  - ◆ Serve queues round-robin bit-by-bit
- **Reality:**
  - ◆ Packet cannot be transmitted one bit-by-bit
  - ◆ But this can be approximated...

## Fair Queuing (cont.)



- **Hash function** determines the queue of a packet
  - ◆ Hash (proto, ToS, SA, DA, SP, DP)
  - ◆ More queues than sessions must exist to reduce hashing collisions
- Note that **varying parameters** result in different queue mappings
  - ◆ ToS through re-marking policy (avoid!)
  - ◆ Applications that change port numbers

## Weighted Fair Queueing



- IP Precedence is used as weight
- IOS reserves
  - ◆ A fixed number of queues depending on interface bandwidth
    - At least 16 queues (BW ≤ 64 kbit/s)
    - At most 256 queues (BW ≥ 512 kbit/s)
  - ◆ 8 queues for system sessions and
  - ◆ Up to 1000 queues for RSVP sessions

## High collision probability!



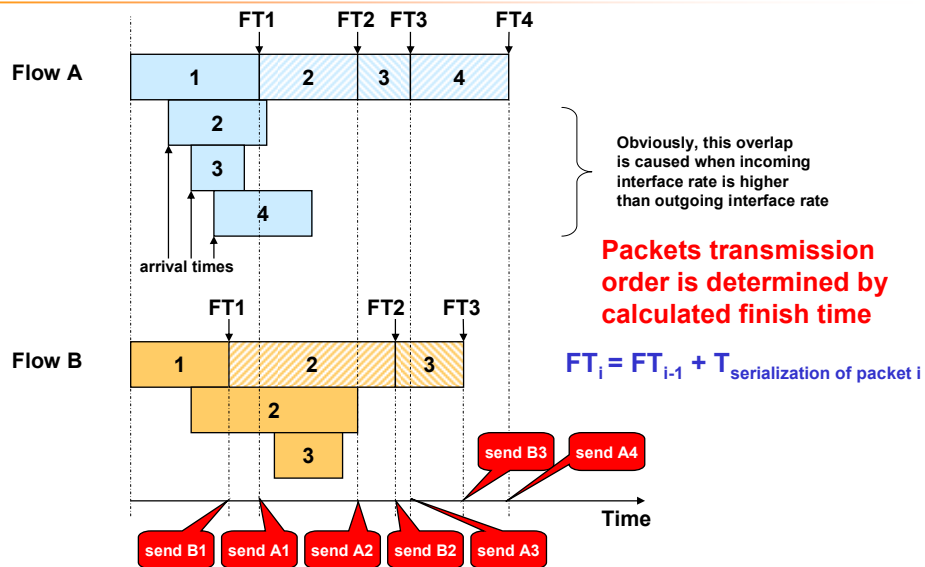
- Collision probability (two flows mapped to the same queue) can be calculated via
  - ◆  $P = Q! / (Q^F (Q-F)!)$  with
    - Q ... number of queues and
    - F ... number of flows
- Conclusion from formula: Q should be at least 20 times greater than F
  - ◆ Seldom achievable

# Finish Time based Scheduling



- **The FQ-Scheduler always sends the packet with the smallest finish time (FT)**
  - ♦ In other words, the packet sent order is given by their calculated FT
- **The FT-based scheduling algorithm is fair because each flow is given the same bandwidth – on average!**
  - ♦ A flow with small packets is serviced more often
  - ♦ A flow with large packets is serviced less often
- **The packet size is proportional to the serialization time**
  - ♦ By constant factor 1/BW

# Finish Time based Scheduling (cont.)



## Weighting



- **Weighted fair queuing gives certain flows a larger portion of the link capacity**
  - ◆ Certain flows have more “weight”
  - ◆ Time division multiplexing with unequal time slots
- **Weighting based on**
  - ◆ IP precedence bits in the TOS (Type of Service) field in the IP packet header
  - ◆ A signalling procedure (RSVP)

## WFQ Congestion Management



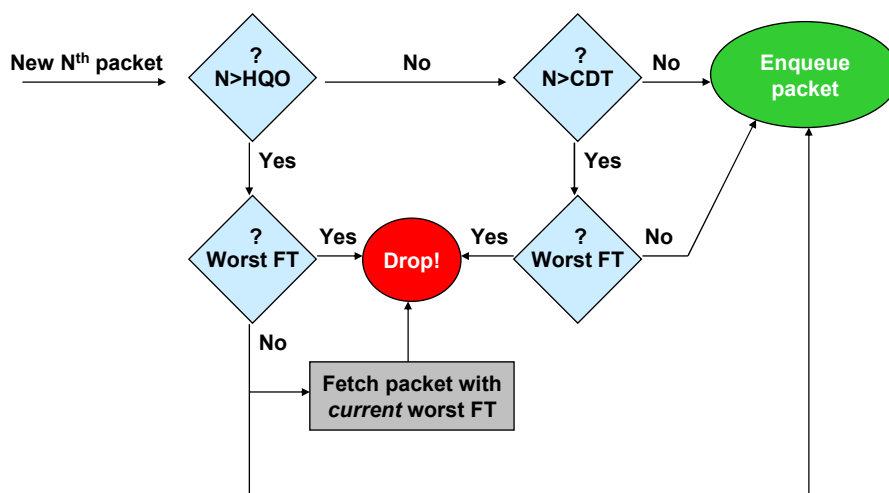
- **Congestion discard threshold (CDT)**
  - ◆ Max total number of packets until starting of **early dropping** of packets of the **most aggressive flow**
- **Hold-queue out (HQO) limit**
  - ◆ Max number of packets the WFQ subsystem can handle
  - ◆ Aggressive dropping of packets of the most aggressive flow when the *total* number of packets reaches HQO
- **Dropping strategy is independent of IP Precedence**

## WFQ Congestion Management (cont.)



- What is the most aggressive flow?
  - ♦ It has the worst Finish Time (FT)
- New  $N^{\text{th}}$  packet can be enqueued only
  - ♦ If  $N < \text{HQO}$  and  $N < \text{CDT}$
  - ♦ Or the packet has NOT the worst FT
- The first packet in a queue is never dropped

## WFQ Enqueuing and Dropping Summary



## Weighting: Virtual packet sizes!



- A "weight" can be assigned to flows if we deal with **virtual packet sizes**
  - ◆ Idea: Reduce the virtual packet size of higher precedence flows!
- Divide the true packet size by the IP Precedence number
  - ◆ (plus 1 to avoid dividing by zero)
  - ◆ Or practically simpler: Multiply the true packet size with a fixed integer depending on the IP Precedence
    - IOS uses as factor either  $\text{floor}(32384/(\text{IPP}+1))$  or  $\text{floor}(4096/(\text{IPP}+1))$  depending on version

## WFQ Facts – Advantages



- **Simple configuration (interface only)**
- **Fair throughput among flows**
- **Supported on all IOS platforms**
  
- **WFQ is per default enabled**
  - ◆ If BW  $\leq$  E1 (2.048 Mbit/s)
  - ◆ On interfaces configured for MLPPP

## WFQ Facts – Drawbacks



- **High collision probability (several flows in one queue)**
  - ◆ Higher rate flows dominate
- **Only checks IP Precedence for weighting**
  - ◆ No other classification possible – at least before 12.4(20)T
  - ◆ \*\*\* NOTE: Since IOS 12.4(20)T fair-queuing is also supported via MQC \*\*\*
- **No fixed bandwidth guarantees**

## WFQ Configuration



```
(config-if) # fair-queue [<congestive-discard-threshold>
                    [<dynamic-queues>
                    [<reservable-queues>]]]
(config-if) # hold-queue <max-limit> out
```

congestive-discard-threshold	Max total number of packets until early dropping of most aggressive flow is started. Values: 1-4096, default: 64 (HQF: 16-4096)
dynamic-queues	Number of queues for normal data traffic Values: 2^n, with n={4..12} which is 16..4096
reservable-queues	Number of queues for reserved conversations such as RSVP Values: 0-1000, default: 0
max-limit	Max total number of packets the FIFO or WFQ system can handle (default: 1000)

# CBWFQ and LLQ

## Little but important extensions

(C) Herbert Haas 2009/05/02

## Class-based WFQ (CBWFQ)



- **CBWFQ allows to freely define traffic classes**
  - ♦ Using DSCP, protocol or port numbers, ACLs, etc. (`class-map` command)
  - ♦ Each traffic class has its own queue
  - ♦ **Max 64 queues (=classes)**
- **For each queue (=class) various parameters can be configured**
  - ♦ **Bandwidth** (kbps | % | remaining %)
  - ♦ **Queue limit** (max packets in this queue)
  - ♦ **Drop mechanism** (tail or early)

(C) Herbert Haas 2009/05/02 [www.perihel.at](http://www.perihel.at)

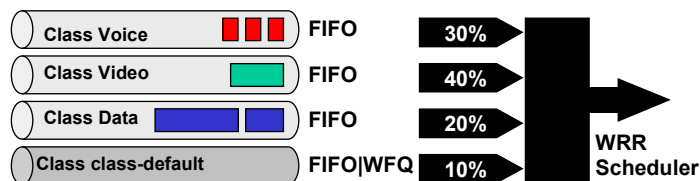
44

## Note



- **CBWFQ is actually not a true fair-queuing method !**
  - ◆ Fair-queuing would require that round-robin is done over sessions and each session is mapped to a dedicated queue
  - ◆ But CBWFQ puts all packets (of all conversations) that match a given characteristic (e. g. DSCP) into a single dedicated queue
- Therefore it should be called "CBWRR"

## CBWFQ



- All user-definable classes are mapped to FIFO-queues
- There is always a class "**class-default**" which is either a FIFO queue or WFQ
  - ◆ WFQ per default unless bandwidth command specified

## Available Bandwidth



- Only the "**available-bandwidth**" can be shared among queues
  - ♦ By default 25% of interface bandwidth is reserved for important network traffic (routing or keepalives, etc.)
  - ♦ **Note: *bandwidth not clock rate !!!***
- Therefore available-bandwidth is **75%**
  - ♦ Can be changed using the **max-reserved-bandwidth** interface command
  - ♦ In a policy map the sum of configured bandwidths cannot exceed 75% of the interface BW
    - Otherwise you cannot attach the policy map on an interface (error message)

## CBWFQ Configuration Example



```
class-map voice_traffic
  match access-group 101
class-map data_traffic
  match input-interface ethernet0
!
policy-map policy1
  class voice_traffic
    bandwidth 64
  class data_traffic
    bandwidth 16
  class-map class-default
    bandwidth 16
!
interface serial0
  ip address 10.1.1.1 255.255.255.0
  bandwidth 128
  service-policy output policy1
!
access-list 101 permit ip any any precedence critical
!
```

2 traffic classes defined

bandwidth assignment for the classes.

If *class-default* is not configured all unmatched traffic will be queued "flow-based" (WFQ)

The aggregate bandwidth must be **<= 75% of the interface bandwidth!**

## Match-all vs. Match-any



- Each class-map may contain one or more match entries
- Per default all entries must match to make the whole class-map result in a match
  - ♦ Corresponds to the default keyword: match-all
- Optionally the class-map may match if only a single match entry matches
  - ♦ Optional keyword: match-any

```
(config)# class-map match-all MY_CLASS_MAP
(config-cmap)# match access-group <ACL>
(config-cmap)# match input-interface <intf>
...
(config)# class-map match-any MY_CLASS_MAP
(config-cmap)# match access-group <ACL>
(config-cmap)# match input-interface <intf>
...
```

## CBWFQ Configuration



### Things to match within a class-map (most important)

```
(config-cmap)# match access-group <ACL>
(config-cmap)# match input-interface <intf>
(config-cmap)# match protocol <proto>
(config-cmap)# match mpls experimental <number>
```

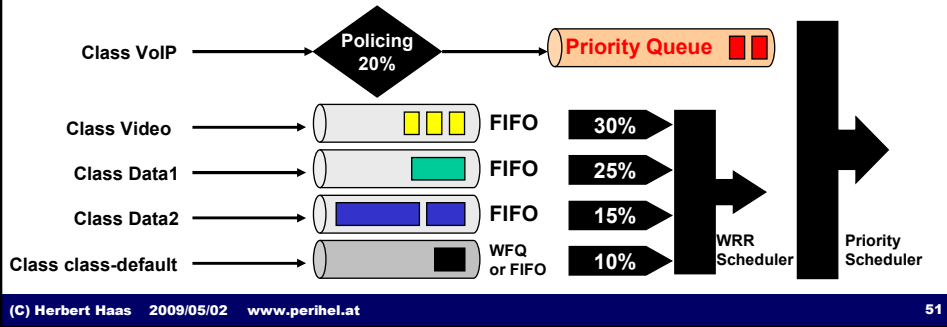
### Options within a policy-map

```
(config-pmap-c)# bandwidth <kbit/s>
(config-pmap-c)# bandwidth percent <percent>
(config-pmap-c)# bandwidth remaining percent <percent>
(config-pmap-c)# queue-limit <number-of-packets>
(config-pmap-c)# random-detect !!! enables WRED using defaults
(config-pmap-c)# random-detect exponential-weighting-constant <exp>
(config-pmap-c)# random-detect precedence <precedence>
                                     <min-threshold>
                                     <max-threshold>
                                     <mark-prob-denominator>
(config-pmap-c)# fair-queue [<CDT> [<number-of-dynamic-queues>]]
```

# Low Latency Queuing (LLQ)



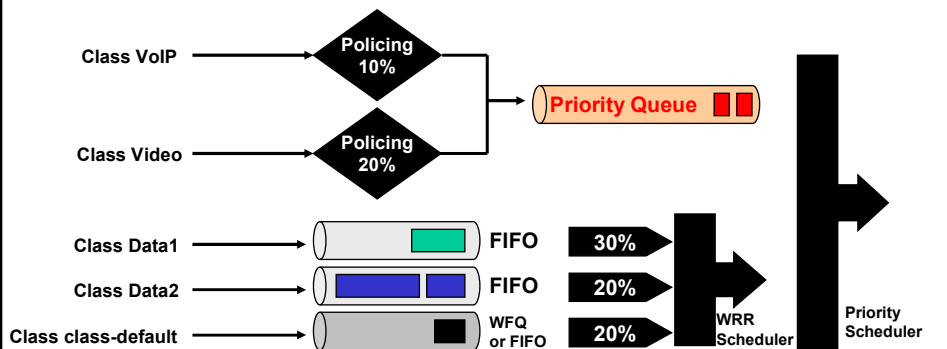
- LLQ = CBWFQ + one PQ
  - ♦ Packets in PQ are always preferred
  - ♦ If PQ empty, WRR scheduler can handle the other queues
- PQ is *policed* to a specified bandwidth to avoid starvation of CBWFQ
  - ♦ Only if interface is congested



# Multiple Priority Classes



- Multiple traffic classes can be configured to use the priority queue
  - ♦ With different policing parameters
  - ♦ But all packets enter the single FIFO PQ



# LLQ Configuration



## Options within a policy-map

```
(config-pmap-c)# priority <bandwidth> [<burst>]  
(config-pmap-c)# priority percent <percentage> [<burst>]
```

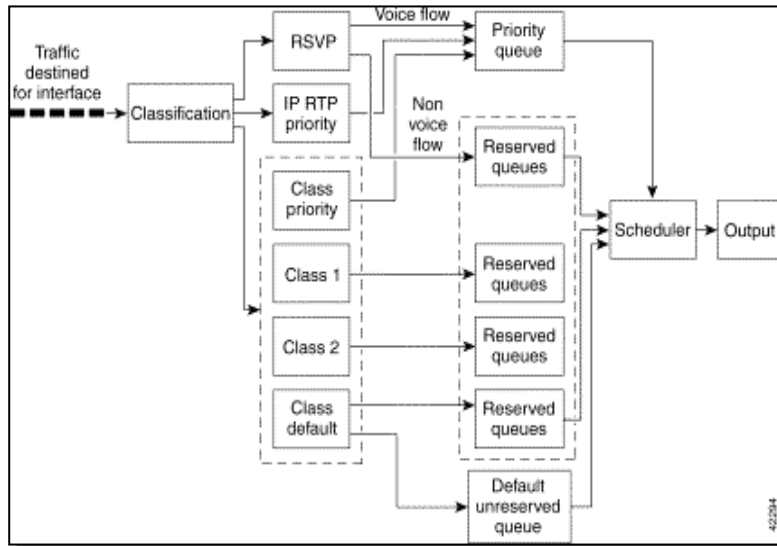
- **Parameters:**
  - ♦ **bandwidth (kbps or %) – cannot exceed 75% of interface BW by default**
  - ♦ **burst (32-2000000 bytes) – measured during 200 ms at interface rate**

# Low Latency Queuing (LLQ)



- **Aggregate bandwidth of PQ and CBWFQ should not be higher than 75% of link bandwidth**
  - ♦ **25% of the link bandwidth should be reserved for important network traffic**
    - E.g. routing protocol updates or layer 2 keepalives
- **Practically:** Consider to remove or reduce this "recommended" setting
  - ♦ **Especially on links where no routing updates are set (WAN to ISP)**
  - ♦ **max-reserved-bandwidth 100** command

# LLQ Details



# LLQ



```

class-map match-any ef-traffic
  match dscp ef
  match access-group 100
class-map match-all af21-traffic
  match dscp af21
class-map match-all af31-traffic
  match dscp af31
class-map match-all af11-traffic
  match dscp af11
class-map match-all cs1-traffic
  match dscp cs1
class-map match-all class-default
  match any
    
```

```

policy-map llq-policy
  class ef-traffic
    priority 200
  class af31-traffic
    bandwidth remaining percent 50
  class af21-traffic
    bandwidth remaining percent 20
  class af11-traffic
    bandwidth remaining percent 15
  class cs1-traffic
    bandwidth remaining percent 5
  class class-default
    bandwidth remaining percent 10
    
```

# LLQ Config



## Example 1: PQ defined via IP RTP-Priority

```
class-map ftp_traffic
  match access-group 101
class-map int_e0
  match input-interface ethernet0
!
policy-map policy1
  class ftp_traffic
    bandwidth 64
    random-detect precedence 0 32 256 100
  class int_e0
    bandwidth 16
!
interface serial0
  service-policy output policy1
  bandwidth 128
  ip rtp priority 16384 16383 40
!
access-list 101 permit tcp any any eq ftp
!
```

2 Traffic Classes defined

Weighted Random Early Detection

Priority Queue

# LLQ Config



## Example 2: PQ defined with access-list

```
class-map match-all voice_traffic
  match access-group 101
class-map match-all voice_signaling
  match access-group 102
!
policy-map voice_policy
  class voice_traffic
    priority 45
  class voice_signaling
    bandwidth 8
    queue-limit 40
  class class-default
    fair-queue 16
    queue-limit 20
!
interface serial0
  bandwidth 128
  service-policy output voice-policy
!
access-list 101 permit udp any any range 16384 32767
access-list 102 permit tcp any eq 1720 any
access-list 102 permit tcp any any eq 1720
!
```

2 Traffic Classes defined

1 Priority Queue

1 Weighted Fair Queue with "tail-drop" after 40 queued packets

16 Dynamic "flow-based" default weighted fair queues with "tail-drop" after 20 queued packets

# LLQ Config



## Example 3: LLQ with VoIP over Frame Relay

```
class-map match-all voice_traffic
  match access-group 101
class-map match-all voice_signaling
  match access-group 102
!
policy-map voice_policy
  class voice_traffic
    priority 45
  class voice_signaling
    bandwidth 8
  class class-default
    fair-queue
!
map-class frame-relay VoIPoverFR
  frame-relay cir 64000
  frame-relay bc 640
  frame-relay mincir 64000
  service-policy output voice_policy
  frame-relay fragment 80
!
interface serial0
  no ip address
  encapsulation frame-relay
  frame-relay traffic-shaping
!
interface serial0.1 point-to-point
  bandwidth 128
  ip address 10.1.1.1 255.255.255.0
  frame-relay interface-dlci 500
  class VoIPoverFR
!
access-list 101 permit udp any any range 16384 32767
access-list 102 permit tcp any eq 1720 any
access-list 102 permit tcp any any eq 1720
```

← 2 Traffic Classes defined

← 1 Priority Queue  
2 Weighted Fair Queues

← Frame-Relay Map-Class with Traffic Shaping parameters and FRF.12 fragmentation setting  
default mincir = CIR / 2; service-policy aggregate bandwidth must be <= mincir!

# Monitoring LLQ



```
!
Router# show queue serial0
Router# debug priority
Router# show policy interface voice_policy
!
```

# Queuing Verification Commands



```
# show interface s0/1    !!! Shows CDT and HQO and current number of
                        !!! conversations.

# show queue s0/1       !!! information about queuing subsystem
                        !!! such as queuing strategy, number of
                        !!! conversations, queue-depth and limits,
                        !!! conversation details (!), etc.

# show queuing
# show queuing fair     !!! Status of fair-queuing
```