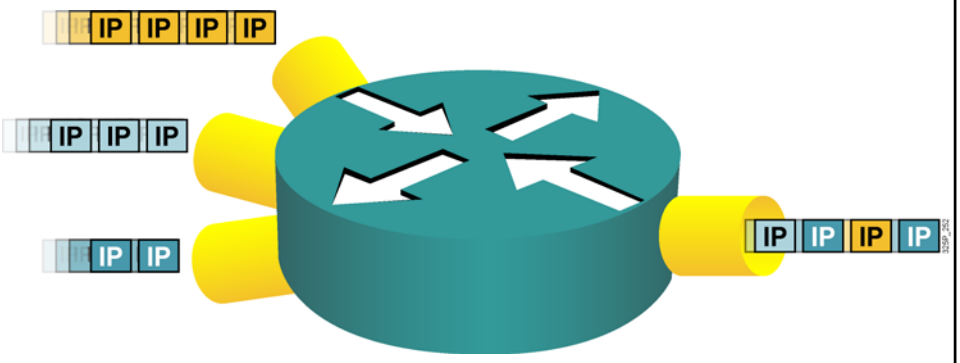
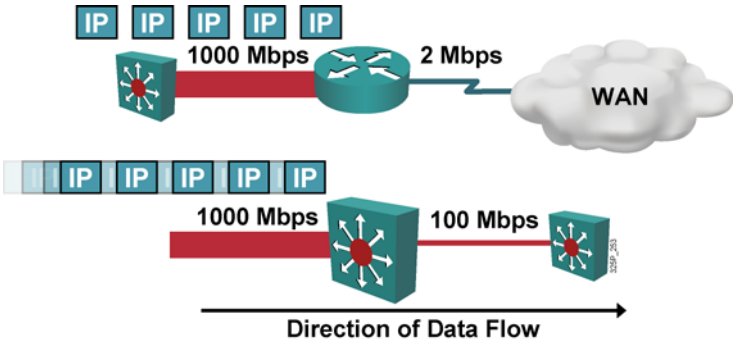
Congestion and Queuing



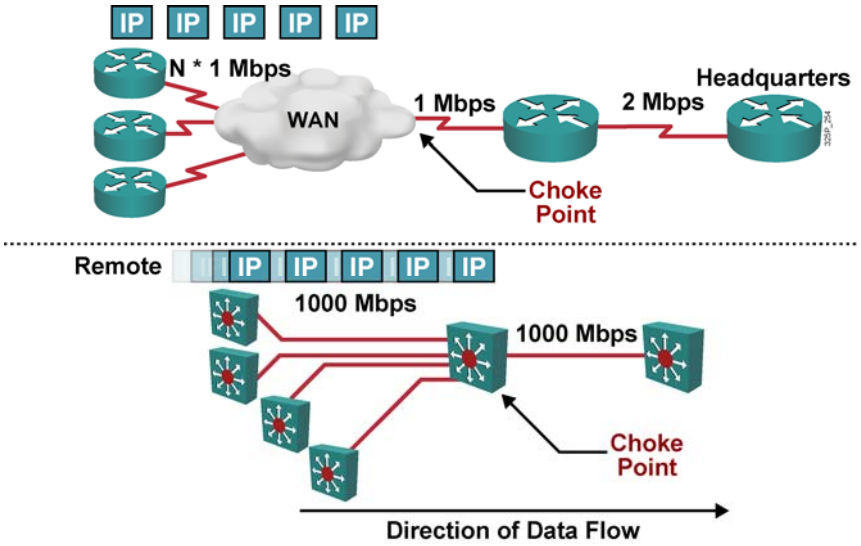
- Congestion can occur at any point in the network where there are points of speed mismatches or aggregation.
- Queuing manages congestion to provide bandwidth and delay guarantees.

Speed Mismatch



- **Speed mismatches are the most typical cause of congestion.**
- **Possibly persistent when going from LAN to WAN.**
- **Usually transient when going from LAN to LAN.**

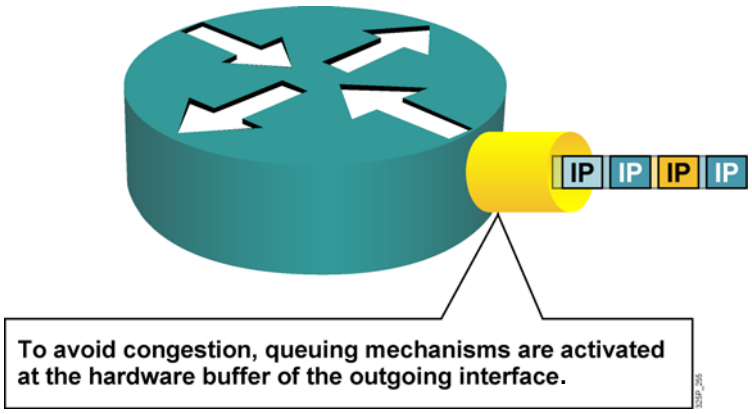
Aggregation



What is Queuing?

- Queuing is a congestion-management mechanism that allows you to control congestion on interfaces.
- Queuing is designed to accommodate temporary congestion on an interface of a network device by storing excess packets in buffers until bandwidth becomes available.

Congestion and Queuing

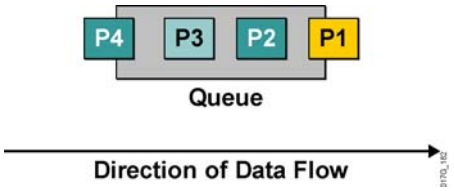


Queuing Algorithms

- First-in, first-out (FIFO)
- Priority queuing (PQ)
- Round robin
- Weighted round robin (WRR)

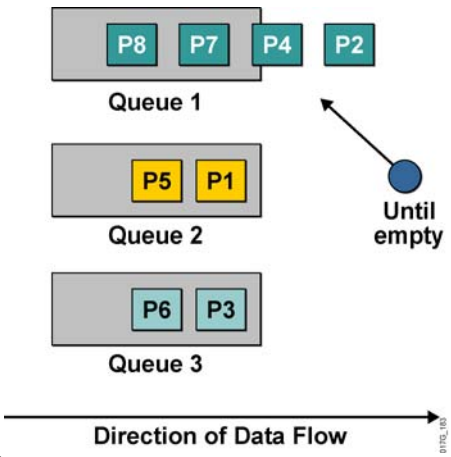
FIFO

- First packet in is first packet out
- Simplest of all
- One queue
- All individual queues are FIFO



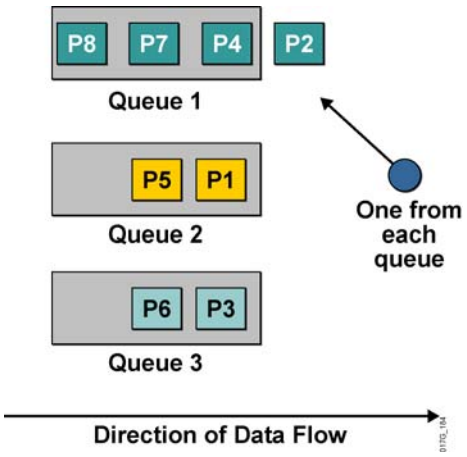
Priority Queuing

- Uses multiple queues
- Allows prioritization
- Always empties first queue before going to the next queue:
- Empty queue number 1.
- If queue number 1 is empty, then dispatch one packet from queue number 2.
- If both queue number 1 and queue number 2 are empty, then dispatch one packet from queue number 3.
- Queues number 2 and number 3 may “starve”



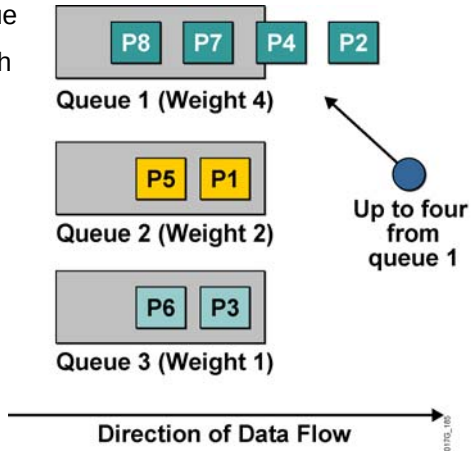
Round Robin Queuing

- Uses multiple queues
- No prioritization
- Dispatches one packet from each queue in each round:
  - One packet from queue number 1
  - One packet from queue number 2
  - One packet from queue number 3
  - Then repeat

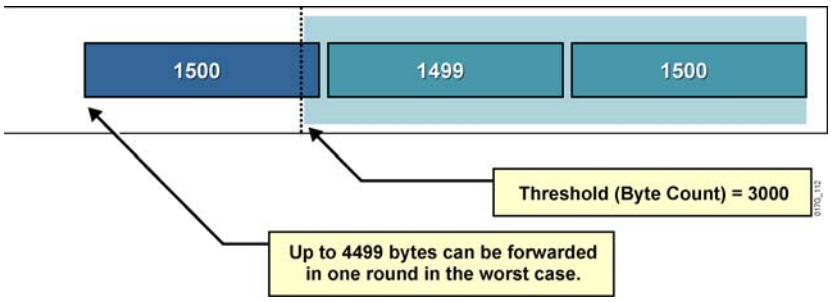


Weighted Round Robin Queuing

- Allows prioritization
- Assign a weight to each queue
- Dispatches packets from each queue proportionately to an assigned weight:
- Dispatch up to four from queue number 1.
- Dispatch up to two from queue number 2.
- Dispatch 1 from queue number 3.
- Go back to queue number 1.



Problems with Weighted Round Robin Queuing

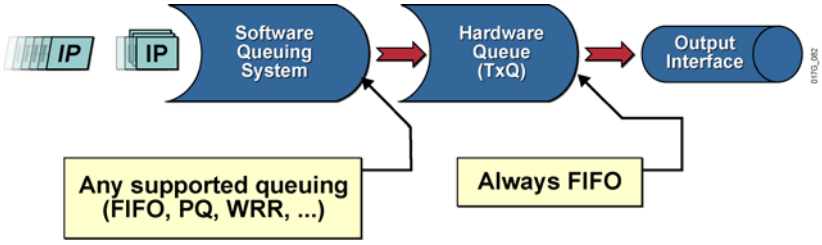


- Problem with WRR:

  Some implementations of WRR dispatch a configurable number of bytes (threshold) from each queue for each round—several packets can be sent in each turn.

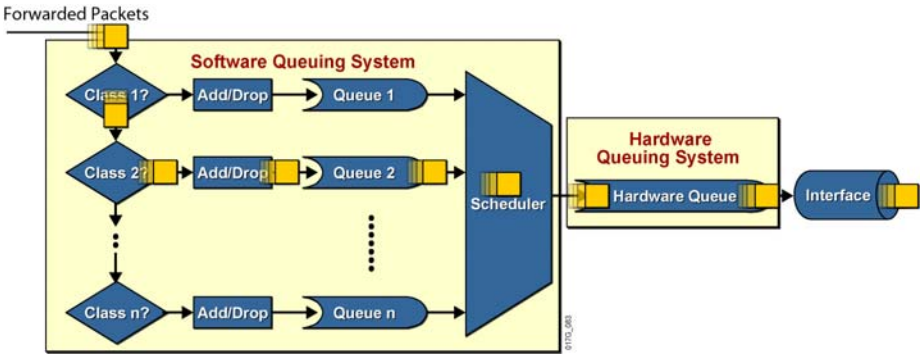
  The router is allowed to send the entire packet even if the sum of all bytes is more than the threshold.

Router Queuing Components



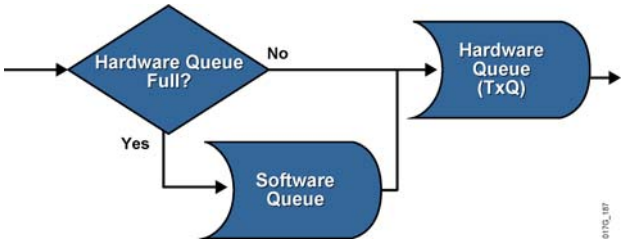
- Each physical interface has a hardware and a software queuing system.

Hardware and Software Router Queuing Components



- The hardware queuing system always uses FIFO queuing.
- The software queuing system can be selected and configured depending on the platform and Cisco IOS version.

The Software Queue



- Generally, a full hardware queue indicates interface congestion, and software queuing is used to manage it.
- When a packet is being forwarded, the router will bypass the software queue if the hardware queue has space in it (no congestion).

The Hardware Queue

- Routers determine the length of the hardware queue based on the configured bandwidth of the interface.
- The length of the hardware queue can be adjusted with the **`tx-ring-limit`** command.
- Reducing the size of the hardware queue has two benefits:
  - It reduces the maximum amount of time that packets wait in the FIFO queue before being transmitted.
  - It accelerates the use of QoS in Cisco IOS software.
- Improper tuning of the hardware queue may produce undesirable results:
  - A long transmit queue may result in poor performance of the software queuing system.
  - A short transmit queue may result in a large number of interrupts, which causes high CPU utilization and low link utilization.

Monitoring Hardware Queue Transmit Queue Length

- The **show controllers serial 0/1/0** command shows the length of the hardware queue.

```
R1#show controllers serial 0/1/0
Interface Serial0/1/0
Hardware is GT96K
DCE V.11 (X.21), clock rate 384000

<...part of the output omitted...>
1 sdma_rx_reserr, 0 sdma_tx_reserr
0 rx_bogus_pkts, rx_bogus_flag FALSE
0 sdma_tx_ur_processed

tx_limited = 1(2), errata19 count1 - 0, count2 - 0
Receive Ring
rxr head (27)(0x075BD090), rxr tail (0)(0x075BCEE0)
  rmd(75BCEE0): nbd 75BCEF0 cmd_sts 80800000 buf_sz 06000000 buf_ptr
75CB8E0
  rmd(75BCEF0): nbd 75BCF00 cmd_sts 80800000 buf_sz 06000000 buf_ptr
75CCC00
<...rest of the output omitted...>
```

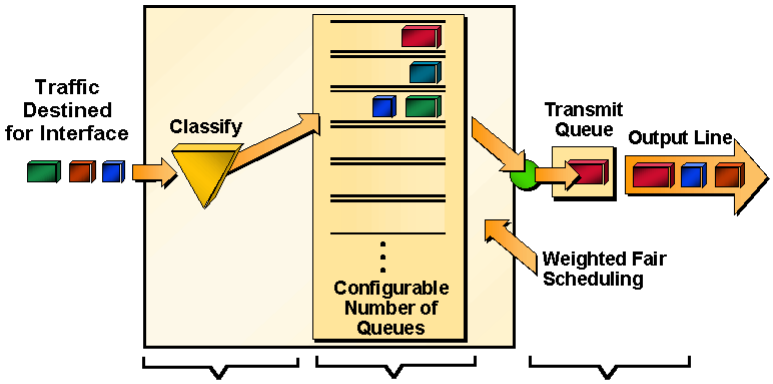
Congestion on Software Interfaces

- Subinterfaces and software interfaces (dialers, tunnels, Frame Relay subinterfaces) do not have their own separate transmit queue.
- Subinterfaces and software interfaces congest when the transmit queue of their main hardware interface congests.
- The tx-ring state (full, not-full) is an indication of hardware interface congestion.
- The terms “TxQ” and “tx-ring” both describe the hardware queue and are interchangeable.

Weighted Fair Queuing (WFQ)

- A queuing algorithm should share the bandwidth fairly among flows by:
  - Reducing response time for interactive flows by scheduling them to the front of the queue
  - Preventing high-volume flows from monopolizing an interface
- In the WFQ implementation, conversations are sorted into flows and transmitted by the order of the last bit crossing its channel.
- Unfairness is reinstated by introducing weight to give proportionately more bandwidth to flows with higher IP precedence (lower weight).
- The terms “WFQ flows” and “conversations” can be interchanged.

WFQ Operation

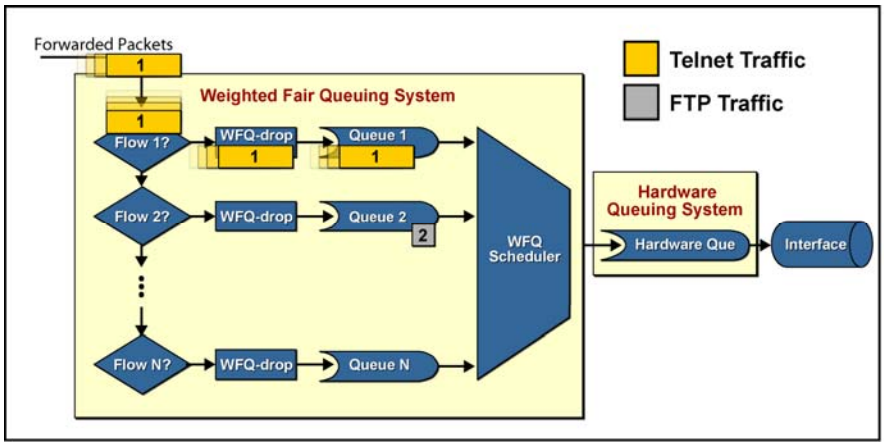


Flow-Based Classification by:
- Source and destination address
- Protocol
- Session identifier (port/socket)

Interface Buffer Resources

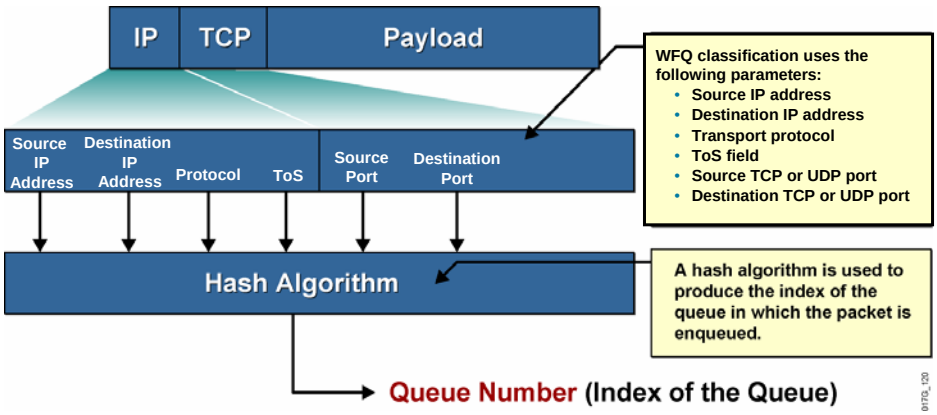
Weight Determined by:
- Required QoS (IP Procedure, RSVP)
- Flow throughput inversely proportional
- Frame Relay FECN, BECN, DE (For FR Traffic)

WFQ Architecture



- WFQ uses per-flow FIFO queues.

WFQ Classification



- Packets of the same flow end up in the same queue.

Benefits and Drawbacks of WFQ

Benefits	– Simple configuration (no need for classification to be configured) – Guarantees throughput to all flows – Drops packets of most aggressive flows – Supported on most platforms – Supported in most Cisco IOS versions
Drawbacks	– Possibility of multiple flows ending up in one queue – Lack of control over classification – Supported only on links less than or equal to 2 Mb – Cannot provide fixed bandwidth guarantees

Monitoring WFQ

```
router>
show interface interface
```

- **Displays interface delays including the activated queuing mechanism with the summary information**

```
Router>show interface serial 1/0
  Hardware is M4T
  Internet address is 20.0.0.1/8
  MTU 1500 bytes, BW 19 Kbit, DLY 20000 usec, rely 255/255, load
147/255
  Encapsulation HDLC, crc 16, loopback not set
  Keepalive set (10 sec)
  Last input 00:00:00, output 00:00:00, output hang never
  Last clearing of "show interface" counters never
  Input queue: 0/75/0 (size/max/drops); Total output drops: 0
  Queueing strategy: weighted fair
  Output queue: 0/1000/64/0 (size/max total/threshold/drops)
     Conversations  0/4/256 (active/max active/max total)
     Reserved Conversations 0/0 (allocated/max allocated)
  5 minute input rate 18000 bits/sec, 8 packets/sec
  5 minute output rate 11000 bits/sec, 9 packets/sec
```


Monitoring WFQ Interface

```
router>
show queue interface-name interface-number
```

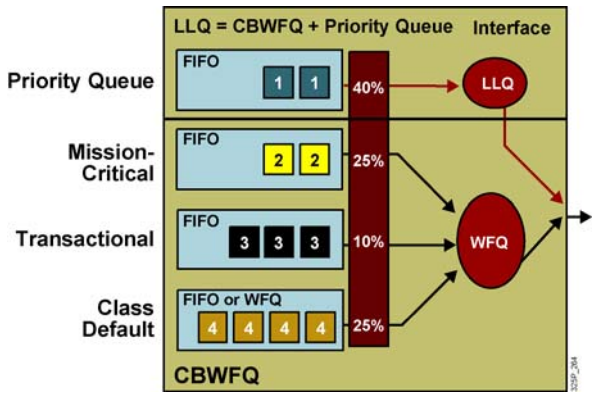
- **Displays detailed information about the WFQ system of the selected interface**

```
Router>show queue serial 1/0
  Input queue: 0/75/0 (size/max/drops); Total output drops: 0
  Queueing strategy: weighted fair
  Output queue: 2/1000/64/0 (size/max total/threshold/drops)
     Conversations  2/4/256 (active/max active/max total)
     Reserved Conversations 0/0 (allocated/max allocated)

  (depth/weight/discards/tail drops/interleaves) 1/4096/0/0/0
  Conversation 124, linktype: ip, length: 580
  source: 193.77.3.244, destination: 20.0.0.2, id: 0x0166, ttl: 254,
  TOS: 0 prot: 6, source port 23, destination port 11033

  (depth/weight/discards/tail drops/interleaves) 1/4096/0/0/0
  Conversation 127, linktype: ip, length: 585
  source: 193.77.4.111 destination: 40.0.0.2, id: 0x020D, ttl: 252,
  TOS: 0 prot: 6, source port 23, destination port 11013
```

Combining Queuing Methods

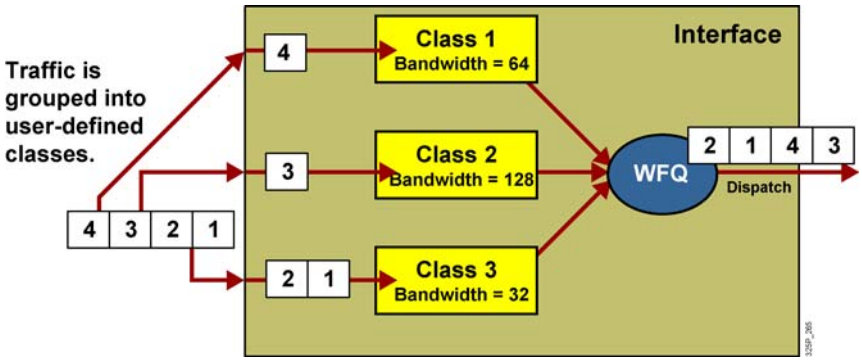


- Basic methods are combined to create more versatile queuing mechanisms.

Class-Based Weighted Fair Queuing

- CBWFQ is a mechanism that is used to guarantee bandwidth to classes.
- CBWFQ extends the standard WFQ functionality to provide support for user-defined traffic classes:
  - Classes are based on user-defined match criteria.
  - Packets satisfying the match criteria for a class constitute the traffic for that class.
- A queue is reserved for each class, and traffic belonging to a class is directed to that class queue.

CBWFQ Architecture



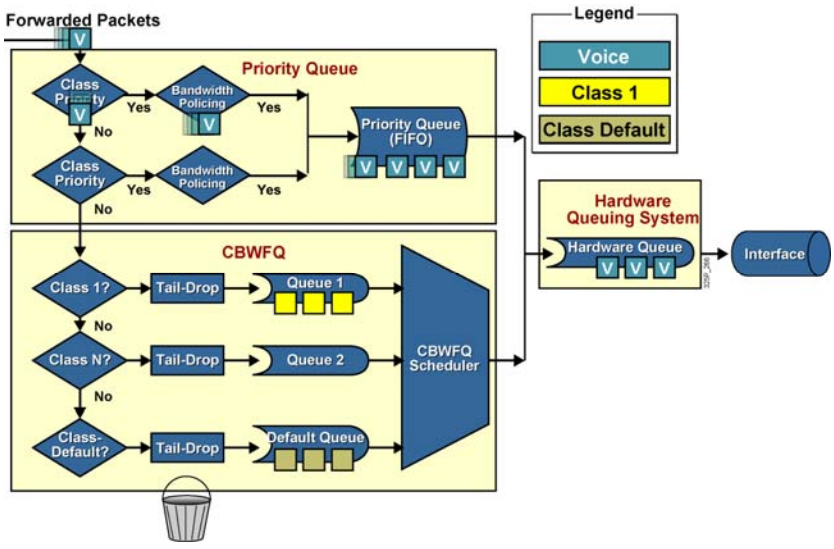
CBWFQ Benefits and Drawbacks

Benefits	–Custom-defined classifications –Minimum bandwidth allocation –Finer granularity and scalability
Drawback	–Voice traffic can still suffer unacceptable delay

Low Latency Queuing (LLQ)

- A priority queue is added to CBWFQ for real-time traffic.
- High-priority classes are guaranteed:
  - Low-latency propagation of packets
  - Bandwidth
- High-priority classes are also policed when congestion occurs—they then cannot exceed their guaranteed bandwidth.
- Lower-priority classes use CBWFQ.

LLQ Architecture



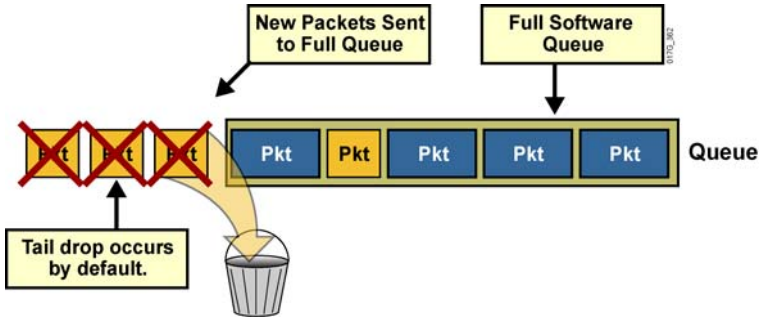
LLQ Benefits

- High-priority classes are guaranteed:
  - Low-latency propagation of packets
  - Bandwidth
- Configuration and operation are consistent across all media types.
- Entrance criteria to a class can be defined by an ACL:
  - Not limited to UDP ports as with IP RTP priority
  - Defines trust boundary to ensure simple classification and entry to a queue

Configuring LLQ (Cont.)

```
class-map voip
 match ip precedence 5
!
class-map mission-critical
 match ip precedence 3 4
!
class-map transactional
 match ip precedence 1 2
!
policy-map Policy1
  class voip
   priority percent 10
  class mission-critical
   bandwidth percent 30
  class transactional
   bandwidth percent 20
  class class-default
   fair-queue
```

Managing Interface Congestion with Tail Drop

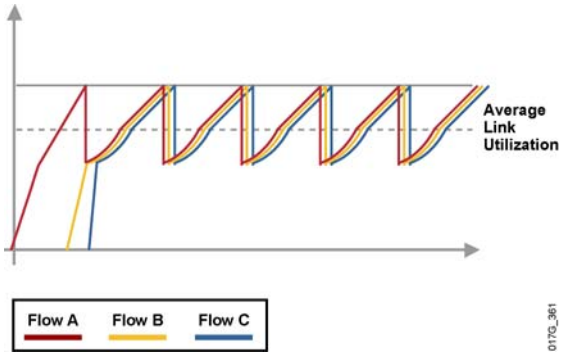


- Router interfaces experience congestion when the output queue is full:
  - Additional incoming packets are dropped.
  - Dropped packets may cause significant application performance degradation.
  - Tail drop has significant drawbacks.

Tail Drop Limitations

- In some situations, simple tail drop should be avoided because it contains significant flaws:
  - Dropping can affect TCP synchronization.
  - Dropping can cause TCP starvation.
  - There is no differentiated drop—high-priority traffic is dropped as easily as low-priority traffic.

TCP Synchronization

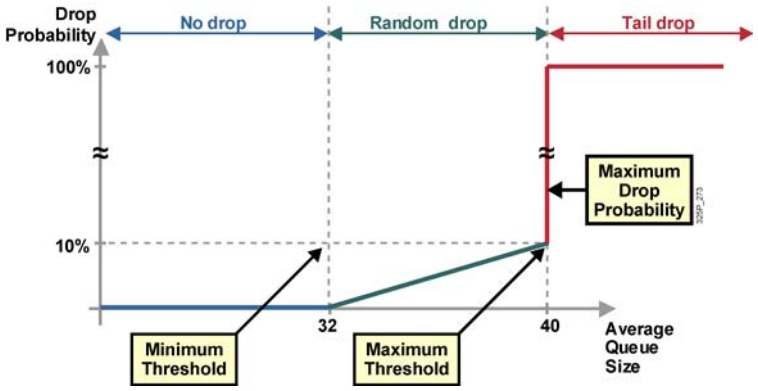


- Multiple TCP sessions start at different times.
- TCP window sizes are increased.
- Tail drops cause many packets of many sessions to be dropped at the same time.
- TCP sessions restart at the same time (synchronized).

Random Early Detection (RED)

- Tail drop can be avoided if congestion is prevented.
- RED is a mechanism that randomly drops packets before a queue is full.
- RED increases drop rate as the average queue size increases.
- RED result:
  - TCP sessions slow to the approximate rate of output-link bandwidth.
  - Average queue size is small (much less than the maximum queue size).
  - TCP sessions are desynchronized by random drops.

RED Drop Profiles



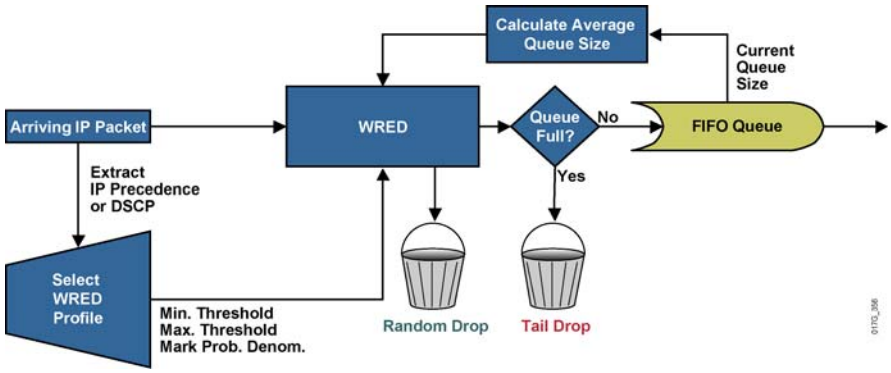
RED Modes

- RED has three modes:
  - No drop: When the average queue size is between 0 and the minimum threshold
  - Random drop: When the average queue size is between the minimum and the maximum threshold
  - Full drop (tail drop): When the average queue size is above the maximum threshold
- Random drop should prevent congestion (prevent tail drops).

Weighted Random Early Detection (WRED)

- WRED can use multiple RED profiles.
- Each profile is identified by:
  - Minimum threshold
  - Maximum threshold
  - Mark probability denominator
- WRED profile selection is based on:
  - IP precedence (8 profiles)
  - DSCP (64 profiles)
- WRED drops less important packets more aggressively than more important packets.
- WRED can be applied at the interface, VC, or class level.

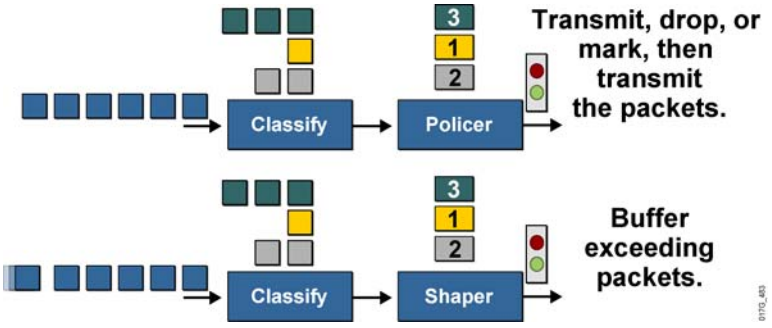
WRED Building Blocks



Traffic Conditioners

- Policing
  - Limits bandwidth by discarding traffic.
  - Can re-mark excess traffic and attempt to send.
  - Should be used on higher-speed interfaces.
  - Can be applied inbound or outbound.
- Shaping
  - Limits excess traffic by buffering.
  - Buffering can lead to a delay.
  - Recommended for slower-speed interfaces.
  - Cannot re-mark traffic.
  - Can only be applied in the outbound direction.

Traffic Policing and Shaping Overview



- These mechanisms must classify packets before policing or shaping the traffic rate.
- Traffic policing typically drops or marks excess traffic to stay within a traffic rate limit.
- Traffic shaping queues excess packets to stay within the desired traffic rate.

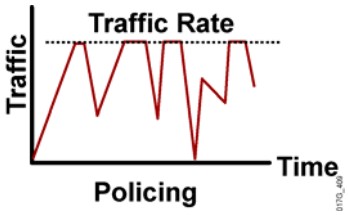
Why Use Policing?

- To limit access to resources when high-speed access is used but not desired (subrate access)
- To limit the traffic rate of certain applications or traffic classes
- To mark down (recolor) exceeding traffic at Layer 2 or Layer 3

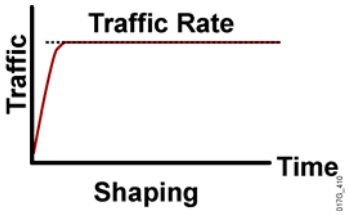
Why Use Shaping?

- To prevent and manage congestion in ATM, Frame Relay, and Metro Ethernet networks, where asymmetric bandwidths are used along the traffic path
- To regulate the sending traffic rate to match the subscribed (committed) rate in ATM, Frame Relay, or Metro Ethernet networks
- To implement shaping at the network edge

Policing Versus Shaping

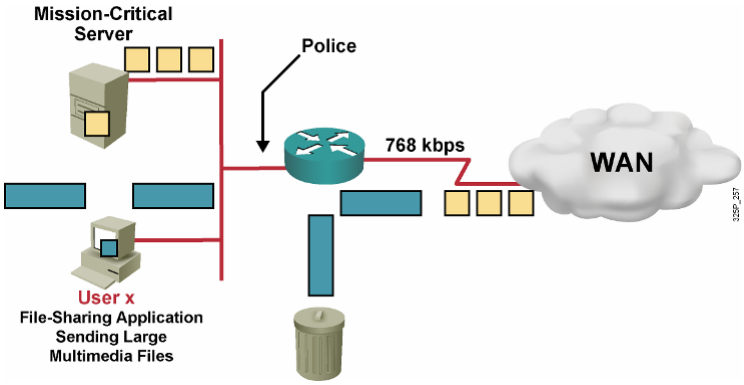


- Incoming and outgoing directions.
- Out-of-profile packets are dropped.
- Dropping causes TCP retransmits.
- Policing supports packet marking or re-marking.



- Outgoing direction only.
- Out-of-profile packets are queued until a buffer gets full.
- Buffering minimizes TCP retransmits.
- Marking or re-marking not supported.
- Shaping supports interaction with Frame Relay congestion indication.

Traffic Policing Example

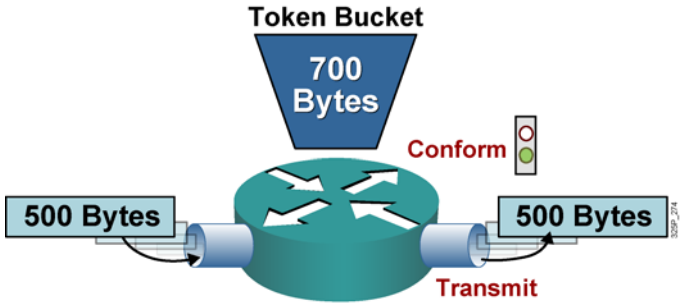


- Do not rate-limit traffic from mission-critical server.
- Rate-limit file-sharing application traffic to 56 kbps.

Token Bucket

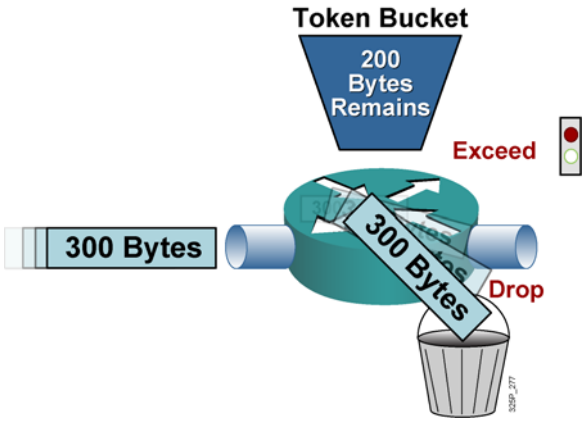
- Mathematical model used by routers and switches to regulate traffic flow.
- Tokens represent permission to send a number of bits into the network.
- Tokens are put into the bucket at a certain rate by IOS.
- Token bucket holds tokens.
- Tokens are removed from the bucket when packets are forwarded.
- If there are not enough tokens in the bucket to send the packet, traffic conditioning is invoked (shaping or policing).

Single Token Bucket



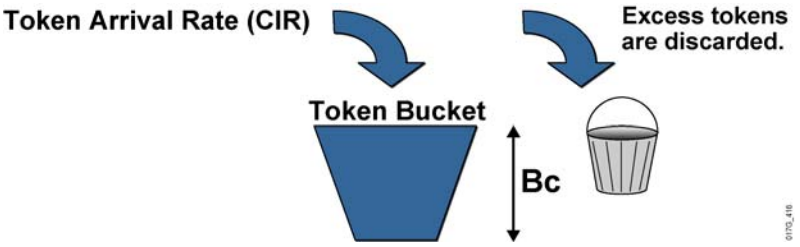
- If sufficient tokens are available (conform action):
  - Tokens equivalent to the packet size are removed from the bucket.
  - The packet is transmitted.

Single Token Bucket Exceed Action



- If sufficient tokens are not available (exceed action):
  - Drop (or mark) the packet.

Single Token Bucket Class-Based Policing



Bc is normal burst size.
Tc is the time interval.
CIR is the committed information rate.
CIR = Bc / Tc

Applying Rate Limiting