IOS-XE commands are in red boxes/font
IOS-XR commands are in blue boxes/font

Congestion Management (Queueing)

What is congestion?

Congestion occurs any time an interface is presented with more traffic than it is able to transmit

It typically occurs in the following places:
> Speed Mismatches
> Aggregation points
> Combination of two or more traffic streaming

Queueing helps by having routers store packets in memory while others are being proceeded.

If the outgoing interface is busy the packet will be placed into a queue. Queueing can be done inbound or outbound.

Ultization & Scheduling

A queue is essentially just a series of pointers to memory buffers.

The following mechanisms utilise queues:

Classification: Decides what packet goes into what queue
Dropping: Is there room in the queue? No? Drop it.
Number of queues: The total number that the mechanism supports.
Reordering: within a queue.

A **scheduler** is also used. This mechanism essentially chooses the next packet for transmission from one of several queues.

IMPORTANT: The size of each packet does not affect the length of the queues or how many packets it can hold. Queues don't hold packets themselves. They just hold the pointers.

Queue Types

IOS-XE

Hardware

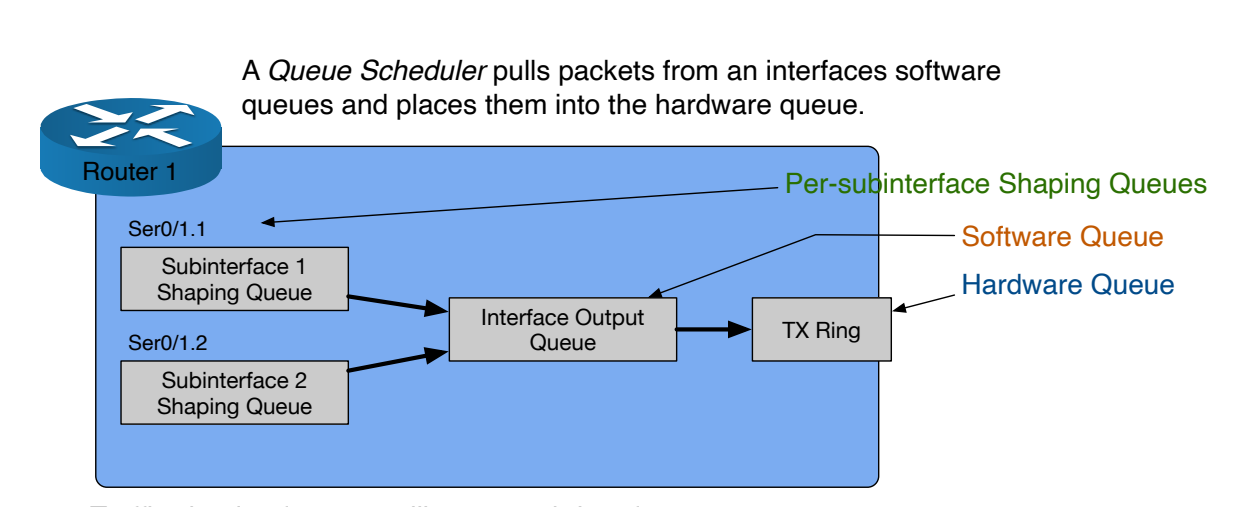
Hardware queue (also called TX queue)
> Is a basic FIFO queue on a physical interface.
> Depending on the hardware, the length may be configurable.
> Packets in hardware queue cannot be reordered.
> Long hardware queue result in long delays and is counter productive for QoS.
> Linked to ASICS.

Software

Software Queues
> All IOS queueing tools operate on software queues.
> Packets are basically placed in the software queues if the hardware queue is not free.
> An interface can have multiple software queues.
> Packets are assigned to queues using a variety of mechanisms.

Queue Scheduler and Sub interfaces

A *Queue Scheduler* pulls packets from an interfaces software queues and places them into the hardware queue.


Router 1
Ser0/1.1
Subinterface 1 Shaping Queue
Ser0/1.2
Subinterface 2 Shaping Queue
Interface Output Queue
TX Ring
Per-subinterface Shaping Queues
Software Queue
Hardware Queue

Traffic shaping features will create sub-interface queues where necessary. There queues will in turn be drained into the main interface output queue. Queueing mechanisms can manage sub-interface queues just like main interface queues.

IOS-XR

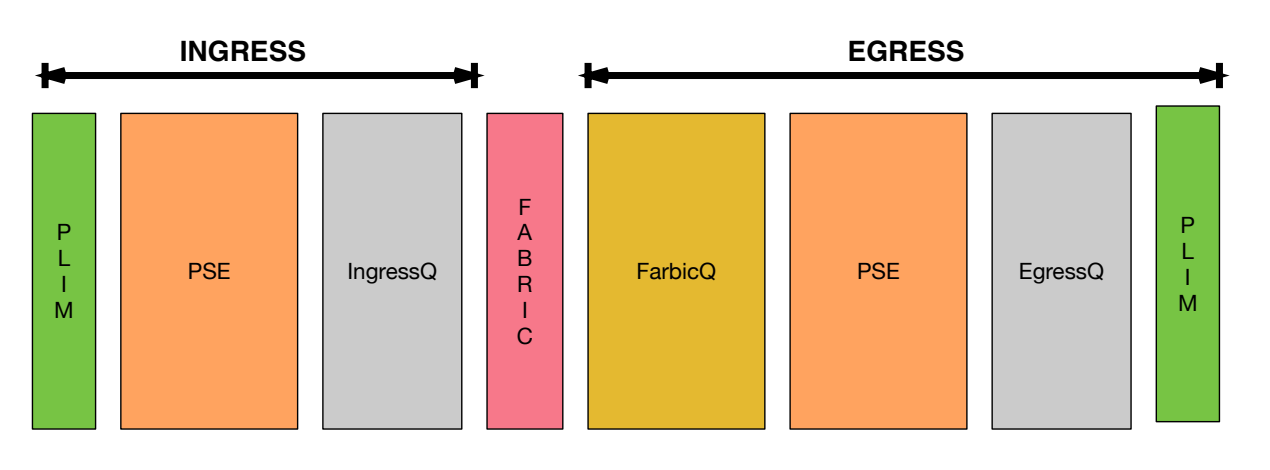
Distributed

Distributed queueing: The concept of software and hardware queues as seen in IOS-XE is extended across interface modules, and the router fabric.

Queueing is supported by ASICs and can be done on ingress/egress ***and*** across the internal fabric. Each stage is configured separately.

Cisco IOS XR uses the concept of *dynamic queue thresholds* to allocate the queueing space on demand.

IOS-XR Forwarding Architecture

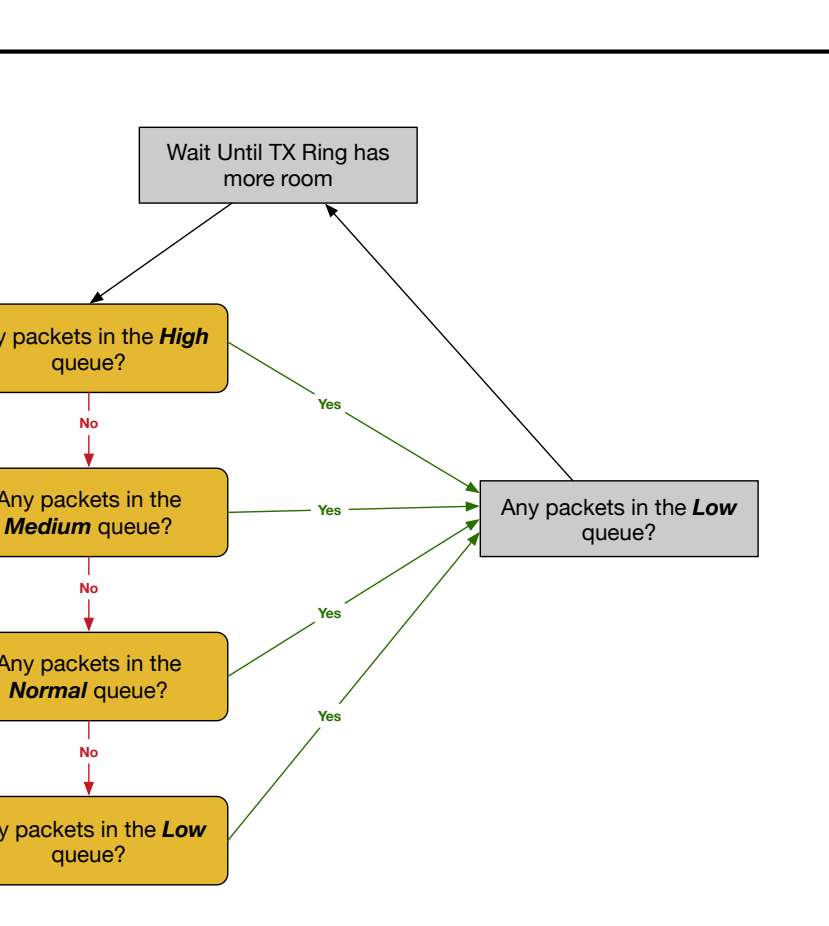
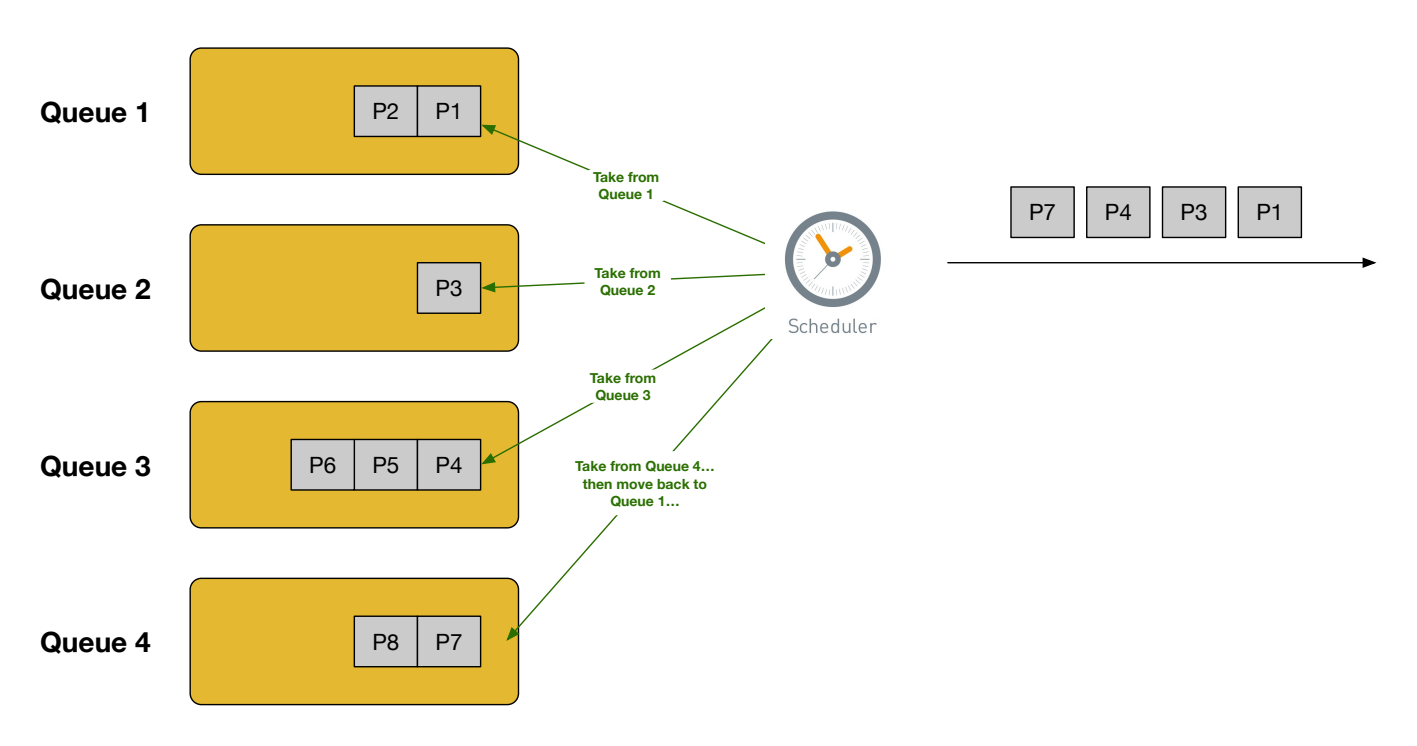
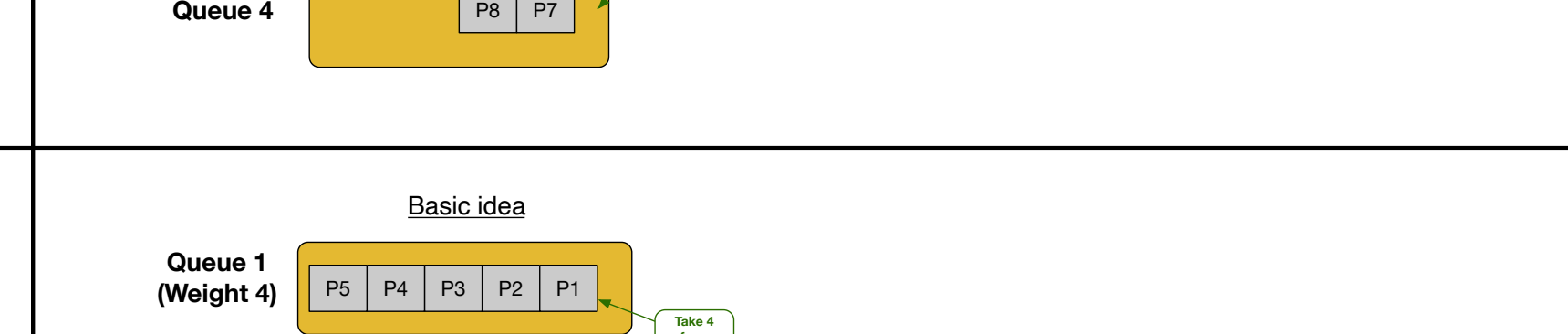
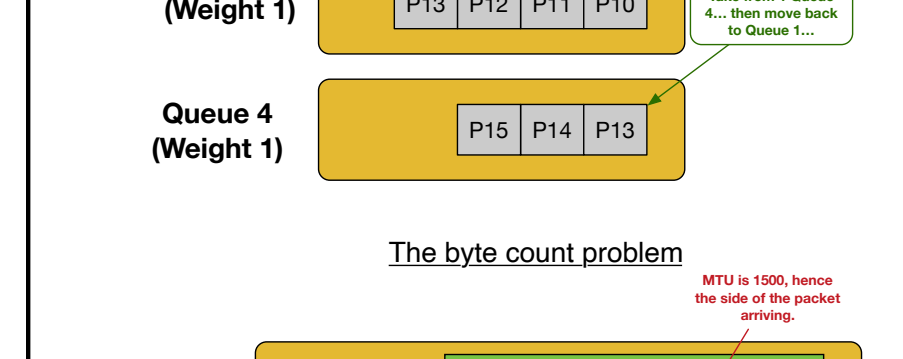
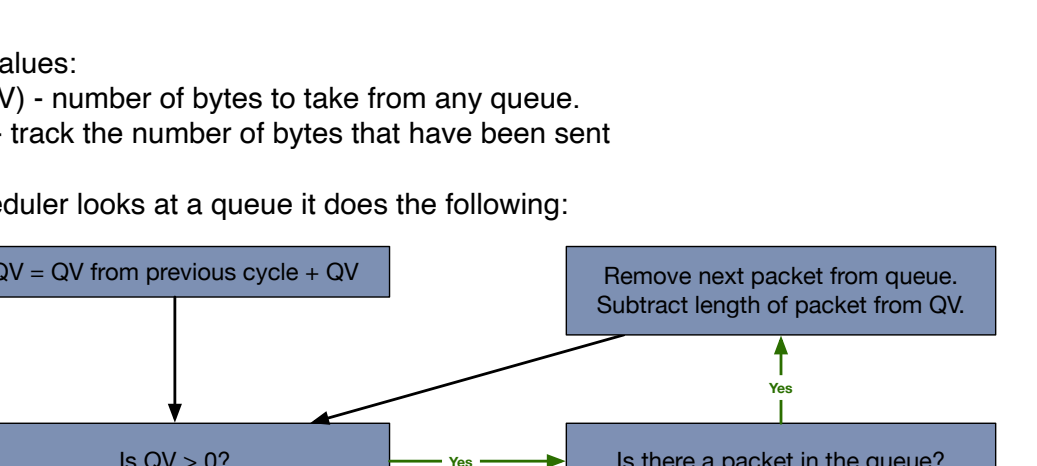
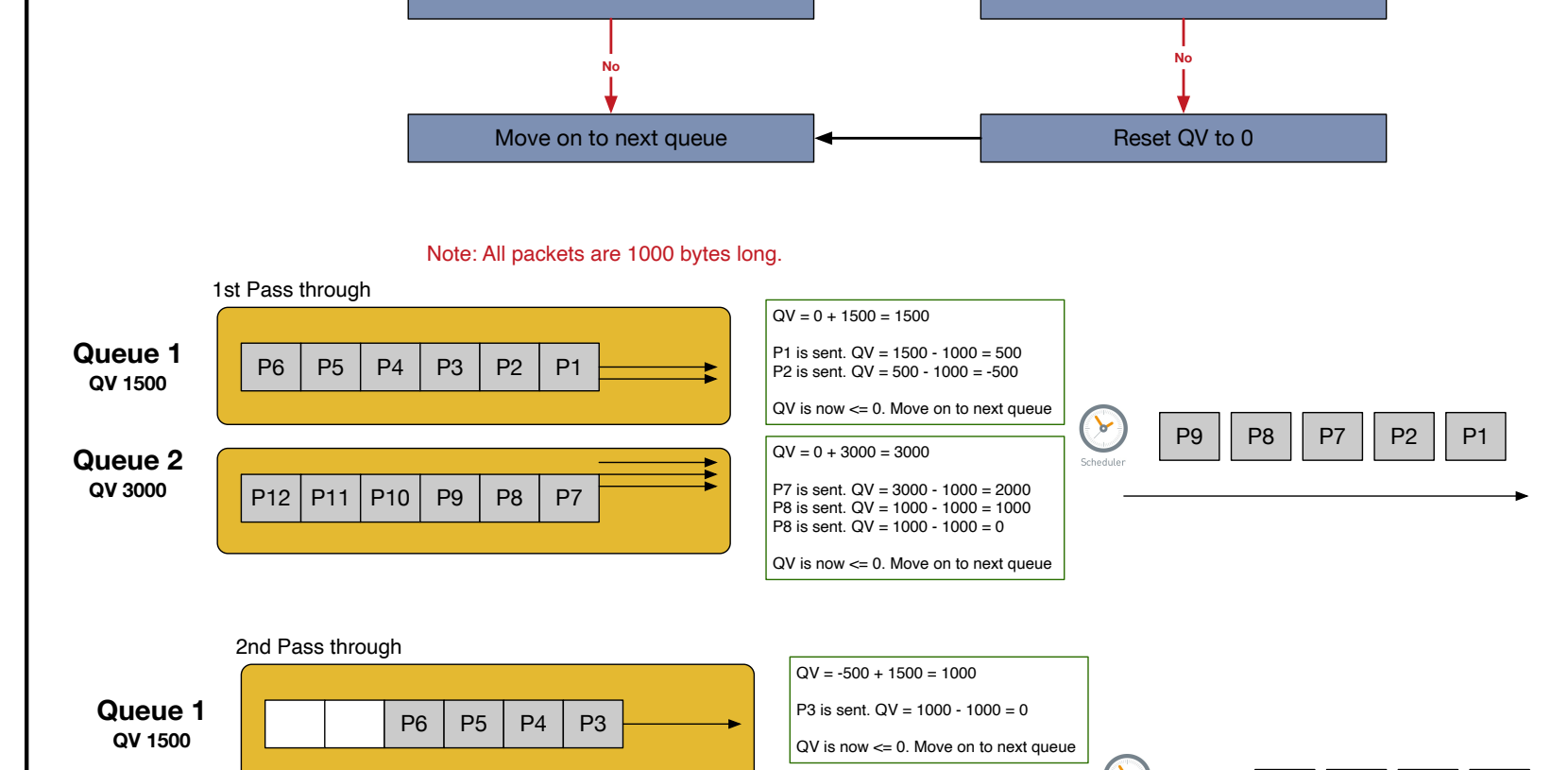
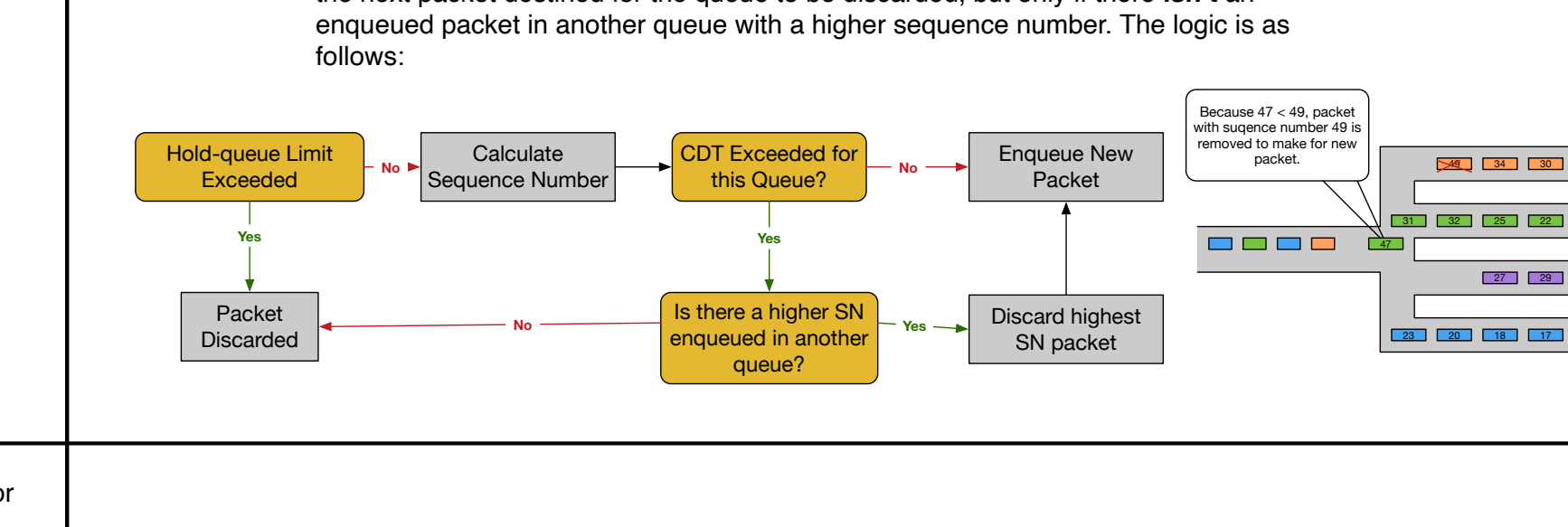
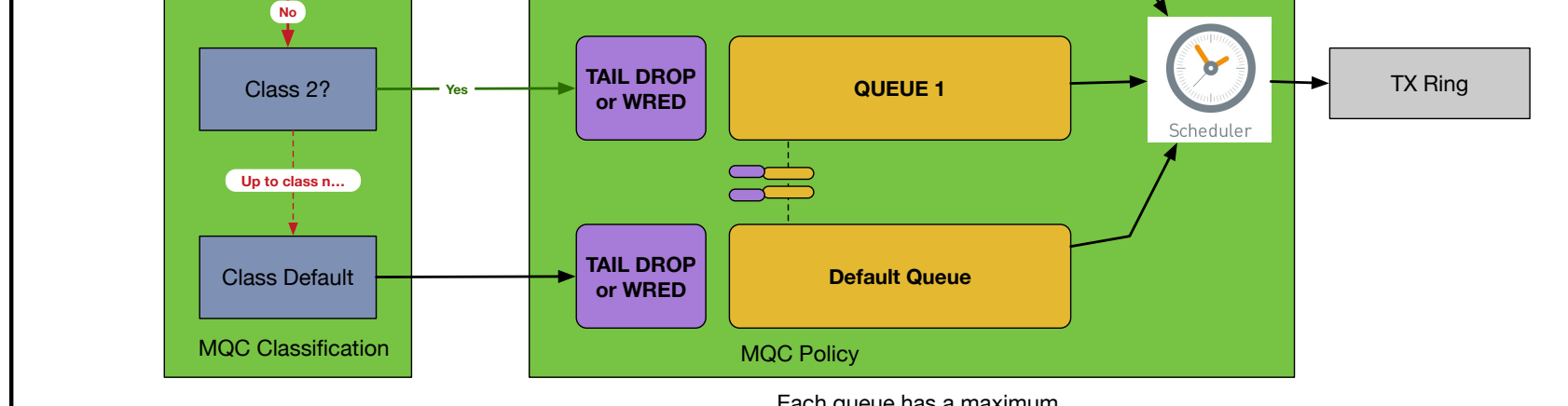
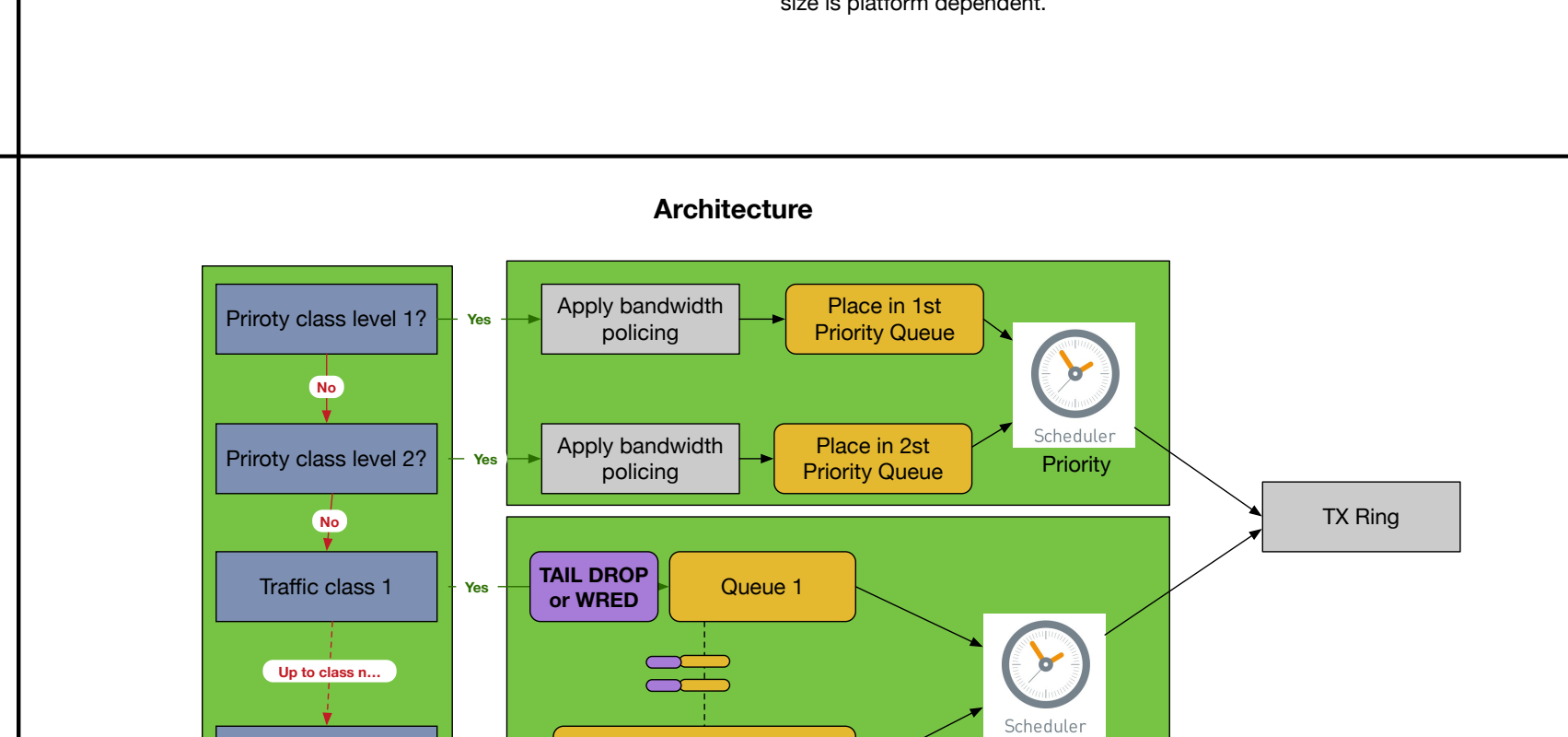
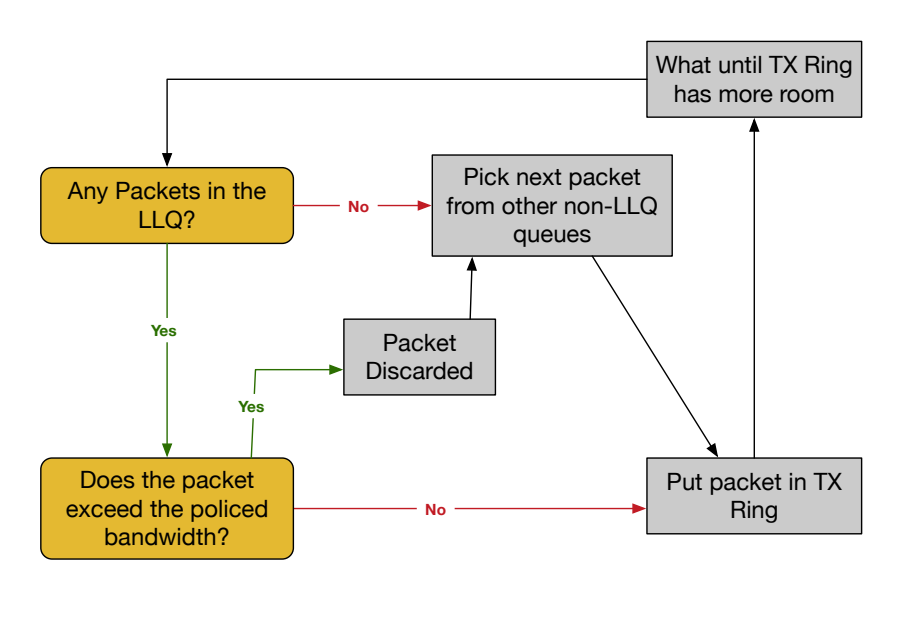
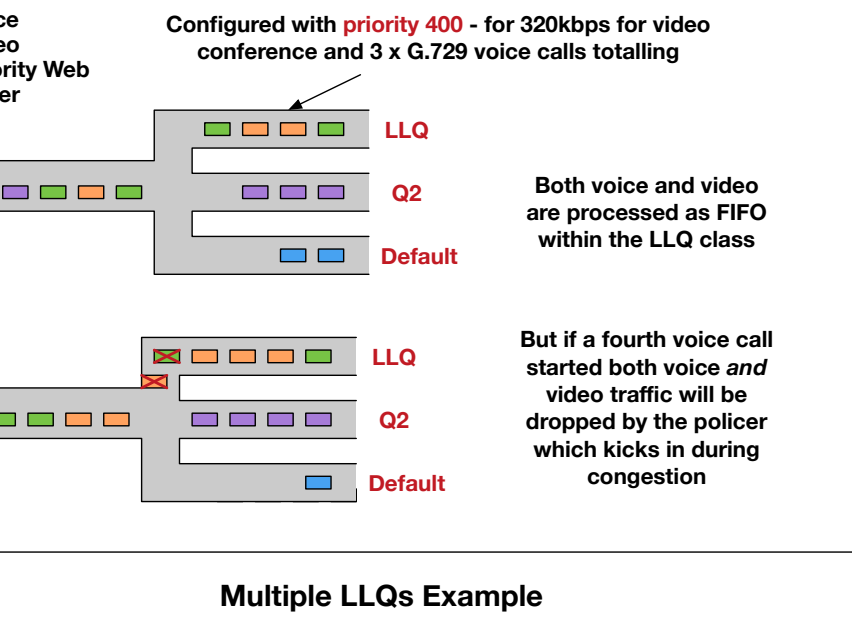
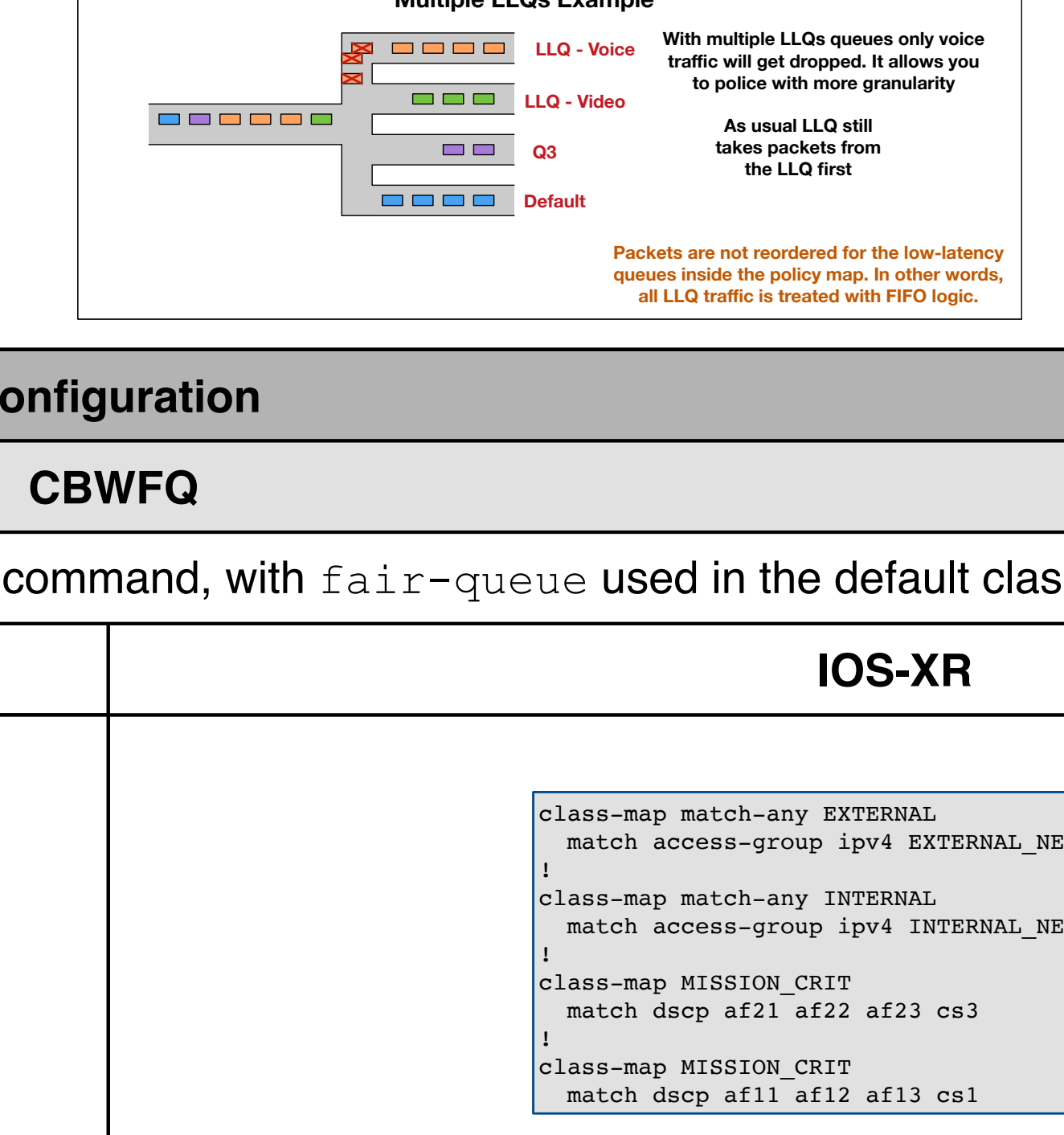

INGRESS
EGRESS
PLIM
PSE
IngressQ
FABRIC
FarbicQ
PSE
EgressQ
PLIM

PLIMs (Physical Layer Interface Modules) provides the interface circuitry

PSEs (Packet Switching Engines) performs packet lookup and processing

Fabric provides the communication between the line cards

Each of these 3 blocks has its own queueing architecture

Queueing Method	Main features	Queuing Algorithm	Defaults
FIFO (First In First Out)	First In First Out. This is affects drop delay and jitter. It does not classify or reorder the traffic. Increasing a FIFO queue will reduce tail drop but increase delay and jitter. **IMPORTANT:** Technically *all* individual queues are FIFO - it is other qeueing algorithms (applying the utilisation and scheduling features) that are used to make each queuing method unique	None. Packets are processed in the order in which they arrive. The hardware queue (or Transmit Ring) is a FIFO queue. To see if FIFO is in use, look for `Queueing Strategy: fifo` in `show interface x` output	If a queuing method is enabled on an interface the hardware queue size is made smaller to minimise its impact. By default. Cisco IOS uses WFQ on serial interfaces running E1 or slower - so to enable FIFO you'll need to turn this off using the `no fair-queue` IOS-XE command. Adjust the queue length (if possible) using the `hold-queue x out` command
PQ (Priority Queuing)	Packets are classified into a maximum of 4 queues. High queues will always be processed before lower priority queue, as long as they contain packets. High priority queues can easily starve lower priority queues.	 Wait Until TX Ring has more room Any packets in the *High* queue? No Any packets in the *Medium* queue? No Any packets in the *Normal* queue? No Any packets in the *Low* queue? Yes Yes Yes Yes Any packets in the *Low* queue?	Maximum of 4 queues on IOS platforms
RR (Round Robin)	Uses multiple queues. The scheduler takes one packet from each queue then moves onto the next. If all packets are the same size then all queues share equal bandwidth. You **cannot** prioritise traffic. So whilst it will not starve any queue, no queue will get special treatment.	 Queue 1 P2 P1 Queue 2 P3 Queue 3 P6 P5 P4 Queue 4 P8 P7 Take from Queue 1 Take from Queue 2 Take from Queue 3 Take from Queue 4... then move back to Queue 1... Scheduler P7 P4 P3 P1	N/A
WRR (Round Robin)	Same principle at Round Robin, but each queue has a priority or weight assigned to it. It was developed as a way to provide prioritisation to Round Robin. Higher weighted queues have more packets removed from them before moving on to the next. **IMPORTANT:** Some implementations of WRR allow each queue to send a number of ***bytes*** rather than a number of ***packets***.	Basic idea  Queue 1 (Weight 4) P5 P4 P3 P2 P1 Queue 2 (Weight 2) P9 P8 P7 P6 Queue 3 (Weight 1) P13 P12 P11 P10 Queue 4 (Weight 1) P15 P14 P13 Take 4 from Queue 1 Take 2 from Queue 2 Take 1 from Queue 3 Take from 1 Queue 4... then move back to Queue 1... Scheduler P13 P10 P7 P6 P4 P3 P2 P1These are removed based on *packet count*. But this is not what happen. In reality byte count is used. If we look at byte count, where some packets are of different sizes, we run into trouble… The byte count problem  MTU is 1500, hence the side of the packet arriving. Queue 1 (Byte count 3000) 1500bytes 1499bytes 1500bytes 3000 byte countIn the above imaage the **weighted byte count threshold** for this queue is 3000. WRR will sent ***all three packets*** in the queue since they *just* fit into 3000s - even though the third packet only has 1 byte in range If the ratio between the MTU and byte count is too small, WRR will not allocate bandwidth correctly. If it is too high, it will cause delays.	N/A
DRR (Deficit Robin)	Similar to WRR - works on bytes being taken from the queue, but also takes into account excess packets taken from a queue on any given cycles. It then subtracts this *excess* from that queues allocation on the next cycle.	DRR defined two values: Quantum Value (QV) - number of bytes to take from any queue. Deficit Value (DV) - track the number of bytes that have been sent Each time the scheduler looks at a queue it does the following:  QV = QV from previous cycle + QV Remove next packet from queue. Subtract length of packet from QV. Is QV > 0? Yes Is there a packet in the queue? Yes No No Move on to next queue Reset QV to 0Note: All packets are 1000 bytes long.  1st Pass through Queue 1 QV 1500 P6 P5 P4 P3 P2 P1 Queue 2 QV 3000 P12 P11 P10 P9 P8 P7 QV = 0 + 1500 = 1500 P1 is sent. QV = 1500 - 1000 = 500 P2 is sent. QV = 500 - 1000 = -500 QV is now <= 0. Move on to next queue QV = 0 + 3000 = 3000 P7 is sent. QV = 3000 - 1000 = 2000 P8 is sent. QV = 1000 - 1000 = 1000 P8 is sent. QV = 1000 - 1000 = 0 QV is now <= 0. Move on to next queue Scheduler P9 P8 P7 P2 P1 2nd Pass through Queue 1 QV 1500 P6 P5 P4 P3 Queue 2 QV 3000 P12 P11 P10 QV = -500 + 1500 = 1000 P3 is sent. QV = 1000 - 1000 = 0 QV is now <= 0. Move on to next queue QV = 0 + 3000 = 3000 P10 is sent. QV = 3000 - 1000 = 2000 P11 is sent. QV = 1000 - 1000 = 1000 P12 is sent. QV = 1000 - 1000 = 0 QV is now <= 0. Move on to next queue Scheduler P12 P11 P10 P3Guaranteed Bandwidth (over time) Based on the following formula: QV_for_queue_x / sum_of_all_QVs The QV for each queue should not be smaller that the MTU to ensure that non-empty queues are always serviced.	Works on IOS-XR and Cisco 12000 Series Routers
MDRR (Modified Deficit Round Robin)	Same as DRR but adds a special low-latency queue that operates in one of two modes: **Strict Mode:** If there is ever a packet in the strict priority queue it will be serviced. **Alternate Mode:** Scheduler alternates between servicing the low-latency queue and any other queue.	With strict mode the priority queue can starve other queues. With alternate mode, how much more the priority queue is services, depends on how many queues there are overall.	Only works on GSR12000 (other queuing methods will not work on GSRs)
WFQ (Weighted Fair Queuing)	Each flow is allocated is own queue. A flow consists of packets that share the following in common: > Source IP > Destination IP > Source Port > Destination Port > Transport Layer Protocol > IP Precendence As packets are added to queues they are assigned a sequence number. The Scheduler will remove packets for the WFQ queues starting at the lowest number. The higher the IP Precedence, the lower the sequence number. In this way, WFQ favours low-volume, high-precedence flows over large-volume, low-precedence flows. WFQ is also called Flow-Based Weight Fair Queuing. It is all automatic and does not allow for customisation. A flow (despite being part of a continuous TCP connection) will be removed after a period of inactivity. So a single TCP connection, for example to a website, could comprise multiple flows. The goal of WFQ is to provide fairness among all queues, with unused bandwidth being available to other queues. High IP Precedence will get more bandwidth based on ratio between each precedence. Precendence 7 gets 8 times more bandwidth that Precedence 0 because (7+1)/(0+1) = 8. Special queues: WFQ keeps 8 hidden queues for traffic generated by router with very low weight. WFQ also creates a queue for each RSVP reserved flow.	When Hardware queue is has a free slot the schedule removes a the packet with the ***lowest*** sequence number. The sequence number is assigned when the packet is placed in a queue. The sequence number is calculated as follows **Seq_Num** = **Prev_Seq_Num** (for that queue) + (**Weight** * **New_Packet_Length**) **Weight** = 32,384 / (IP_Prec + 1)  High IP Prec. Low Length Low Weight Low previous Seq. Num. Low Seq. Num. Low Wait Time (serviced sooner)Lower length, weight or previous sequence number results in a lower sequence number which results in quicker service When a new flow (and thus a new queue) is started, the previous sequence number is the sequence number of the last packet sent to the TX Ring (Hardware Queue). The ***hold-queue limit*** is the absolute limit on the number of packets enqueued amongst all queues. Each queue has a congestive discard threshold (CDT) which, if exceeded, will cause the next packet destined for the queue to be discarded, but only if there ***isn't*** an enqueued packet in another queue with a higher sequence number. The logic is as follows:  Hold-queue Limit Exceeded No Calculate Sequence Number CDT Exceeded for this Queue? No Enqueue New Packet Yes Is there a higher SN enqueued in another queue? Yes Discard highest SN packet No Yes Packet Discarded Because 47 < 49, packet with surpence number 49 is removed to make for new packet.	**On IOS-XE** You can have, 4096 queues ***per interface*** - this is configurable but must be a power of 2 between 16 and 4096. WFQ is enabled by default The ***hold-queue limit*** default is 1000. CDT can be configured to a value between 1 and 4096 (inclusive) Enabled by default on all serial interfaces with bandwidth set at T1 and E1 speeds or below. WFQ uses modified tail drop.
CBWFQ (Class Based Weighted Fair Queuing)	CBWFQ extends the WFQ functionality to allow support for user-defined traffic classes. A queue is reserved for each class. Each class gets a minimum bandwidth that will be guaranteed during congestion. This is done by assigning weights. Weights can be assigned using the following criteria: > **Bandwidth** - a specified fixed amount > **Percentage of bandwidth** available on interface - the total bandwidth on the interface should reflect actual interface speed but can be adjusted using the ***bandwidth*** interface command. > **Percentage of remaining bandwidth** - typically used together with the ***bandwidth*** command at the parent level. A single service policy cannot have mixed types of weights at the same level. **IMPORTANT:** On egress, the actual bandwidth of the interface is determined to be the Layer 2 capacity excluding CRC. These have to be included because they are applied per packet, and the system cannot predict how many packets of a particular packet size are being sent out.	 Incoming packet P1 Does it match the criteria for Class 1? No Class 2? Up to class n... Class Default MQC Classification Yes Yes CBWFQ Policy TAIL DROP or WRED QUEUE 1 TAIL DROP or WRED QUEUE 1 TAIL DROP or WRED Default Queue Scheduler TX Ring MQC PolicyEach queue has a maximum number it can hold. The maximum size is platform dependent.	**IOS-XE** CBWFQ has 64 available queues. **IOS-XR** Uses dynamic thresholds for its queues. For both IOS and XR tail drop is the default drop mechanism but WRED can be configured
LLQ (Low Latency Queuing)	LLQ is basically CBWFQ with a strict priority queue for real time traffic that is delay sensitive. Priority queues are serviced first. LLQ enables the use of a strict priority queue with CBWFQ at the class level. This is done using the ***priority*** command within a class. IOS-XR uses two strict priority queues. However in order to avoid starvation of the lower queues the priority queues are policed to ensure LLQ can still provide bandwidth guarantees to the non-priority queues. You can give more than one class priority status.When multiple classes within a single policy map are configured as priority classes, all traffic from these classes is enqueued to a single strict priority queue. **IMPORTANT:** The LLQ is only policed when there is congestion on the link and the software queuing is engaged.	Architecture  Priority class level 1? Yes No Priority class level 2? Yes No Traffic class 1 Yes Up to class n... Class Default? Apply bandwidth policing Place in 1st Priority Queue Apply bandwidth policing Place in 2st Priority Queue Scheduler Priority TAIL DROP or WRED Queue 1 Default Queue Scheduler CBWFQ TX RingDecision Flow Diagram  What until TX Ring has more room Any Packets in the LLQ? No Pick next packet from other non-LLQ queues Yes Packet Discarded Yes Does the packet exceed the policed bandwidth? No Put packet in TX RingSingle LLQ Example Multiple LLQs Example  Voice Video Priority Web Other Configured with priority 400 - for 320kbps for video conference and 3 x G.729 voice calls totalling LLQ Q2 Default Both voice and video are processed as FIFO within the LLQ class LLQ Q2 Default But if a fourth voice call started both voice and video traffic will be dropped by the policer which kicks in during congestion LLQ - Voice LLQ - Video Q3 Default With multiple LLQs queues only voice traffic will get dropped. It allows you to police with more granularity As usual LLQ still takes packets from the LLQ first Packets are not reordered for the low-latency queues inside the policy map. In other words, all LLQ traffic is treated with FIFO logic.	Default polling thresholds Default Maximum threshold for priority queues is 10ms Default maximum threshold for regular queues is 100ms

Configuration

CBWFQ

Typically configured using the `bandwidth` command, with `fair-queue` used in the default class

IOS-XE

```
Router#show running-config class-map match-all VOIP_RTP
  match ip rtp 16384 16383
!
class-map match-all HIGH_PRIORITY
  match ip dscp ef
!
!
policy-map VOIP_AND_BE
  class VOIP_RTP
   set ip dscp 46
  class class-default
   set ip dscp 0
!
policy-map DSCP_QUEUE
  class HIGH_PRIORITY
   bandwidth 58
   queue-limit 30
 class class-default
  random-detect dscp-based
  fair-queue
!
interface Ethernet0/0
 description LAN link
 ip address 192.168.4.253 255.255.255.0
 service-policy input VOIP_AND_BE
!
interface serial0/1
 clock rate 128000
 bandwidth 128
 service-policy output DSCP_QUEUE
```

```
Router#show policy-map interface serial 0/1
 Serial0/1
  Service-policy output: DSCP_QUEUE
Class-map: HIGH_PRIORITY (match-all)
  8654 packets, 553856 bytes
  30 second offered rate 51000 bps, drop rate 0 bps
  Match: ip dscp ef
  Weighted Fair Queueing
    Output Queue: Conversation 41
    Bandwidth 58 (kbps) Max Threshold 30 (packets)
    (pkts matched/bytes matched) 8441/530238
    (depth/total drops/no-buffer drops) 8/0/0
Class-map: class-default (match-any)
  653 packets, 734357 bytes
  30 second offered rate 73000 bps, drop rate 0 bps
  Match: any
  Weighted Fair Queueing
    Flow Based Fair Queueing
    Maximum Number of Hashed Queues 32
    (total queued/total drops/no-buffer drops) 14/0/0
exponential weight: 9
```

Adjusting bandwidth percent

If `bandwidth percent x` is used, the percentage taken is the `bandwidth` specified under the interface.

For example:
`interface gigabitEthernet0/1`
`bandwidth 128`

Traffic matching `bandwidth percentage 50` in a service policy applied under this interface would receive 64 kbps.

When configuring the `bandwidth` command under a policy map, use *either* `bandwidth` or `bandwidth percent x` but not a combination.

```
Router(config-cmap)#policy-map EXPL_BW
Router(config-cmap)#class CLASS_A
Router(config-cmap-c)#bandwidth 64
Router(config-cmap-c)#class CLASS_B
Router(config-cmap-c)#bandwidth percent 25
All class with bandwidth should consistent
units
Router(config-cmap-c)#bandwidth 32
```

The `max-reserved-bandwidth` command defaults to 75% (this is the equivalent of putting the `max-reserved-bandwidth 75` command on the interface). This means 25% is allocated to the class-default queue.

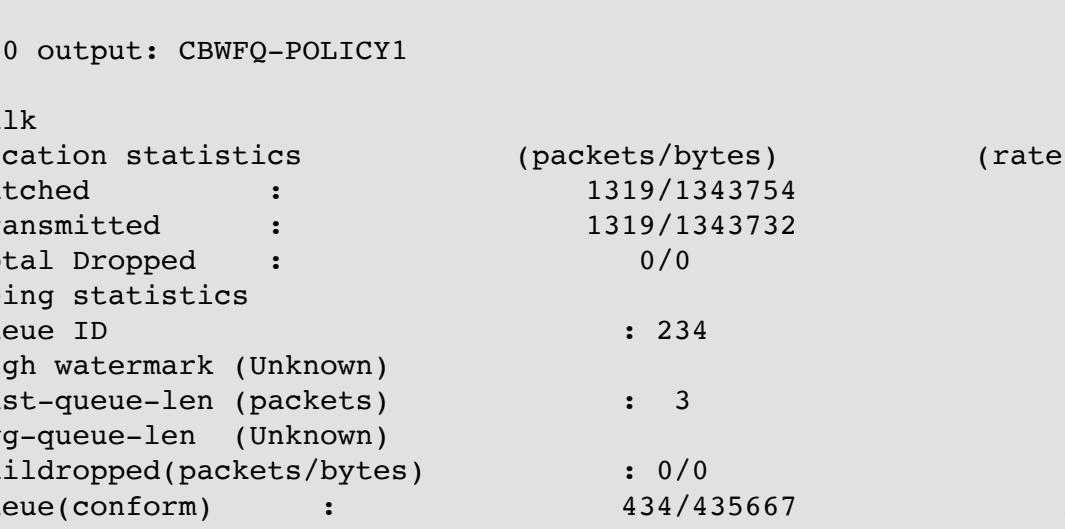
If you attempt to apply a service-policy to an interface that will use more that 75% of the available bandwidth in the `bandwidth` command you will get an error.

IOS-XR

```
class-map match-any EXTERNAL
  match access-group ipv4 EXTERNAL_NETS
!
class-map match-any INTERNAL
  match access-group ipv4 INTERNAL_NETS
!
class-map MISSION_CRIT
  match dscp af21 af22 af23 cs3
!
class-map MISSION_CRIT
  match dscp af11 af12 af13 cs1
```

```
policy-map CBWFQ_CHILD
  class INTERNAL
    bandwidth remaining percent 80
  !
  class EXTERNAL
    bandwidth remaining percent 20
!
policy-map CBWFQ_PARENT
  class MISSION_CRIT
    service-policy CBWFQ_CHILD
    bandwidth percent 30
  !
  class BULK
    service-policy CBWFQ_CHILD
    bandwidth percent 40
!
```

```
interface GigabitEthernet0/0/0/0
    service-policy output CBWFQ_PARENT
```


CBWFQ_PARENT
MISSION_CRIT
30%
BULK
40%
class-default
30% for mission critical and 40% for bulk after class-default minimum is accounted for.
CBWFQ_CHILD
INTERNAL
EXTERNAL
24% to internal mission critical (80% of 30%)
6% to external mission critical (20% of 30%)
CBWFQ_CHILD
INTERNAL
EXTERNAL
32% to internal bulk (80% of 40%)
8% to external bulk (20% of 40%)

On a Cisco IOS-XR CBWFQ implementation, the algorithm used in dequeuing the packets from each CBWFQ queue is based on MDRR instead of WFQ.

```
RP/0/RSP0/CPU0:Router# show policy-map interface Gi0/0/0/0
Gi0/0/0/0 direction input: Service Policy not installed

Gi0/0/0/0 output: CBWFQ-POLICY1

Class Bulk
Classification statistics           (packets/bytes)          (rate - kbps)
      Matched          :               1319/1343754                154
      Transmitted      :               1319/1343732                154
      Total Dropped    :                  0/0                       0
   Queueing statistics
      Queue ID                            : 234
      High watermark (Unknown)
      Inst-queue-len (packets)            :  3
      Avg-queue-len  (Unknown)
      Taildropped(packets/bytes)          : 0/0
      Queue(conform)      :               434/435667               53
      Queue(exceed)       :               876/867435               89
      RED random drops(packets/bytes)     : 0/0
<snip>
```

Note The "exceed" in this case does **not** equate to a packet drop, but rather a packet that is above the CIR rate on that queue.

These are the general commands for configuring CBWFQ on both IOS and XR. Random detect refers to the congestion avoidance mechanism.

Command	Mode and Function
`bandwidth [bandwidth-kpbs | percent percentage]`	Class subcommand; sets literal or percentage of bandwidth for the class. Must use either actual bandwidth or percent on all classes in a single policy map.
`bandwidth [remaining percent percentage]`	Class subcommand; sets percentage of remaining bandwidth for the class. Remaining bandwidth is **bandwidth** minus other explicit bandwidth reservations.
`queue-limit queue-limit`	Class subcommand; sets the maximum number of packets in the queue for that class. This can be ms on XR.
`fair-queue [queue-limit queue-value]`	Class subcommand; enables WFQ in the class (class-default only)
`random-detect dscp dscp-value min-thresh max-thresh [mpd]`	Class subcommand; enables DSCP-based WRED in the class
`random-detect precedence prec-value min-thresh max-thresh mpd`	Class subcommand; enables precendence-based WRED in the class
`max-reserved-bandwidth percent`	Interface subcommand; defines the percent of link bandwidth that can be reserved for CBWFQ queues besides class-default.

LLQ

Typically configured with the `priority` command using the `police rate` command to specify its rate

IOS-XE

IOS-XE configuration is the same as CBWFQ but uses `priority` command instead of `bandwidth`. The full syntax is as follows:

`priority { <bandwidth-kbps> | percent <percent>} [burst]`

For this class, LLQ will be enabled, bandwidth is reserved and the policing function (with default burst of 20 percent if not specified) is enabled.This is, in effect, setting a guaranteed minimum bandwidth which, because of how LLQ works (using a policer to discard excess), is also the maximum bandwidth. Bursting works as normal for policing.

The `bandwidth remaining percent` command can be used in conjunction with explicit `bandwidth` statement (as opposed to `bandwidth percent`)

If `bandwidth remaining percent` is used with LLQ, the explicitly defined amount of bandwidth for the priory queue is removed first, then the `max-reservable-bandwidth` is taken for the class-default queue. Finally the `bandwidth remaining percent` commands reserved the relevant percentage of what is left after the above two steps are complete.

For example:

```
policy-map REMAINING
  class CLASS_1
    priority 55
  class CLASS_2
    bandwidth remaining percent 25
  class CLASS_3
    bandwidth remaining percent 35
  class CLASS_4
    bandwidth remaining percent 40
!
interface Serial0/1
  bandwidth 256
  service-policy output REMAINING
```

The interface has 256 kbps of available bandwidth.
Class 1 receives 55kbps.
The class-default queue receives (by default) 25% of the of the interface bandwidth or 64kbps.

256 - 55 - (256 * 0.25) = 256 - 55 - 64 = 137kbps are remaining

Class 2 receives 137 * 0.25 = 24kbps
Class 3 receives 137 * 0.35 = 48kbps
Class 4 receives 137 * 0.40 = 55kbps

```
Router#show policy-map DSCP_QUEUE
    Policy Map DSCP_QUEUE
    Class scp-ef
      Weighted Fair Queueing
        Strict Priority
        Bandwidth 58 (kbps) Burst 1450
(Bytes)
<snip>
```

Shows LLQ

The default burst size is equal to 200ms of traffic.
58,000bps * 0.2s = 11,600 bits = 1,450 bytes

IOS-XR

Cisco IOS-XR uses the `priority [1|2]` to assign a priority queue.

```
policy-map LLQ_POLICY
  class VOICE_INT
    priority level 1
    police rate percent 5
    queue-limit 20 ms
  !
  class BULK
    bandwidth percent 60
    queue-limit 50 ms
  !
  class VOICE_EXT
    priority level 1
    police rate percent 10
  !
  class VID
    priority level 2
    police rate percent 20
  !
interface GigabitEthernet0/0/0/0
  service-policy input LLQ_POLICY
!
interface GigabitEthernet0/0/0/1
  service-policy output LLQ_POLICY
```

Default minimum threshold for priority queues is 10ms

CBWFQ queue with minimum bandwidth guarantee

Default minimum threshold for regular queues is 100ms

All traffic with same priority level directed to the same queue

Lower priority than level 1

```
RP/0/RSP0/CPU0:Router# show policy-map interface Gi0/0/0/0
Gi0/0/0/0 direction input: Service Policy not installed

Gi0/0/0/0 output: CBWFQ-POLICY1

Class VOICE_INT
Classification statistics           (packets/bytes)          (rate - kbps)
      Matched          :               1319/1343754                154
      Transmitted      :               1319/1343732                154
      Total Dropped    :                  0/0                       0
   Queueing statistics
      Queue ID                            : 234
      High watermark (Unknown)
      Inst-queue-len (packets)            :  3
      Avg-queue-len  (Unknown)
      Taildropped(packets/bytes)          : 0/0
      Queue(conform)      :               434/435667               53
      Queue(exceed)       :               2/532                    6
      RED random drops(packets/bytes)     : 0/0
<snip>
```

Statistics for priority queue presented as for any other queue.

by Steven Crutchley

www.netquirks.co.uk