Medianet WAN/VPN QoS Design

The Case for QoS in Medianet WAN/VPN Networks

The case for QoS over the enterprise medianet WAN/VPN is largely self-evident, as these links are often orders of magnitude slower than the (Gigabit or Ten Gigabit Ethernet) campus or branch LAN links to which they connect. As such, these WAN/VPN edges usually represent the greatest bottlenecks in the network and therefore require the most attention to QoS design. QoS policies on the WAN/VPN serve to control jitter and packet loss.

Classification and marking are typically performed within the campus or branch. As such packets may be assigned QoS treatments on the WAN/VPN edge based on their DSCP markings. Queuing and congestion avoidance are critical QoS functions that are performed over the medianet WAN/VPN.

Three strategic QoS design principles that apply to WAN/VPN QoS deployments include:

- Always perform QoS in hardware rather than software when a choice exists.
- Enable queuing policies at every node where the potential for congestion exists,
- Enable congestion avoidance mechanisms to improve the efficiency of elastic flows

Medianet WAN/VPN QoS Design Considerations

There are several considerations that will impact IOS QoS designs within the medianet WAN/VPN:

- Modular QoS Command Line Interface
- Hierarchical Queuing Framework
- Transmit Ring
- Class-Based Weighted Fair Queuing
- Low-Latency Queuing
- Weighted Random Early Detect
- Resource Reservation Protocol
- QoS Roles in a medianet WAN/VPN

Modular QoS Command Line Interface (MQC)

MQC is a configuration syntax to construct QoS policies in the form of:

- class-maps which identify the flows using packet markings or other matching criteria
- policy-maps which specify policy actions to be taken on a class-by-class basis
- service-policy statements which apply a specific policy-map to an interface(s)

Hierarchical Queuing Framework (HQF)

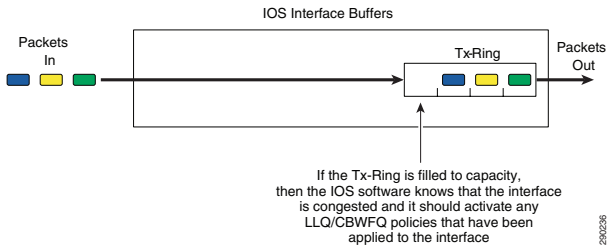
HQF refers to the common queuing behavior that Cisco has implemented on its distributed and non-distributed routing platforms. HQF provides the ability to:

- Provision multiple levels of packet scheduling
- Support integrated class-based shaping and scheduling
- Apply fair-queuing pre-sorters on a per-class basis

Transmit Ring (Tx-Ring)

The Tx-Ring is final IOS output buffer for a WAN interface (a relatively small FIFO queue) that maximizes physical link bandwidth utilization by matching the outbound packet rate on the router with the physical interface rate, as shown in Figure 1.

Figure 1 Cisco IOS Transmit Ring Operation



When the Tx-Ring fills to its queue-limit, it signals the IOS software to engage any LLQ/CBWFQ policies that have been attached to the interface. Subsequent packets are then queued within IOS according to these LLQ/CBWFQ policies, dequeued into the Tx-Ring, and then sent out the interface in a FIFO manner.

The Tx-Ring can be tuned with the **tx-ring-limit** interface configuration command.

Class-Based Weighted Fair Queuing (CBWFQ)

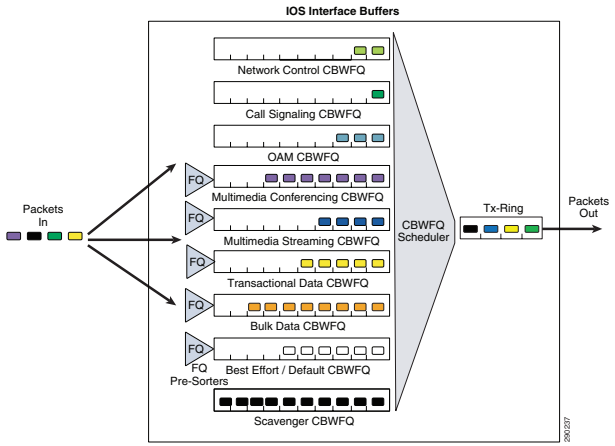
CBWFQ is an IOS queuing algorithm that combines the ability to guarantee bandwidth with the ability to dynamically ensure fairness to flows within a class.

The IOS software engages CBWFQ policies only if the Tx-Ring for the interface is full (which occurs only in the event of congestion). Once congestion has thus been signaled to the IOS software, each policy-map class configured with a **bandwidth** command is assigned its own queue.

CBWFQ queues may also have a fair-queuing pre-sorter applied to them with a **fair-queue** policy-map configuration command. In this manner, multiple flows contending for a single CBWFQ queue are managed fairly.

Additionally, each CBWFQ queue is serviced in a Weighted-Round-Robin (WRR) fashion based on the bandwidth assigned to each class. The CBWFQ scheduler then forwards packets to the Tx-Ring. The operation of CBWFQ is illustrated in Figure 2.

Figure 2 Cisco IOS CBWFQ Operation



Bandwidth allocated to a CBWFQ is not a static bandwidth reservation, but rather represents a minimum bandwidth guarantee, provided there are packets offered to the class. If there are no packets offered to the class, then the scheduler services the next queue and can dynamically redistribute unused bandwidth allocations to other queues, as needed.

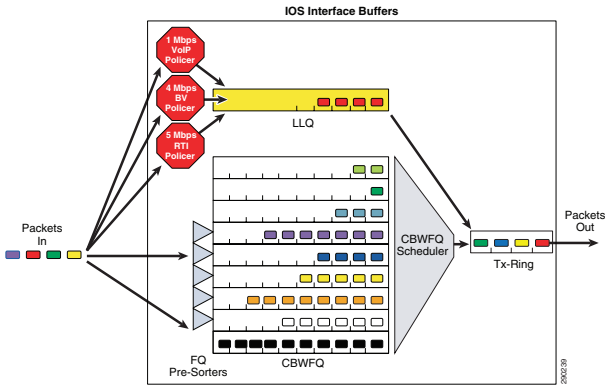
Low Latency Queuing (LLQ)

LLQ adds a strict-priority queue to the CBWFQ sub-system and is configured by a **priority** policy-map class configuration command.

An interesting facet of IOS LLQ is the inclusion of an implicit policer that admits packets to the strict-priority queue. This implicit policer limits the bandwidth that can be consumed by servicing the realtime queue and thus prevents bandwidth starvation of the non-realtime flows serviced by the CBWFQ scheduler. The policing rate for this implicit policer is always set to match the bandwidth allocation of the strict-priority queue; if more traffic is offered to the LLQ class than it has been provisioned to accommodate, then the excess traffic will be dropped by the policer.

The functionality offered by the implicit LLQ policer can be leveraged to prevent multiple types of realtime flows from interfering with each other. For example, per RFC 4594, there are up to three classes of traffic that may be provisioned with an Expedite Forwarding Per Hop Behavior (EF PHB): VoIP, Broadcast Video, and Realtime-Interactive. However, if these three traffic classes were provisioned with a single **priority** statement, then bursts from the Broadcast Video and/or Realtime-Interactive class could potentially interfere with (the better behaved) VoIP class. On the other hand, if each of these classes were provisioned with multiple **priority** statements, then each of these classes would be metered by a dedicated implicit policer to ensure that adequate strict-priority queuing is guaranteed per EF class, as shown in Figure 3.

Figure 3 Cisco IOS Multi-LLQ Operation



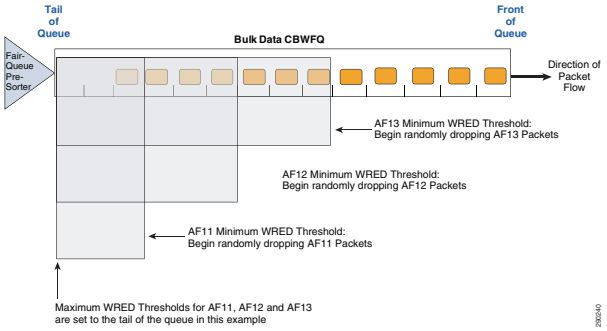
With such a multi-LLQ policy, traffic vying for strict-priority queuing will be serviced on a first-come, first-serve basis, provided the packets are admitted by their respective policers.

Weighted Random Early Detect (WRED)

While congestion management mechanisms like LLQ/CBWFQ manage the front of the queue, congestion avoidance mechanisms like Weighted-Random Early Detect (WRED) manage the tail of the queue. Congestion avoidance mechanisms work best with TCP-based applications because selective dropping of packets causes the TCP windowing mechanisms to "throttle-back" and adjust the rate of flows to manageable rates.

The primary congestion avoidance mechanism in IOS is WRED, which randomly drops packets as queues fill to capacity. However, the randomness of this selection can be skewed by traffic weights. The weight can either be IP Precedence values, as is the case with default WRED which drops lower IPP values more aggressively (for example, IPP 1 would be statistically dropped more aggressively than IPP 6) or the weights can be AF Drop Precedence values, as is the case with DSCP-Based WRED which statistically drops higher AF Drop Precedence values more aggressively (for example, AF23 is dropped more aggressively than AF22, which in turn is dropped more aggressively than AF21). DSCP-based WRED is enabled with the **dscp-based** option in conjunction with the **random-detect** policy-map class configuration command. The operation of DSCP-based WRED is illustrated in Figure 4.

Figure 4 Cisco IOS DSCP-WRED Operation

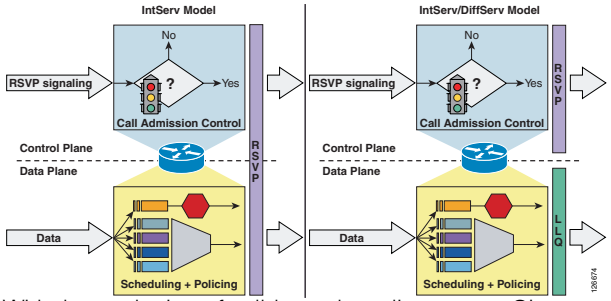


Resource Reservation Protocol (RSVP)

RSVP can be deployed in two operational models, as shown in Figure 5.

- The IntServ Model is the legacy RSVP operational model and has been largely abandoned due to inherent scalability limitations.
- The IntServ/DiffServ Model separates control plane operations from data plane operations. RSVP operation is limited to admission control only, with DiffServ mechanisms handling classification, marking, policing, and scheduling operations. As such, the IntServ/DiffServ model is highly scalable and flexible.

Figure 5 Cisco IOS RSVP Operational Models

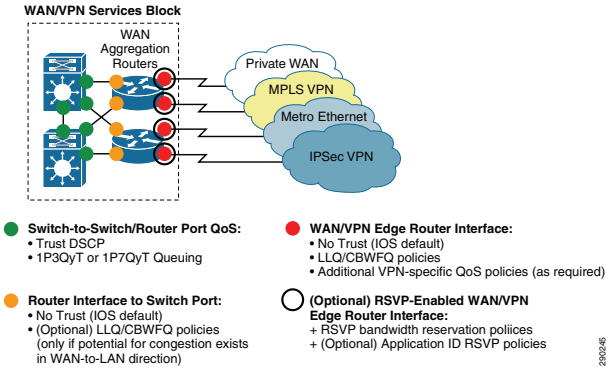


With the explosion of call-based media streams Cisco recommends deploying the IntServ/DiffServ model as this allows for efficient scaling of QoS policies along with dynamic network-aware admission control.

QoS Roles in a Medianet WAN/VPN

QoS roles in a medianet WAN/VPN network are illustrated in Figure 6.

Figure 6 Medianet WAN/VPN Interface QoS Roles



For more details, see Medianet WAN Aggregation QoS Design 4.0: http://www.cisco.com/en/US/docs/solutions/Enterprise/WAN_and_MAN/QoS_SRND_40/QoSWAN_40.html.