A Review of Congestion Management Algorithms on Cisco Routers

SZILÁGYI Szabolcs[1], ALMÁSI Béla[2]
[1] Department of Computer Science,
University of Oradea, Faculty of Electrical Engineering and Information Technology,
Universităţii 1, 410087 Oradea, Romania, E-Mail: szilagyi.szabolcs@inf.unideb.hu, sszilagyi@uoradea.ro

[2]Department of Informatics Systems and Networks,
University of Debrecen, Faculty of Informatics,
Kassai 26, 4028 Debrecen, Hungary, E-Mail: almasi.bela@inf.unideb.hu

***Abstract* – This paper presents one of the features of DS (Differentiated Services) architecture, namely the queuing or congestion management. Packets can be put in separate buffer queues, on the basis of DS value. Several forwarding policies can be used to favor high priority packets in different ways. The main reason for queuing is that a router must hold a packet in its memory, and meanwhile the outgoing interface is busy with sending another packet. The queuing tools are covered in the order in which they were added as Cisco IOS features: FIFO (First-In First-Out), CQ (Custom Queuing), PQ (Priority Queuing), WFQ (Weighted Fair Queuing), CBWFQ (Class Based Weighted Fair Queuing) and LLQ (Low Latency Queuing).**

***Keywords:* congestion; queuing; FIFO; PQ; CQ; WFQ; CBWFQ; LLQ.**

I. INTRODUCTION

In the beginning, the Internet was designed for data processing applications where delays were not very important. In the majority of cases a best effort delivery service was enough, and when data was lost or corrupted, the TCP protocol took care of the retransmission and recovery which was necessary. Today these expectations have changed due to the growth of multimedia applications, which require higher bandwidth (they need megabits per second instead of kilobits per second which was used for data processing applications). Nowadays applications are quite sensitive for the delays experienced when transmitting over the Internet. Therefore it is important to keep track of the delay and delay variation (jitter) and ensure they don't overgrow.That is why it is needed to support a variety of traffic with different quality of service (QoS) [1]. The most important side of this is how to share the existing resources while experiencing congestion. In order to proceed with this, it is needed that different mechanisms help differentiate between the types of traffic (prioritize).

The mechanisms which facilitates the queuing on each interface consists in hardware and software components.[2] If the hardware queue, often named "transmit queue" (TxQ) is not congested or full (exhausted), the packets are not kept in the software queue. They are switched directly to the hardware queue, where they are transferred quickly to the medium using FIFO order. In the case when the hardware queue is full, the packets are held in the software queue, processed, and released to the hardware queue based on the software queuing discipline. (Figure 1) [3].The software queuing discipline could be FIFO (First-In First-Out), CQ (Custom Queuing), PQ (Priority Queuing), WFQ (Weighted Fair Queuing), CBWFQ (Class Based Weighted Fair Queuing) and LLQ (Low Latency Queuing).

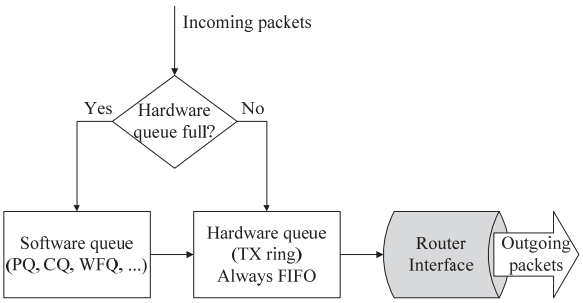

Figure 1. Queuing components.

II. FIFO QUEUING

First-In First-Out (FIFO) is the most simple modality of queuing. The incoming packets are put in a single queue and are processed in the order of receiving them. (Figure 2) Packets are dropped when the FIFO queue is full (tail drop). [2] This queuing type requires little computation and its behavior is very predictable, i.e. the delay of packet is a direct function of the queue size. Cisco IOS defaults to use FIFO on Fast Ethernet and Gigabit Ethernet interfaces with bandwidths above E1 speeds (2.048 Mbps). [4] There are several undesirable characteristics related to this queuing policy, because of its simplistic approach. [1]

- It is not possible to offer different services for different packet classes as all packets are put into the same queue.

- If an incoming flow suddenly becomes bursty, then it is possible for the entire buffer space to be filled by this single flow and other flows will not be serviced until the buffer is emptied.

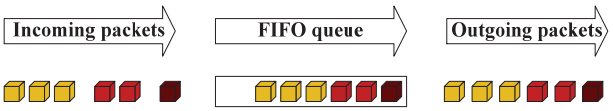

Figure 2. FIFO.

III. PRIORITY QUEUING

"A simple way to offer different services to different classes of packets is Priority Queuing. Its operation involves classifying each incoming packet into different priorities and placing them into separate queues accordingly." [1] In Cisco IOS, PQ uses up to four queues, named high, medium, normal, and low (Figure 3) [5], and they are scheduled as shown in Figure 4. [4] The packets having the highest priority are transferred on the output port before the lower priority packets. [1] PQ uses tail-drop logic, so when a new packet arrives for a particular queue, and the queue is full, the new packet is dropped. Even if this queuing type is suitable of providing differentiated service, it also has some drawbacks, such as large continuous flow of high priority traffic into the queue, equals excessive delay, and perhaps even service starvation for lower priority packets [1].

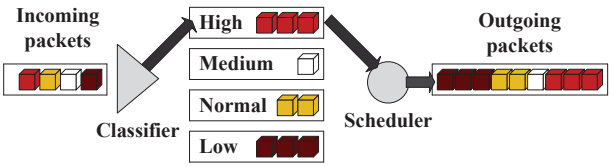

Figure 3. PQ.

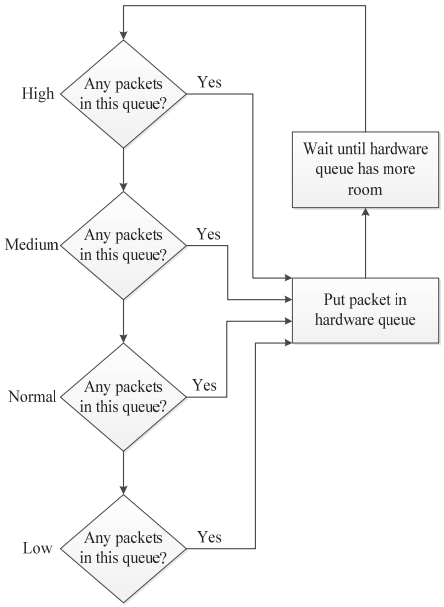

Figure 4. PQ scheduling logic [4].

IV. CUSTOM QUEUING

CQ addresses the biggest shortcoming of PQ ensuring a guaranteed minimum bandwidth to each queue, thereby queue starvation is avoided. With CQ up to 16 queues can be created by the network administrator in order to categorize traffic. (Figure 5) The queues are emptied one by one in a round-robin fashion, starting with queue 1. CQ takes packets from the queue, until the total byte count which was specified for the queue has been met or exceeded. After the queue has been serviced for the defined byte count, or when the queue does not have any more packets, CQ moves on to the next queue and repeats the process. (Figure 6)

One of the CQ queues can be setup as a default queue in order to manage traffic that is not identified specifically by the classification process. There is also one system queue which is hidden, used for important overhead traffic (routing protocol hellos, etc.). This system queue is serviced before all other queues. Cisco permits use of this queue 0, but does not recommend it. CQ uses the same classification options, and can use tail drop only for managing drops. [4]

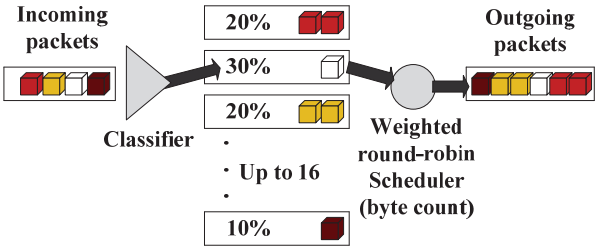

Figure 5. CQ.

The disadvantage of CQ, as compared to PQ, is the lack of a high-priority queue that is always serviced first. That is, CQ has no way to provide guaranteed low-latency service to any traffic. The CQ scheduler reserves an approximate percentage of overall link bandwidth for each queue, but instead of configuring actual percentages, CQ approximates the bandwidth percentages using a simple algorithm.

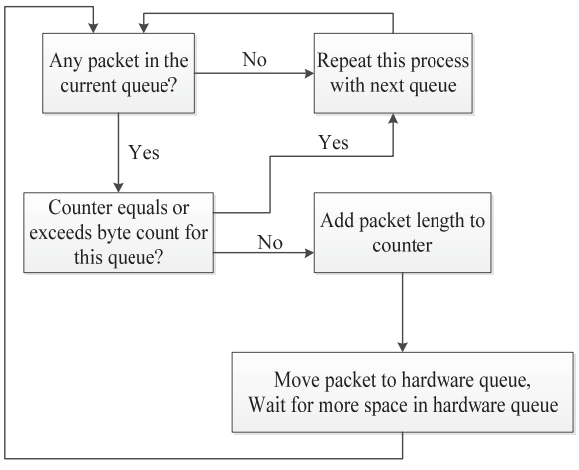

Figure 6. CQ scheduling logic for current queue [4].

V. WEIGHTED FAIR QUEUING

Processor Sharing (PS) is a type of queuing methodology having the purpose to allow fair access for each incoming flow and to prevent a bursty flow from consuming all the output bandwith. PS includes a queue for each distinct flow and packets from each flow are put into its appropriate queue. Then the system serves the queues one packet at a time using a round-robin approach.

Weighted Fair Queuing (WFQ) [6] is a subtype of Processor Sharing (PS) and it supports flows with different bandwidth requirements.[1] Weighted fair queuing differs from PQ and CQ in some significant features. Most obviously we can mention that WFQ does not allow classification options to be configured. Based on flows, WFQ classifies packets automatically, with each flow being placed into a separate queue. (Figure 7)

For the purposes of WFQ, a flow can be described as all packets with the same values for: source IP address, destination IP address, Transport layer protocol, TCP or UDP source port, TCP or UDP destination port and IP Precedence. Because WFQ puts packets of different flows in different queues, must have a greater number of queues than all of the non-flow-based queuing instruments. The WFQ scheduler uses logic that is somehow different from the logic of other queuing tools in order to be able to deal with the larger number of queues. [4]

In WFQ, there is a time stamp on each incoming packet with a finish time in addition to placing into its corresponding flow queue. In contrast to Processor Sharing, the selection of the packet to be served is now based on the time stamp on each packet. Further packets are serviced by examining their finish times. “The ones with earlier finish times are transmitted before the ones which have later finish times. It is possible for a later packet to have a finish time stamp that is smaller than an earlier packet.” [1]

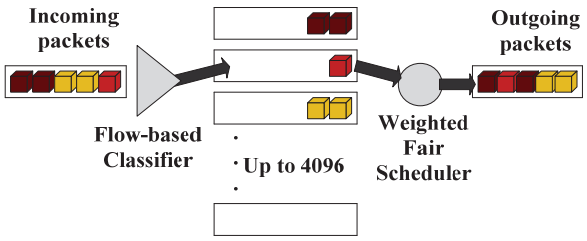

Figure 7. WFQ.

The WFQ scheduler takes the packet having the lowest finish time *(FT)* (sometimes called *sequence number*, or *SN*) when it needs to move the next packet to the hardware queue. WFQ associates to each packet an SN when the packet is added to a WFQ flow queue. The WFQ scheduler includes both the packet length and IPP when calculating the SN. The formula to calculate the SN for a packet is as below:

$$Previous\_SN + (weight * new\_packet\_length) \quad (1)$$

Where *weight* is calculated as follows:

$$weight = 32{,}384 / (IP\_Precedence + 1) \quad (2)$$

“The formula considers the length of the new packet, the weight of the flow, and the previous SN. By considering the packet length, the SN calculation results in a higher number for larger packets, and a lower number for smaller packets. By including the SN of the previous packet enqueued into that queue, the formula assigns a larger number for packets in queues that already have a larger number of packets enqueued. And by putting the weight (IPP + 1) in the denominator, packets with higher IPP values end up with lower SNs.

WFQ uses a two-step process called modified tail drop to choose when to drop packets. First, WFQ considers the absolute limit on the number of packets enqueued among all queues. This limit is called the hold-queue limit. If a new packet arrives, and the hold-queue limit has been reached, the packet is discarded. That part of the decision is based not on a single queue, but on the whole WFQ queuing system for the interface. Second, WFQ considers the length of the queue into which the newly arrived packet will be placed. Before adding a new packet to its queue, the congestive discard threshold (CDT) is checked against the actual length of that queue. If that queue is longer than CDT packets long, one packet is discarded—but maybe not the newly arrived packet. Figure 8 depicts the WFQ drop decision process.

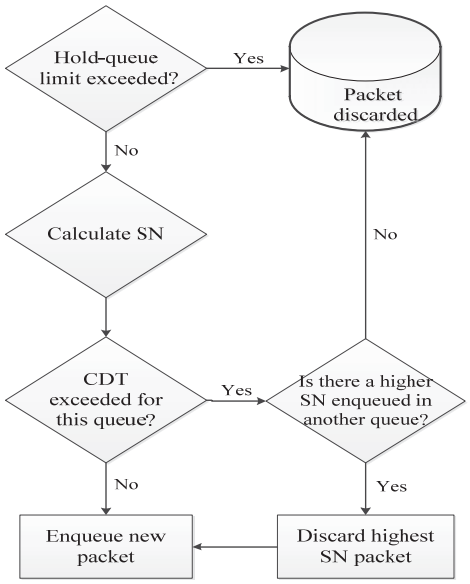

Figure 8. WFQ modified tail drop and congestive discard threshold [4].

WFQ can be configured for a maximum of 4096 queues. The allowed configurable values are powers of 2, between 16 and 4096, inclusive. IOS restricts the values because WFQ performs a hash algorithm to classify traffic, and the hash algorithm only works when the number of queues is one of these valid values. Additionally, WFQ keeps eight hidden queues for

overhead traffic generated by the router. WFQ uses a very low weight for these queues to give preference to the overhead traffic." [4].

VI. CLASS BASED WEIGHTED FAIR QUEUING

"CBWFQ [7] carries the WFQ algorithm further by allowing user defined classes, which allow greater control over traffic queuing and bandwidth allocation. CBWFQ provides the power and ease of configuration of WFQ, along with the flexibility of custom queuing. CBWFQ allows the creation of up to 64 individual classes plus a default class. (Figure 9) The number and size of the classes are based on the bandwidth. By default, the maximum bandwidth that can be allocated to user-defined classes is 75 percent of the link speed. This maximum is set so that there is still some bandwidth for Layer 2 overhead, routing traffic (BGP, EIGRP, OSPF, and others), and best-effort traffic.

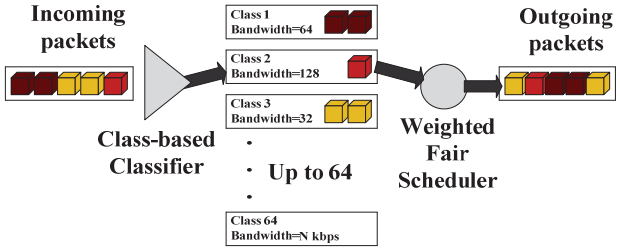

Figure 9. CBWFQ.

Each user-defined class is guaranteed a certain bandwidth, but classes that exceed that bandwidth are not necessarily dropped. Traffic in excess of the class's guaranteed bandwidth may use the "free" bandwidth on the link. "Free" is defined as the circuit bandwidth minus the portion of the guaranteed bandwidth currently being used by all user-defined classes. Within this "free" bandwidth, the packets are considered by fair queuing along with other packets, their weight being based on the proportion of the total bandwidth that was guaranteed to the class. [8]

All packets not falling into one of the defined classes are considered part of the default class. The default class can be configured to have a set bandwidth like other user-defined classes, or configured to use WFQ in the remaining bandwidth and treated as best effort. When the fair queuing buffers overflow, packets are dropped with tail drop unless WRED (Weighted Random Early Detection) [9] has been configured for the class's policy. In the latter case, packets are dropped randomly before buffers totally run out in order to signal the sender to throttle back the transmission speed." [4]

VII. LOW LATENCY QUEUING

Neither WFQ nor CBWFQ can provide guaranteed bandwidth and low-delay guarantee to selected applications such as VoIP. "That is because those queuing models have no priority queue. Certain applications such as VoIP have a small end-to-end delay budget and little tolerance to jitter. LLQ [10] includes a strict-priority queue that is given priority over other queues, which makes it ideal for delay and jitter-sensitive applications. Unlike the plain old PQ, whereby the higher-priority queues might not give a chance to the lower-priority queues and effectively starve them, the LLQ strict-priority queue is policed. This means that the LLQ strict-priority queue is a priority queue with a minimum bandwidth guarantee, but at the time of congestion, it cannot transmit more data than its bandwidth permits. If more traffic arrives than the strict-priority queue can transmit, it is dropped. Hence, at times of congestion, other queues do not starve, and get their share of the interface bandwidth to transmit their traffic. Figure 10 shows an LLQ.

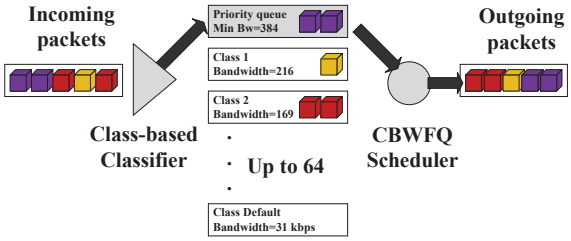

Figure 10. LLQ.

As we can see, LLQ is effectively a CBWFQ with one strict-priority queues added. It is possible to have more than one strict priority queue. This is usually done so that the traffic assigned to the two queues – voice [11], [12] and video traffic, for example - can be separately policed. However, after policing is applied, the traffic from the two classes is not separated. It is sent to the hardware queue based on its arrival order (FIFO). As long as the traffic that is assigned to the strict-priority class does not exceed its bandwidth limit and is not policed and dropped, it gets through the LLQ with minimal delay. This is the benefit of LLQ over CBWFQ." [3] The LLQ scheduler logic is shown in Figure 11.

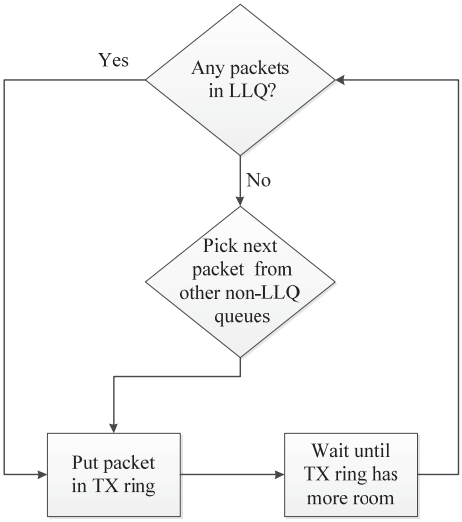

Figure 11. LLQ scheduler logic [4].

VII. CONCLUSIONS

Table 1 summarizes some of the key points regarding the IOS queuing tools covered in this paper.

TABLE 1. Queuing protocol comparison [4].

Feature	FIFO	PQ	CQ	WFQ	CBWFQ	LLQ
Includes a strict-priority queue		Yes				Yes
Polices priority queues to prevent starvation						Yes
Reserves bandwidth per queue			Yes		Yes	Yes
Includes robust set of classification fields					Yes	Yes
Classifies based on flows				Yes	Yes[2]	Yes[2]
Maximum number of queues	1	4	16[1]	4096	64	64

[1]Also includes a system queue that is unavailable for customer use.
[2]WFQ can be used in the class-default queue, or in all CBWFQ queues in 7500 series routers.

REFERENCES

[1] T. Svensson, A. Popescu, *"Development of laboratory exercices based on OPNET Modeler"*, Master thesis, Blekinge Institute of Technology, Department of Telecommunications and Signal Processing, 2003

[2] QOS, *"Implementing Cisco Quality of Service"*, Student Guide, Volume 2, Version 2.2, © 2006 Cisco Systems Inc.

[3] A. S. Ranjbar, *"CCNP ONT Official Exam Certification Guide"*, Cisco Press, 2007

[4] W. Odom, J. Geier, N. Mehta, *"CCIE Routing and Switching Official Exam Certification Guide"*, Second Edition, Cisco Press, 2006

[5] *"Cisco IOS Quality of Service Solutions Configuration Guide"*, Release 12.4T, © 2008 Cisco Systems Inc.

[6] M. Barreiros, P. Lundqvist, *"QOS-Enabled Networks - Tools and Foundations"*, A John Wiley & Sons, Ltd., Publication, 2011

[7] W. Odom, R. Healy, D. Donohue, *"CCIE Routing and Switching Certification Guide"*, 4th Edition, Cisco Press, 2010

[8] B. Durand, J. Sommerville, M.Buchmann, R. Fuller, *"Administering Cisco QoS in IP Networks"*, Syngress Publishing, Inc., 2001

[9] R. S. Benn, S.C. Kronenberg, E. Rozell, *"Configuring Cisco AVVID – Architecture for Voice, Video, and Integrated Data"*, Syngress Publishing, Inc., 2001

[10] K. Wallance, *"Authorized Self-Study Guide - Cisco Voice over IP (CVOICE)"*, 3rd Edition, Cisco Press, 2009

[11] P. K. Verma, L. Wang, *"Voice over IP Networks - Quality of Service, Pricing and Security"*, Lecture Notes in Electrical Engineering, Vol. 71, Springer, 2011

[12] S. Kashihara, *"VoIP Technologies"*, INTECH Open Access Publisher, 2011