Volume 2, Issue 11, November 2012 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Avoiding Congestion by using Weighted Fair Queuing

V.K Gupta *, **Mohamed Seidi Ahmed Hmadi**
vinodgupta602@gmail.com *, *hmadi2595@gmail.com*
Department of Computer Science & Engineering
Nims Institute of Engineering and Technology, NIMS University, Rajasthan,
Jaipur (India)

Abstract: ***Is a sophisticated set of techniques, which is designed to reduce delay variability and provide predictable throughput and response time for traffic flows? A goal of WFQ is to offer uniform service to light and heavy network users alike. WFQ ensures that the response for the low-volume applications is consistent with the response time for high-volume applications. Applications that send small packets are not unfairly starved of bandwidth by applications that send large packets. Bandwidth is divided up equitably in an automated way. WFQ is a flow-based queuing technique that recognizes a flow for an interactive application and schedules that applications traffic to the front of the queues to reduce the response time. Low-volume traffic streams for interactive applications, which introduce a large percentage of the applications on most Networks, are allowed to transmit their entire offered loads in a timely manner. High-volume traffic streams share the remaining capacity. WFQ adapts automatically to change network traffic conditions and requires little to no configurations.***

Keywords— Static WFQ, Dynamic WFQ, Weighted Fair Queuing, time- complexity, Minimum- WFQ

1. INTRODUCTION

For applications that use the IP precedence field in IP header, WFQ [1] can allot bandwidth based on precedence. The technique allocates more bandwidth to conversations with higher precedence, and makes sure those conversations get served more quickly when congestion occurs. WFQ assigns a weight to each flow, which determines transmitting order for queued packets IP precedence helps determine the weighting factor. WFQ [2] also works with RSVP [3] uses WFQ to allocate buffer space, schedule packets, and guarantee bandwidth based on resource reservations. Additionally WFQ understands the discards eligibility bit (DE), the forward explicit congestion notification (FECN) [4], and backward explicit congestion notification (BECN) [5] mechanisms in a frame relay network. After congestions have been identified, the weights used by WFQ are altered so that a conversation encountering congestion transmits less frequently. Weighted, when packets are classified, Runs on all standard IOS platform [6].

1.1 STATIC WFQ

Weighted Fair Queuing (WFQ)[7-9] offers fair queuing that divides the available bandwidth across queues of traffic based on weights. Each flow or aggregate thereof is associated with an independent queue assigned with a weight, to ensure that important traffic gets higher priority over less important traffic. In times of congestion, the traffic in each queue (a single flow or an aggregate of them) is protected and treated fairly, according to its weight.

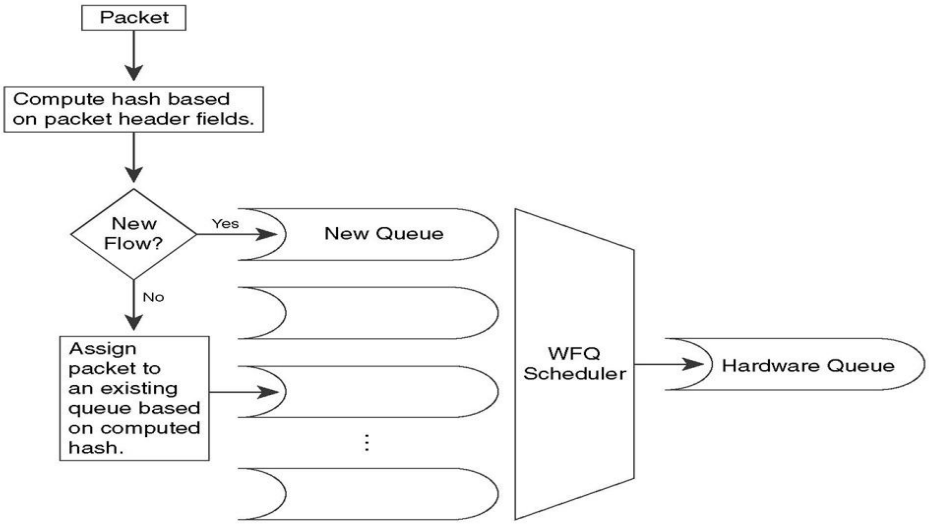

Fig.1 Static WFQ

Figure (1) how full the interface hold queue is, and based on whether the packet queue size is beyond a congestive discard threshold value, the packet might end up being dropped. It is worth mentioning that when a packet arrives, it is assigned a sequence number for scheduling purposes. The priority of a packet or flow influences its scheduling sequence number.

Arriving packets are classified into different queues by inspection of the packet header fields, including characteristics such as source and destination network or MAC address, protocol, source and destination port and socket numbers of the session or Diff-Serv-Code-Point (DSCP) value. Each queue shares the transmission service proportionally to the associated weight. All traffic in the same class is treated indistinctly. WFQ can certainly ensure satisfactory response time to critical applications, such as interactive and transaction based ones that are intolerant to performance degradation, in particular whether deployed in an Int-Serv architecture. In a Diff-Serv architecture, WFQ can be IP DSCP-aware. This means that it is able to detect higher priority packets marked with precedence and can schedule them faster, providing superior response time for these traffic aggregates. In summary, from a technical point of view WFQ has three desirable properties. First, because it approximates GPS (General Processor Sharing) scheduler, it protects traffic of different queues from each other, which is fundamental in a service differentiation context. Second, traffic in a queue can obtain worst-case end-to-end queuing delay that is independent of the number of hops it traverses and of the behavior of traffic in the other queues.

1.2 Dynamic WFQ

A Dynamic WFQ [9-12] is able to dynamically and consistently adapt the queue weights according to the time-variant amount of the incoming traffic and the pre-assigned target QoS. The fundamental issue is to correlate the burstiness of the traffic with the weight value in order to achieve given delay and loss guarantees. The measurement of the burstiness of the aggregate entering each queue could be realized by evaluating the resulting buffer dimension, which is somehow related to the worst-case delay experienced by packets in the queue. For what concerns the QoS, we could apply a proportional relative model. Each queue has assigned a static parameter and the performance guarantees provided to the set of queues should be in line with the mutual ratio of the said parameters. For example, if the queue Qi and Qj have the parameters Pi and Pj respectively, with Pj=2*Pi, the QoS provided to Qj should be two times

better then the one granted to Qi. Hence, no absolute QoS assurances are supported in this case. Where also preliminary results are reported, if in a given interval Tn, Bn represents the average buffer dimension of Qn, which is associated with the parameters Pn, the queue weights could be determined by the resolution of a linear system whose equations are of the form:

$(Bi/Bj)*(Pi/Pj) = (Wi/Wj)$

where, Wi and Wj are the weights to be assigned to the queues Qi and Qj, respectively.

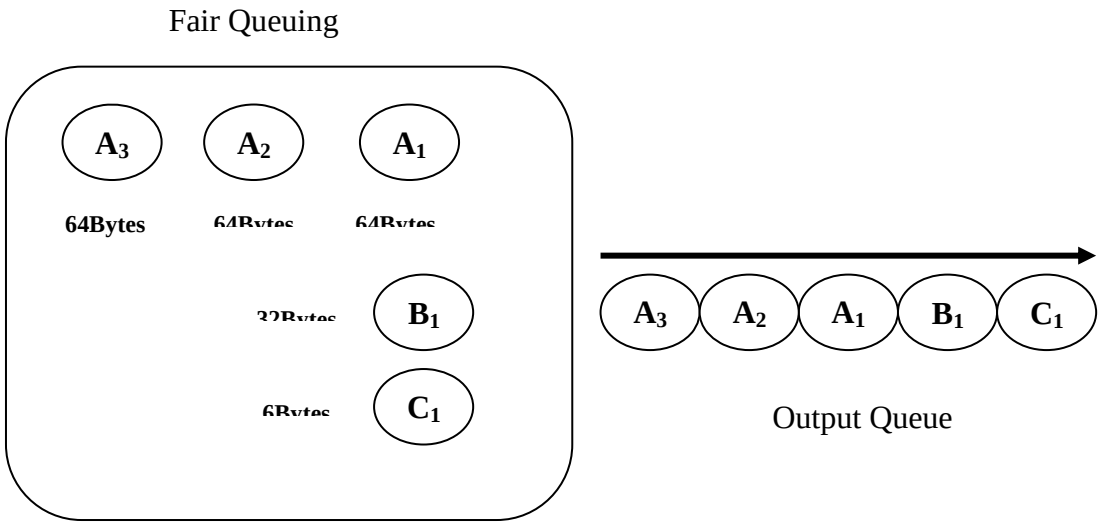

Figure 2. Fair Queuing

With WFQ, a packet's IP Precedence influences the order in which that packet is emptied from a queue. The order of packet servicing with WFQ is based on sequence numbers, where packets with the lowest sequence numbers are transmitted first. The sequence number is the weight of packet multiplied by the number of byte-t-byte rounds that must be completed to service the packet.

Prior to IOS 12.0.5T , the formula for weight was as follows :

Weight = 4096 / (IP Precedence + 1)

In more recent IOS,the formula for weight is as follows :

Weight = 32384 / (IP Precedence + 1)

So let's take back our previous example:

A1 = (32384 / (5+1)) *64 ≈ 345429

A2 = (32384 / (5+1)) *64 + A1 ≈ 690859

A3 = (32384 / (5+1)) *64 + A2 ≈ 1036287

B1 = (32384 / (0+1)) * 32 ≈ 1036288

C1 = (32384 / (0+1)) * 6 ≈194304
So now with WFQ applied, the queues are emptied in the following order.

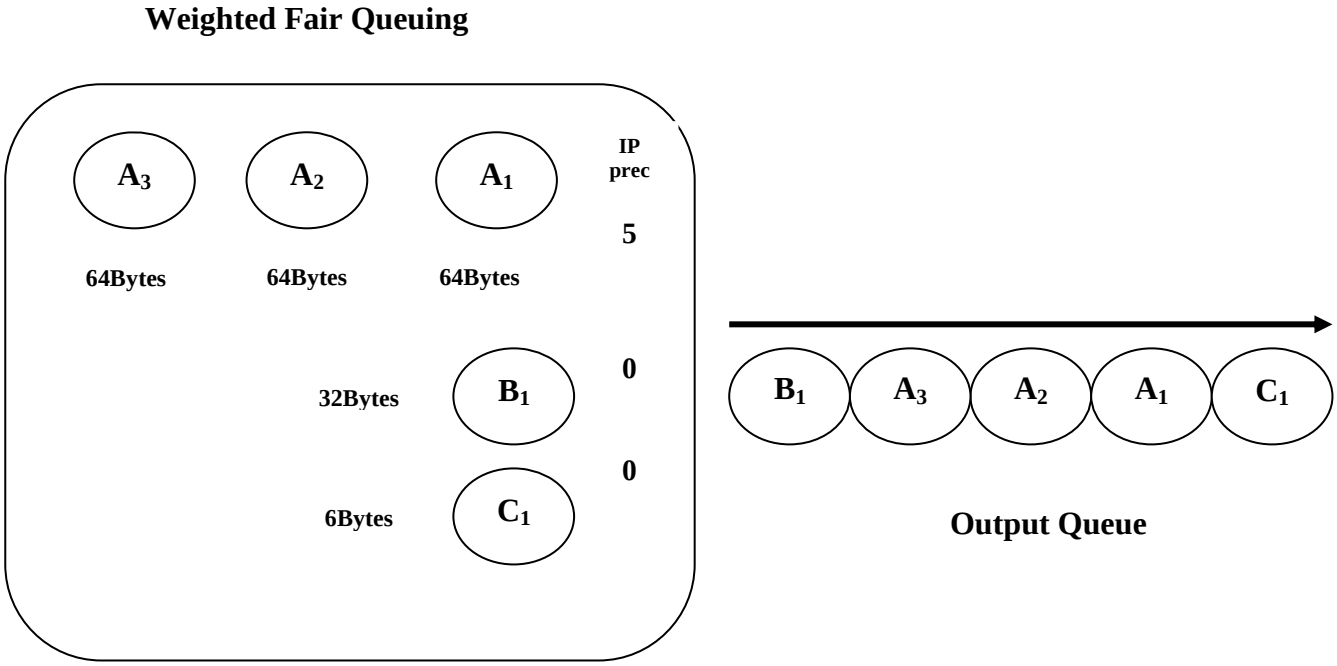

Fig. 3: Weighted Fair Queuing

Although WFQ is easy to configure (it is enabled by default on interest below 2048 Mbps), although WFQ is supported on all IOS versions, it has its limitations .Specifically, WFQ canot guarantee a specific amount of bandwidth for an application. And also if more than 256 flows exist, by default, more than one flow one flow can be forced to share the same queue.

2. Weighted Fair Queuing (WFQ)

Instead of paying attention to the most demanding protocols, Weighted Fair Queuing (WFQ) [13-16] views all incoming packets as part of a "network flow" (also called a "conversation") between two network nodes. It can use the source and destination IP addresses, TCP/UDP ports, or anything else to determine the flow to which the packets belong. Once it knows which packets belong to which flows, WFQ can provide all conversations with equal access to the interface.
WFQ uses IP Precedence to weight the traffic flows and then handle those with the highest weights preferentially. The fact that WFQ uses IP Precedence is one of its advantages. Another advantage is that it ensures that an extremely busy flow will starve itself before causing dropped packets in other more well-behaved flows.

3. Weighted Fair Queuing Configuration

For purposes, in WFQ [17] the length of the queue is not measured in packets but in the time it would take to transmit all the packets in the queue. WFQ adapts the number of flows and allocates equal amounts of bandwidth to each flow. Small packet flows that are usually interactive flows (voice and video) typically receive better service because they do not need a lot of bandwidth. They do need and receive low delay because smaller packets have a lower finish time. Finish time is the sum of the current time and the time it takes to transmit the packet. Current time is zero if there are no packets in the queue.
The first packet into this queue will have a finish time of current time + transmission time.

Example:

Current time = 0ms (no packets in the queue)
First packet = 100ms to transmit this packet
Finish time = 0 + 100ms = 100ms
Packet two comes into the queue
Current time = 100ms
2nd Packet = 20ms to transmit this packet
Finish Time = 100ms + 20ms = 120ms

WFQ [18] will place the packets into the hardware queue based on the finish time lowest to highest. The last piece is to use the finish time and the IP precedence to introduce the weight into the calculation of which queues will be serviced in which order.
The calculation to add weight is finish time divided by IP Precedence plus one (to prevent division by zero). In Cisco routers this calculation is done differently to decrease the load on the routers CPU. The Cisco router uses the packet size instead of the

transmission time as they are proportional to each other. In addition, the packet size is not divided by IP Precedence; instead the packet size is multiplied by a fixed value (one value for each IP Precedence value).

This is done because division is a CPU intensive operation in comparison to multiplication.

IP Precedence	Weight
0	**16192**
1	**10794**
2	
3	**8096**
4	**6476**
5	**5397**
6	**4626**
7	**4048**
32 virtual IP precedence	**128**
1024 virtual IP precedence	**4 RSVP**

4. Supporting Congestion by using WFQ

Networking performance is measured in throughput and response time [19]. It is also important to achieve the optimal performance operating point without over-utilizing the system resources. To achieve such optimal operating point, network adapter vendors need to examine several areas, including:

- Which network adapter hardware capabilities are implemented
- Network adapter driver implementation for critical path length and scalability
- Dynamically adjustable hardware and software parameters to allow for auto-tuning

There are always tradeoffs in deciding which hardware functions to implement on a network adapter. It is becoming increasingly important to consider adding task offload features that allow for interrupt moderation, dynamic tuning on the hardware, improving the use of the PCI bus, and supporting Frames. These are particularly important for the high-end network adapter that will be used in configurations requiring top performance.

- TCP and IP Checksum Offload: Checksum calculation is the most expensive function in the networking stack for two reasons:
- It contributes to long path length
- It causes cache churning effects (typically on the sender)
-

5. Conclusion

We developed a partial characterization of when congestion control schemes can guarantee convergence, incentive compatibility, and efficiency, leaving several directions for future work. An apparently nontrivial problem that we leave open is to determine whether WFQ Queuing converges in the general case. It would also be interesting to study other queuing policies such as weighted Fair Random Early Drop (WFRED). While we have given sufficient conditions and necessary conditions for incentive compatibility and efficiency, they are not tight. A very interesting direction would be to derive conditions that are both sufficient and necessary, or at least to narrow the gap between the two sides. It would also be interesting to see whether in doing so, qualitative differences in convergence remain related to incentive compatibility, as we have demonstrated in the difference between local queuing policies (which can converge only asymptotically and cannot guarantee both incentive compatibility and efficiency) and WFQ (which converge within a finite time interval and can guarantee properties). WFQ has a hold policy that covers every queue, when the hold threshold is exceeded any new packets are dropped, this is called 'aggressive dropping'. Aggressive dropping does have one exception, if the destination queue is empty then the packet will be accepted. Every queue has a Congestion Discard Threshold (CDT), if the hold queue is not full but the CDT is reached then the packet will be dropped. This is called 'early dropping', the only exception is if a packet has a higher sequence number then that will be dropped instead.

Acknowledgement

The author is thankful to the Department of Computer Science and Engineering, Nims Institute of Engineering and Technology, NIMS University, Jaipur (Rajasthan) for providing the computing facilities to carry out this work. He is also thankful to his guide Mr. V.K Gupta and friend Naveen Bilandi.

References

[1] A. Akella, S. Seshan, R. Karp, S. Shenker, and C. Papadimitriou. Selfish behavior and stability of the internet: a game-theoretic analysis of tcp.

[2] A. Demers, S. Keshav, and S. Shenker. Analysis and simulation of a fair queuing algorithm.

[3] A. Demers, S. Keshav, and S. Shenker. Analysis and simulation of a fair queuing algorithm. Journal of Internetworking Research and Experience, pages 3{26, October 1990.

[4] J. Feigenbaum, C. H. Papadimitriou, R. Sami, and S. Shenker. A BGP-based mechanism for lowest-cost routing. Distributed Computing.

[5] Changwang Zhang, Jianping Yin, Zhiping Cai, and Weifeng Chen, RRED: Robust RED Algorithm to Counter Low-rate Denial-of-Service Attacks, IEEE Communications Letters, vol. 14, pp. 489-491, 2010.

[6] Sally Floyd, Van Jacobson. Random Early Detection Gateways for Congestion Avoidance (1993). IEEE/ACM Transactions on Networking, vol.1(4): pp.397–413. Invented Random Early Detection (WRED) gateways.

[7] Deploying IP and MPLS QoS for Multi-service Networks: Theory and Practice" by John Evans, Clarence Filsfils (Morgan Kaufmann, 2007, ISBN 0-12-370549-5)

[8]Kennedy I., Lost Call Theory, Lecture Notes, ELEN5007 – Teletraffic Engineering, School of Electrical and Information Engineering, University of the Witwatersrand, 2005.

[9]Peuhkuri M., IP Quality of Service, Helsinki University of Technology, Laboratory of Telecommunications Technology, 1999.

[10]Farr R.E., Telecommunications Traffic, Tariffs and Costs – An Introduction For Managers, Peter Peregrinus, 1988.

[11]Flood, J.E., Telecommunications Switching, Traffic and Networks, Chapter 4: Telecommunications Traffic, New York: Prentice-Hall, 1998.

[12]Ritter, M., Phuoc, P., Multi-Rate Models for Dimensioning and Performance Evaluation of ATM Networks, COST 242, Institute of Computer Science, University of Würzburg, June 1994.

[13] "The World's Technological Capacity to Store, Communicate, and Compute Information", Martin Hilbert and Priscila López ,2011.

[14] Worldwide Telecommunications Industry Revenues, Internet Engineering Task Force, June 2010.

[15] Introduction to the Telecommunications Industry, Internet Engineering Task Force, June 2012.

[16] C. Vogt, "Credit-Based Authorization for Mobile IPv6 Early Binding Updates", draft-vogt-mobopts-credit-based-authorization-00.txt, February 2005.

[17] W. Haddad, F. Dupont, L. Madour, S. Krishnan, and S. Park, "Optimizing Mobile IPv6 (OMIPv6)", draft-haddad-mipv6-omipv6-01.txt, February 2004.

[18] W. Haddad, L. Madour, J. Arkko, and F. Dupont, "Applying Cryptographically Generated Addresses to Optimize MIPv6 (CGA-OMIPv6)", draft-haddad-mip6- cga-omipv6-03, October 2004.

[19] W. Haddad, F. Dupont, L. Madour, A. Kavanagh, S. Krishnan, S. Park, and H. Kari,
"BUB: Binding Update Backhauling", draft-haddad-mipv6-bub-01.txt

