Volume 2, Issue 10, October 2012 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

# Enhancing QoS by Using Weighted Fair Queuing Techniques

**Mohamed Seidi Ahmed Hmadi**
Nims Institute of Engineering and Technology,
NIMS University, Rajasthan,
Jaipur (India)

***Abstract: Must applications in communications and Virtual Private Networks, (VPNs) require data networks to provide Quality-of-Service (QoS) guarantees, such as delay and jitter bounds to individual Network flows, providing such guarantees can be achieved by the link scheduling techniques along the path of these packets. Among many packet scheduling techniques proposed for this problem, Weighted Fair Queuing Techniques (WFQ)offers the best delay and fairness guarantees. Moreover, all previous work on WFQ has been focused on developing inefficient approximations of the scheduler because of perceived scalability problems in the WFQ computation. This work proves that the previously well accepted O (N) time- complexity for WFQ implementation, where N is the number of active flows handled by the scheduler, is not true. The other contribution of this paper is a Minimum Weighted Queuing (Minimum- WFQ) techniques which is a linear O (1) algorithm for implementing WFQ techniques. In addition the paper represents several performance studies demonstrating the power of the proposed techniques in providing precise delay bound to a large number of sessions with diverse QoS requirements.***

***Keywords- Virtual Private Networks, Weighted Fair Queuing, time- complexity, Minimum- WFQ***

## 1. Introduction

Applications with strict QoS [1] enhanced, such as bounded delay and jitter require the enforcement of some form of scheduling discipline along the path of packets guaranteed by these application. Scheduling methods vary there are two classes of scheduling in methodology and assumptions. In general, there are two classes of scheduling method; per-flow scheduling, aggregate scheduling. In Per-flow scheduling, a stream is assumed to have its own separate queue which contains only its packets. When the link becomes idle and there are packets in the queues, the scheduler arbitrates between the different stream queues choosing the packet that is to leave the link next (depending upon delay deadlines). In aggregate scheduling streams [2] are Aggregated and Per-flow scheduling is applied to these aggregate. Both Per-flow and aggregate scheduling methods deal with streams or flows. A flow can be viewed as a sequence of packets having a set of common characteristics. These characteristics vary depending on the way flows are classified and where in the network they are being classified. For example, a common way to classify flows is a combination of the source and destination Ip Address, the source and destination port number and possibly the application generating the packets. The reason behind such classification is that, traffic generated by a particular application from a given host and destined to another host usually has similar QoS requirements. Packet scheduling as a tool for providing per-flow or per traffic-class Quality-of-Service (QoS) guarantees in packet networks is well-understood and strongly supported by both fundamental theoretical arguments, as well as practical tests. The current Internet is based on a best-effort service model that does not provide any QoS assurances to different applications. This lack of service differentiation has serious impact on the type of applications[3,4] that require end-to-end QoS assurance over the Internet. For example, real-time communications and interactive applications over the Internet require resource reservation and scheduling at involved hosts and intermediate nodes. For such applications, the networks must provide guaranteed rates, bounded end-to-end delays, restricted packet loss, fairness, etc., to individual flows. With the proper dimensioning of network resources, the most important performance attributes of a packet-scheduling algorithm become its delay and fairness bounds for each flow. Delay bounds are important for a wide range of time-sensitive or real-time services. Fairness bounds are important for providing a sufficient degree of isolation to a flow of packets, so that the service guaranteed to that flow is not affected by the behavior or misbehavior of other packet flows, sharing the same link. To provide such guarantees, it is normally assumed that packet flows have been conditioned using an appropriate traffic shaper, such as a leaky-bucket conditioner, and that the policing is in effect at the network edges. Providing end-to-end delay bounds to individual flows in a packet network, such as the Internet, requires the use of schedulers that can guarantee packet service rates as well as fair allocation of excess bandwidth.

 

## 2. Generalized Processor Sharing (GPS)

Is an ideal scheduler [5-7] that provides every flow its guaranteed bit-rate and distributes excess bandwidth fairly among flows according to their relative bandwidth weights? As a result, GPS [5] can provide end-to-end delays and fairness guarantees to packet flows that are shaped by leaky bucket traffic conditioners. GPS works by assigning a distinct queue to each flow (or session), then servicing an infinitesimally small amount from each session according to a weighted cyclical schedule. Unfortunately, GPS is un-realizable in practice because it services a small part of a packet at a time. A real scheduler must complete the service of an entire packet from a session before it moves to the next session. Packet-by-packet GPS, commonly known as Weighted Fair Queuing (WFQ), is one of the GPS emulation algorithms that transmits packets according to their finish order under GPS , WFQ simulates a GPS fluid-model in parallel with the actual packet-based scheduler in order to calculate the virtual finish number (used as a timestamp) for packets arriving to the scheduler. To calculate the finish number, WFQ [6] maintains the state of the system by means of a Virtual Time function V(t) which is a piecewise linear function of real time and whose slope changes depending on the number of backlogged sessions and their service rates. To perform scheduling in real-time, WFQ must update the virtual time before any packet arrival, so that every arriving packet gets the proper virtual finish number (as if it will be departing under GPS). The virtual-time function is impacted by arrivals (to empty queues), as well as departures of packets (that result in empty session queues). The problem is that, an undetermined (and possibly large) number of session queues can become empty at the same time, because under GPS many packets can end up having the same virtual finish time. Therefore, updating the virtual time function in between two consecutive packet arrivals may incur a large number of computations. In particular, if a link is shared by up to iV active sessions, then updating the virtual time can incur a computation on O(N) sessions or queues. This problem is usually referred to as iterated deletion, and is the main reason.

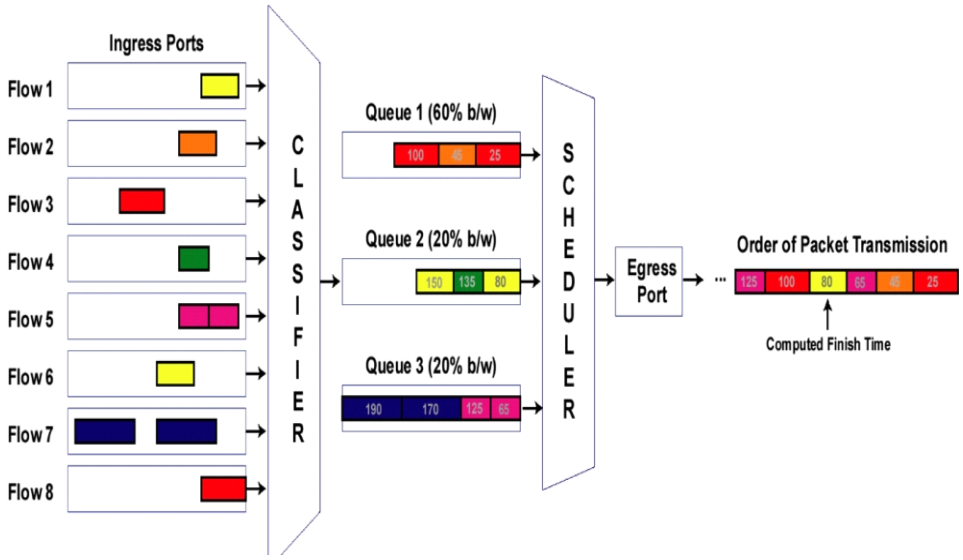


Figure (1): Weighted Fair Queuing Techniques (WFQT)

## 3. Assumptions and Terminology

In the following theorem, we will find sufficient conditions under which WFQ [8-14] will produce equal timestamps. Although WFQ [15]can produce equal timestamps without them, we will make a few assumptions to help us find simple closed form expressions for the relationship between packet length and session share. In these two theorems, packets arrive in a back-to-back manner, with no inter-packet gaps, and belong to unique sessions. The first packet arrives at the start of the busy period. In the second theorem, we assume that the busy period has already started. In this case, we are interested in generating equal timestamps for packets arriving at or after that moment in time. We also assume in both theorems that the input link is at least as fast as the output link.

Let $t_i$ be the arrival time of the ith packet to the GPS [8] system such that its session number is also i, Li is the length of the ith packet,Cin is the input rate, and C is the output link rate such that Cin > C. Assume that t1 = 0, L1= L, and assume that packets arrive in back-to-back manner with no inter-packet gaps, i.e.

 

$$t_n - t_{n-1} = \frac{L_n}{C_{in}}$$

Assume also, that at time t = 0, the virtual time V (0) = 0 and that the share of session *i* is $\alpha_i, where \alpha_1 = 1, \alpha_i \succ 0, i \geq 2.$

**Note** that the above assumptions imply that only one packet arrives to each session and therefore all packet arrivals are to idle (empty) session queues.

**3.1 Theorem 1**

Assuming that m packets arrive to m idle session queues, one packet to each queue (indexed from 1 to m), such that the packet lengths satisfy the relation[20-24]:

$$\frac{L_n}{L_{n-1}} = \beta \frac{\alpha_n}{\alpha_{n-1}} \frac{\sum_{i=1}^{n-1} \alpha_i}{\beta \sum_{i=1}^{n-1} \alpha_i + \alpha_n}, 2 \leq n \leq m \leq M \text{ ---------- (1)}$$

Where $\beta = \frac{C_{in}}{C}$, then all the m packets will have the same timestamp value as that of the first packet.

**Proof:** According to the assumptions, the guaranteed rate $r_i$ of session *i* according to GPS is $r_i = \alpha_i r$ where $r = \frac{C}{\sum_{i=1}^{N} \alpha_i}$.

When packet 1 arrives at time $t_1$=0, it receives a timestamp equal to

$TS(t_1)= TS(0)=V(0)+ \frac{L_1}{\alpha_1 r} =0+ \frac{L}{r} = \frac{L}{r}$

(Since $\alpha_1 = 1$ and $L_1 = L$). Using the fact that, between times $t_{n-1}$ and $t_n$ only sessions 1, 2…, n-1 are active, we conclude that the virtual time slope during the same period of time is

$$\frac{C}{r \sum_{i=1}^{n-1} \alpha_i}$$

Therefore, we have

$$V(t_n) = V(t_{n-1}) + \frac{C}{r \sum_{i=1}^{n-1} \alpha_i} (t_n - t_{n-1}) \text{----------(1.1)}$$

From which we get

$$V(t_n) = V(t_{n-1}) + \frac{C}{r \sum_{i=1}^{n-1} \alpha_i} (\frac{L_n}{C_{in}})$$

$$= V(t_{n-1}) + \frac{L_n}{r\beta \sum_{i=1}^{n-1} \alpha_i} \text{ ---------- (1.2)}$$

Since each packet arrives to an idle session, the timestamp of the $n^{th}$ packet is:

$TS (t_n) = V(t_n)+ \frac{L_n}{\alpha_n r}$ ----------(1.3)

Substituting (1.2) into (1.3) we get:

$TS(t_n)= V(t_{n-1}) + \frac{L_n}{r\beta \sum_{i=1}^{n-1} \alpha_i} + \frac{L_n}{\alpha_n r}$

 

$$= V(t_{n-1}) + \frac{L_n}{r}\left(\frac{1}{\beta\sum_{i=1}^{n-1}\alpha_i} + \frac{1}{\alpha_n}\right)$$

$$TS(t_n) = V(t_{n-1}) + \frac{L_n}{r}\left(\frac{\alpha_n + \beta\sum_{i=1}^{n-1}\alpha_i}{\alpha_n\beta\sum_{i=1}^{n-1}\alpha_i}\right)$$

$$= V(t_{n-1}) + \frac{\alpha_{n-1}L_n}{\alpha_{n-1}r}\left(\frac{\alpha_n + \beta\sum_{i=1}^{n-1}\alpha_i}{\alpha_n\beta\sum_{i=1}^{n-1}\alpha_i}\right) \text{----------(1.4)}$$

Substituting (1) into (1.4) we get:

$$TS(t_n) = V(t_{n-1}) + \frac{L_n}{\alpha_{n-1}r}\frac{L_{n-1}}{L_n} = V(t_{n-1}) + \frac{L_{n-1}}{\alpha_{n-1}r} = TS(t_{n-1}) \text{--------(1.5)}$$

By using induction we can show that all timestamps is equal to $\frac{L}{r}$, which is the timestamp of the first packet , i.e. $TS(t_n) = TS(t_{n-1}) = \ldots\ldots = TS(t_2) = TS(t_1) = \frac{L}{r}$

packet scheduling algorithms [9-12]here studied, showing the increment to the fluid delay bound introduced by each one WFQT

| **Techniques** | **Delay bound on node** |
|---|---|
| WFQ | $(\sigma_i + L_i)/\rho_i + L_{\max}/C$ |
| SCFQ | $(\sigma_i + (V-1)L_i)/\rho_i + L_{\max}/C$ |
| STFQ | ---------- |
| VC | $(\sigma_i + L_i)/\rho_i + L_{\max}/C$ |
| $W^2FQ$ | $(\sigma_i + L_i)/\rho_i + L_{\max}/C$ |
| $W^2FQ+$ | ---------- |

Table 1- Known delay bounds at a node, assuming the incoming traffic shaped by a leaky bucket of parameters $(\sigma_i,\rho_i)$ and an assigned bandwidth not smaller than the token rate of the leaky bucket

## 4. Conclusion and Simulation Results

In the simulated scenario, 3 traffic flows (1, 2 and 3) were considered, as well as a server with a capacity of 3 Kbytes/sec which assigned 1 Kbyte/sec to each flow. The incoming traffic of flow 1, used as test flow, was shaped by a leaky bucket of parameters ($\sigma$ = 500 Kbytes; $\rho$ = 1 Kbyte/sec).

**References**

1. Ivan Marsic,"Computer Networks, Performance and Quality of Service" Rutgers University, the state university of New jersey, 2010.
2. Rodrigo Sierra. Master Thesis, Internetworking"Fair queuing in data networks" , 2002.
3. Mohammed Hawa,"Stochastic Evaluation of fair Scheduling with applications of QoS in broadband wireless access networks", Doctor of Philosophy, the University of Kansas,USA, August 2003.
4. A.V. Aho, J.E. Hopcroft and J.D. Ullman , "The Design and Analysis of Computer Algorithms", Addison-Wesley, Reading, Mass., 1974.

 

5. Arik Chikvashvili , Master thesis , M-WF$^2$Q: A WF2Q Algorithm Running At O(logN) Complexity Under Bursty Arrivals , December 2009.
6. Mark Richard Stemm, Ph.D Thesis , "An n Network Measurement Architecture for Adaptive Applications", GRADUATE DIVISION of the UNIVERSITY of CALIFORNIA at BERKELEY , USA , 1999.
7. Dimitrios P. Pezaros,Ph.D Thesis ,"Network Traffic Measurement for the Next Generation Internet" , Computing Department , Lancaster University , England , August 2005.
8. H.Shi, and H.Sethu. Greedy fair queueing: A goal-oriented strategy for fair real-time packet scheduling. In Proceedings of the Real-Time Systems Symposium (RTSS) December 2003.
9. I. Stoica and H. Abdel-Wahab. Earliest eligible virtual deadline _rst: A exible and accurate mechanism for proportional share resource allocation. Technical Report TR- 95-22, Old Dominion University, November 1995.
10. I. Stoica, H. Zhang "Exact emulation of an output queuing switch by a combined input/output queuing switch" In IWQoS '98, May 1998.
11. L. Ashkenazi and H. Levy, Burstiness and Fairness in Packet Scheduling. In HET-NETs '04, July 2004.
12. N. McKeown B. Prabhakar. "On the Speedup Required for Combined Input and Output Queued Switching" In Computer Systems Technical Report CSL-TR-97-738. November 1997.
13. Abhay Kumar J. Parekh , "A Generalized Processor Sharing Approach to Flow Control In Integrated Services Networks" Doctor of Philosophy, Massachusetts Institute of Technology , 1992.
14. Ion Stoica , Scott Shenker, and Hui Zhang "Core-Stateless Fair Queueing:A Scalable Architecture to Approximate Fair Bandwidth Allocations in High Speed Networks" , SIGCOMM'98.
15. S. Golestani. A self-clocked fair queuing scheme for broadband applications. In Proceedings of IEEE INFOCOM '94, pages 636-646, Toronto, CA, June 1994.
16. J. C. R. Bennett, H. Zhang, "WF2Q: Worst-case fair weighted fair queueing", in Proc. IEEE INFOCOM '96, San Francisco, CA, Mar. 1996
17. A self-clocked fair queueing scheme for broadband applications", S.JamallodinGolestaniBellcore , 445 south street Morristown , NJ 079690-6438 , in Proc. INFOCOM '94, Apr. 1994. , 0774-166X/94 $3.00 , IEEE.
18. J. C. R. Bennett, H. Zhang, "Hierarchical Packet Fair Queueing Algorithms", IEEE/ACM Transactions on networking,Vol.5.,No.5,October1997.
    Victor A. Clincy, and Ajay Sitaram "IP Queuing Analysis" ,College of Science and Math Computer Science and Information Systems Department, Kennesaw State University , Kennesaw, Georgia 30144 vclincy@kennesaw.edu, 770-420-4440
19. P. Goyal, H. M. Vin, H. Cheng, "Start-time Fair Queueing: A Scheduling Algorithm for Integrated Services Packet Switching Networks", IEEE/ACM Transactions on Networking, Vol. 5, No. 5, October 1997.
20. D. Stidialis, A. Varma, "Efficient Fair Queueing Algorithms for Packet-Switched Networks", IEEE/ACM Trasactions on Networking, Vol. 6, No. 2, April 1998.
21. Carey Williamson ,"Internet Traffic Measurement" , Department of Computer Science , University of Calgary , November 24, 2001
22. Oumar Ndiaye ,"An Efficient Implementation of a Hierarchical Weighted Fair Queue Packet Scheduler " , Massachusetts Institute of Technology May, 1994 , master thesis.
23. J. Davin and A. Heybey, "A simulation study of fair queueing and policy enforcement", Computer Communications Review, vol. 20, no. 5, Oct. 1990.


 