Sliding weighted fair queueing scheme for real-time applications

Y.-S. Yen, W. Chen, J.-C. Zhhuang and H.-C. Chao

Abstract: Weighted fair queueing (WFQ) is a popular scheme because of its guaranteed bandwidth and bounded delay. To make WFQ more flexible, sliding weighted fair queueing (SWFQ) is proposed by combining priority-driven and share-driven scheduling for a real-time network. In this proposal, SWFQ can balance the share and priority-driven characteristics and can allow the application of WFQ to various network environments. This queueing algorithm can also be applied to either IPv4 or IPv6 networks as long as it can provide an adequate quality of service (QoS) support.

1 Introduction

Scheduling or queueing has been proposed in the literature for many years. Examples include weighted fair queue (WFQ) [1], priority queue (PQ) [2], weighted round robin (WRR) [3], delay earliest-due-date (Delay-EDD), etc. These algorithms provide different services for different priority levels. The type of service (ToS) field in the IPv4 header and the traffic class field in the IPv6 header are used to distinguish the different types of traffic to provide specific services.

The simplest and most widely used scheduling method is called 'first come first served' (FCFS) [2]. Packets arriving from different flows are treated equally by placing them into the queue. This type of method service does not support service differentiation and the differentiation is too strong in the case of priority scheduling. Thus, FCFS cannot support priority fairness. Alternatively, priority queueing service packet is based on the packet priorities. High priority packets will be serviced first regardless of their arrival time. Then, low priority packets will often be greatly delayed or not sent at all. This scheduling method will not satisfy the requirements of future network protocols. Fair queueing [1] provides better quality of service for high priority packets and the bandwidth is shared by both low and high priority packets, which allows a high transmission quality.

WFQ was publicly adopted because of its guaranteed bandwidth and combined access control properties. However, the session delay bound is related to the service sharing allocation. Therefore, high service sharing has a lower delay bound. A priority-driven scheduling system that would give WFQ better guaranteed bandwidth and bounded delay was attempted. Unfortunately, priority-driven scheduling and WFQ could not be used at the same time because the traditional priority-driven scheduling system serves the higher priority packets first. If higher priority packets are not sent, the lower priority packets must wait until a high priority packet is received. Thus, only high priority packets would occupy the entire bandwidth. In other words, if priority-driven scheduling and WFQ are used at the same time, the bandwidth and bounded delay guarantee could not be preserved. As a consequence, the sliding window concept was developed. In this paper, we propose a novel queueing algorithm combining WFQ and a sliding window, called sliding weighted fair queueing (SWFQ).

WFQ is like having several entries. When a packet arrives it is classified by the classifier and assigned to one of the entries. The entry is to a queue that is served together with some other in weighted round-robin order. In our proposed scheme, the packet service order is determined by the WFQ scheduler. Then the packets will be examined if their virtual finish time falls within the window. The server will choose the packet with the highest priority and send it out; the virtual finish time of the packets that are smaller than the sent packet will increase one unit in the current window. A more detailed explanation is given in the following Section.

In the sliding window, the session bandwidth is guaranteed and the bounded delay is adjusted according to the session sharing and priority. SWFQ decouples the effect of the service sharing limit in WFQ. In other words, a session with low sharing but with a high priority could still receive a small delay. The proposed queueing algorithm can be applied to either IPv4 or IPv6 networks [4, 5]. The IPv4 type of service field or IPv6 traffic class field can label the session priority order. In this paper, the IPv6 network is taken as an example.

2 Delay bounds of both GPS and WFQ

Generalised processor sharing (GPS) [6] is an idealised fluid-flow system that serves packets in separate logical queues. GPS can visit each nonempty queue in turn and serve an infinitesimally small amount of data from each queue. Whenever communication sessions have data to be served, they are associated with a pre-allocated share (or weight) and receive service in proportion to their share. GPS is a share-driven scheduling scheme.

GPS is an idealised model of the queueing system that does not transmit packets as entities and therefore it cannot

© IEE, 2005
IEE Proceedings online no. 20045139
doi:10.1049/ip-com:20045139
Paper first received 28th July 2004 and in final revised form 5th January 2005

Y.-S. Yen, J.-C. Zhhuang and H.-C. Chao are with the Department of Electrical Engineering, National Dong Hwa University, Hualien, Taiwan, Republic of China
W. Chen is with the Department of Information Management, National Dong Hwa University, Hualien, Taiwan, Republic of China
E-mail: hcc@mail.ndhu.edu.tw

be implemented. Only one session can receive service at a time in an actual packet network system. A complete packet must be sent before another packet can be served. Many algorithms have been proposed to allow practical packet network systems that can emulate GPS. Among the proposed algorithms, the most popular is the weighted fair queueing (WFQ), also known as packet generalised processor sharing (PGPS) [6, 7]. WFQ is a packet scheduling technique allowing guaranteed bandwidth services. The purpose of WFQ is to let several sessions share the same link. WFQ is an approximation of GPS which, as the name suggests, is a generalisation of processor sharing (PS) [8]. Using this concept, virtual time can be used to measure the network progress in the ideal GPS server. Assume the virtual time function $V(t)$ as a function of the real-time t over the (real-)time interval $[\tau, T]$. Now suppose that the kth session i packet arrives at time a_i^k. Then in WFQ, the virtual packet completion time (virtual finish time) is defined as:

$$F_i^k = \max\{F_i^{k-1} \cdot V(a_i^k)\} + \frac{L_i^k}{\phi_i} \tag{1}$$

where F_i^k is the virtual finish time for the kth packet in session i. ϕ_i is the service sharing of session i and L_i^k is the packet size of the kth packet. The system sends out the packet with the smallest virtual finish time first. In [6], when every busy period session is in greedy mode for the system that starts from the beginning, the worst-case session delay could occur. If a session sends as many packets as possible at all times, the progress of packets that arrive from other sessions will be impeded. When the traffic that enters the network is shaped by the leaky-bucket model [9], the flow pattern is as follows:

$$A_i(\tau, t) \le \sigma_i + \rho_i(t-\tau) \quad \forall \le \tau \le t \tag{2}$$

The leaky bucket model is shown more clearly in Fig. 1. $A_i(\tau, t)$ is the amount of traffic in session i that leaves the leaky bucket and enters the network during the interval $[\tau, t]$. Tokens are generated at a fixed rate ρ_i, and packets can be released into the network only after the required number of tokens is acquired from the leaky bucket, containing σ_i bits worth of tokens at most.

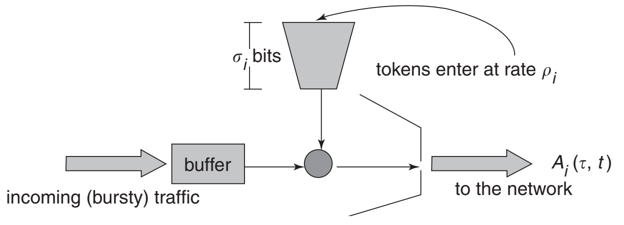


Fig. 1 *Leaky bucket model*

In [9], the delay bound for a session in a GPS signal server is:

$$D^* \le \frac{\sigma_i}{g_i} \tag{3}$$

where g_i is the guaranteed service rate of session i:

$$g_i = \frac{\phi_i}{\sum_{j=1}^{N} \phi_j} r \tag{4}$$

ϕ_i is the service share of session i in N sessions. r is the server service rate. In general, $g_i \ge \rho_i$, otherwise the session i queue will not be locally stable and may grow without limits. g_i is independent from the behaviour of other sessions. The delay bound of a session in a single node WFQ server is shown as follows:

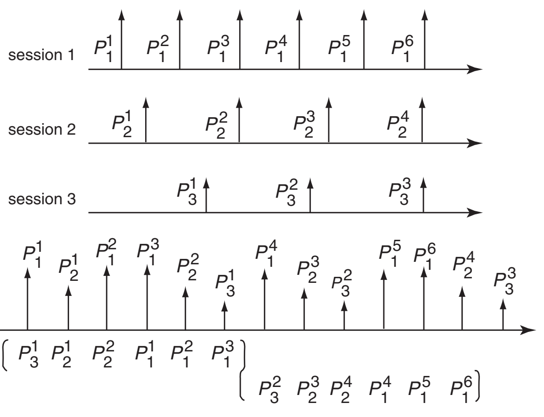
$$D^* \le \frac{\sigma_i}{g_i} + \frac{L_{\max}}{r} \tag{5}$$

where $L_{\max}$ is the maximum packet size among all sessions. From (5), a packet delay in a WFQ system is larger than in the corresponding GPS system by no more than the transmission time for sending one maximum sized packet. However, the delay bound depends on g_i.

3 Overview of priority based weighted fair queueing (PWFQ)

In [10], WFQ uses service share as the only factor for making packet scheduling decisions such that the delay bound and the share are tightly coupled. A priority based weighted fair queueing (PWFQ) scheduler is proposed for real-time networks to de-couple session delay and its allocated bandwidth share [11]. A communication session is needed to reserve a lower delay bound for a larger share. Therefore, fewer sessions are admitted to the system and the link bandwidth may not be utilised efficiently. In order to make WFQ more flexible, PWFQ introduces the priority attribute to the WFQ scheduler to better manage the delay bounds of various sessions. The method to integrate WFQ and the priority-driven scheme is to use a sliding window. In order to determine which packet should be sent out first within the window, fixed priority is used. As a result, packets in a session with high priority may achieve lower delay.

Figure 2 shows the PWFQ idea more clearly. We proposed the priority order from high to low as session 3, session 2 and session 1 in turn. The service sharing order from high to low is session 1, session 2 and session 3 in turn. P_i^k is the virtual finish time for the kth packet in session i. If WFQ is used, the order in which the packets are sent would be P_1^1, P_2^1, P_1^2, P_1^3, P_2^2, P_3^1, and so on. Using the PWFQ mechanism, packets P_1^1, P_2^1, P_1^2, P_1^3, P_2^2, P_3^1 all initially fall into the window. The server determines their service order according to the priority within the sliding window. If PWFQ is used, the order in which the packets are sent would be P_3^1, P_2^1, P_2^2, P_1^1, P_1^2, P_1^3,.... Since session 3 has the highest priority, P_3^1 will be sent first. Although the relative packet service order inside the window has been changed by priority levels from the share point of view, the number of packets sent within the sliding window by session 1 still receives enough bandwidth according to its share.



session 1
P_1^1 P_1^2 P_1^3 P_1^4 P_1^5 P_1^6
session 2
P_2^1 P_2^2 P_2^3 P_2^4
session 3
P_3^1 P_3^2 P_3^3
WFQ
P_1^1 P_2^1 P_1^2 P_1^3 P_2^2 P_3^1 P_1^4 P_2^3 P_3^2 P_1^5 P_1^6 P_2^4 P_3^3
PWFQ
P_3^1 P_2^1 P_2^2 P_1^1 P_1^2 P_1^3
P_3^2 P_2^3 P_2^4 P_1^4 P_1^5 P_1^6

Fig. 2 *Sliding window in PWFQ*

4 Sliding weighted fair queueing (SWFQ)

4.1 Introduction to SWFQ

To make WFQ more flexible, PWFQ introduces the priority attribute to the WFQ scheduler to better manage the delay bounds of various sessions. We propose a novel architecture, sliding weighted fair queueing (SWFQ), which is different from PWFQ. Figure 2 shows the PWFQ. The window size is fixed immediately after all the packets in the window are sent out. The window is then shifted back to the same size. Although the priority for session 3 is higher than that for sessions 1 and 2, the second packet of session 3 is the seventh one to be sent in the queue. We felt that this algorithm behaviour was not good enough for the performance of session 3.

Figure 3 shows the SWFQ. We proposed the priority order from high to low as session 3, session 2 and session 1 in turn. The service sharing order from high to low is session 1, session 2 and session 3 in turn. P_i^k is the virtual finish time for the kth packet of session i. If WFQ is used, the order in which packets are sent would be $P_1^1, P_2^1, P_1^2, P_1^3, P_2^2, P_3^1, \ldots$. Using the SWFQ mechanism, packets P_1^1, P_2^1, P_1^2, P_1^3, P_2^2, P_3^1 all initially fall into the window. The server determines their service order according to the priority within the sliding window. If SWFQ is used when the server sends out a packet, the virtual finish time of the packets would be smaller than that of the sent packet, and will increase one unit in current window. Then, the segment will be shifted back one unit in response to this. Thus, the second packet of Session 3 is the fourth packet to be sent in the queue. Then the order of packets sent would be P_3^1, P_2^1, P_2^2, P_3^2, P_2^3, P_1^1, $P_2^4, \ldots$. The SWFQ could therefore reduce the delay for high priority packets. However, a problem exists in which a packet with lower priority would cause a serious delay. Therefore, we will use an example to discuss this problem in the next Section. SWFQ reduces high priority packet delay and reduces the delay ratio for low priority packet greater than the delay ratio reduction for high priority packets.

A window defines a virtual time interval. All packets whose virtual finish times fall into the window are considered to have a similar finish time within the virtual time interval. Before the server selects the next packet to be sent, it must check the current window and compare all packets whose virtual finish times are inside the window. The server then chooses a packet with the highest priority. The highest priority packets are then placed in the window. The highest priority packets will be sent early in the sliding window based on the window size compared to the original schedule.

The number of eligible packets is controlled at any given time to guarantee the available bandwidth for any session by adjusting the size of the sliding window for all packets whose virtual finish times fall inside the window. To reduce the delay for high priority sessions, priority-based scheduling is used for all packets inside the window. In this way, both the share and delay are effectively controlled at the same time. The balance between share-driven and priority-driven schemes is matched to the window size. If the window size is too large, the session service share may not be achieved. If the window size is too small, the priority system will be violated.

In this mechanism, the sliding window size has a big impact on packet delay. In this mechanism, the window size is set to infinity. When all packets arrive in this system, they are all eligible in this case. The scheduler depends on the session priority that schedules them. This makes the scheduler a priority-driven scheduler. Another extreme window size is zero. In this case, the algorithm behaves exactly like a WFQ scheduler and packet priority is not considered at all.

The unique feature that the SWFQ algorithm has is that it strikes a balance between priority-driven and WFQ (or share-driven) scheduling. By adjusting the window size, the sliding window mechanism can integrate the share and priority. If the window size is appropriate, a high-priority, low-share session will have enough lead-time to be served first. Therefore, the packet delay can be managed better. Alternatively, a low-priority, high-share packet also receives enough service. Therefore, the minimum bandwidth guarantee can be obtained.

Figure 4 shows a SWFQ flow chart. The SWFQ processing program is as follows:

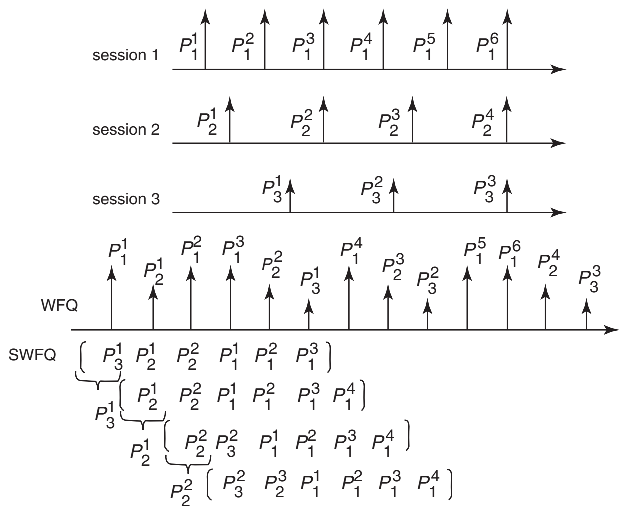

session 1
P_1^1 P_1^2 P_1^3 P_1^4 P_1^5 P_1^6
session 2
P_2^1 P_2^2 P_2^3 P_2^4
session 3
P_3^1 P_3^2 P_3^3
WFQ
P_1^1 P_2^1 P_1^2 P_1^3 P_2^2 P_3^1 P_1^4 P_2^3 P_3^2 P_1^5 P_1^6 P_2^4 P_3^3
SWFQ
[P_3^1 P_2^1 P_2^2 P_1^1 P_1^2 P_1^3]
P_3^1 [P_2^1 P_2^2 P_1^1 P_1^2 P_1^3 P_1^4]
P_2^1 [P_2^2 P_3^2 P_1^1 P_1^2 P_1^3 P_1^4]
P_2^2 [P_3^2 P_2^3 P_1^1 P_1^2 P_1^3 P_1^4]

Fig. 3 *Sliding window in SWFQ*

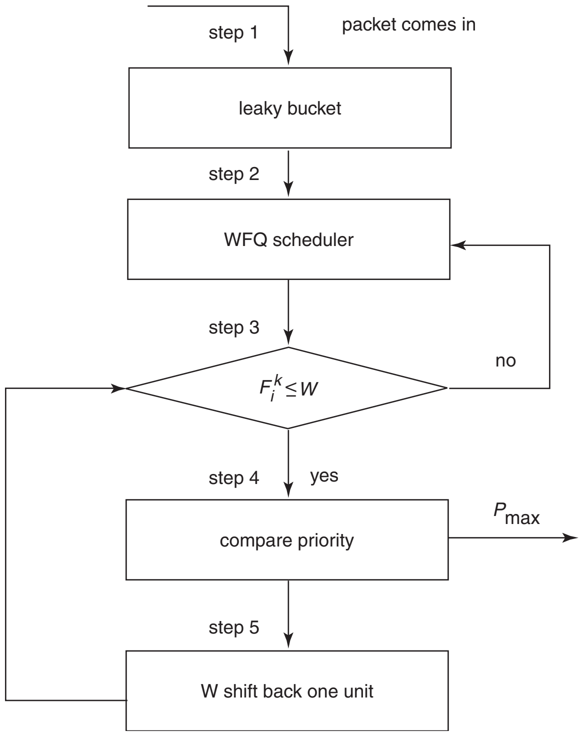


Fig. 4 *Flow chart for SWFQ*
W: window size
F_i^k: virtual finish time for the kth packet of session i