A Novel Sliding Weighted Fair Queueing Scheme for Multimedia Transmission

Yun-Sheng Yen, Weimin Chen*, Jia-Cheng Zhhuang and Han-Chieh Chao
Department of Electrical Engineering
*Department of Information Management
National Dong Hwa University
Hualien, Taiwan, Republic of China
E-mail: d9223008@em92.ndhu.edu.tw; m9123012@em91.ndhu.edu.tw; wmchen@mail.ndhu.edu.tw; hcc@mail.ndhu.edu.tw

Abstract

Weighted Fair Queueing (WFQ) is a popular scheme because of its guaranteed bandwidth and bounded delay. To make WFQ more flexible, we propose Sliding Weighted Fair Queueing (SWFQ) by combining priority-driven and share-driven scheduling for a real-time network. In our proposal, SWFQ can balance the share and priority-driven characteristics and can allow WFQ application 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 are: 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 service 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]. This type of method services packets is based on their arrival time. If a packet comes first into the server, it will be serviced first. When a high priority packet comes into the server later than a low priority packet, the high priority packet must wait until the low priority packet is serviced. Thus, FCFS cannot support priority fairness. On the other hand, Priority Queueing services packets is based on their priority. High priority packets will be serviced first regardless of their arrival time. Then again, 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 [4] 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 boundary is related to the service sharing allocation. Therefore, high service sharing has a lower delay boundary. 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 proposed a novel Queueing algorithm combining WFQ and a sliding window, called Sliding Weighted Fair Queueing (SWFQ). 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, if 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 [5, 6]. The IPv4 type of service field or IPv6 traffic class field can label the session priority order. In this paper, IPv6 network is taken as an example.

The remainder of this paper is organized as follows: Section 2 presents an overview of Priority-
based Weighted Fair Queueing (PWFQ). Section 3 presents the details of the proposed method, a novel Queueing algorithm with Sliding Weighted Fair Queueing (SWFQ). Section 4 analyzes the delay of PWFO and SWFQ. Section 5 shows the determining method of priority. Section 6 describes our simulation and results. Section 7 presents our conclusions.

2. An Overview of Priority-based Weighted Fair Queueing (PWFQ)

In [7], 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 has been proposed for real-time network to decouple session delay and its allocated bandwidth share [8]. 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 efficiently utilized. 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 idea, which integrated 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.

In [7], 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 has been proposed for real-time network to decouple session delay and its allocated bandwidth share [8]. 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 efficiently utilized. 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 idea, which integrated 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.

3. Sliding Weighted Fair Queueing (SWFQ)

To make WFQ more flexible, PWFQ introduces the priority attribute to the WFQ scheduler to better manage the delay boundaries of various sessions. We proposed a novel architecture, Sliding Weighted Fair Queueing (SWFQ), which is different from PWFQ. Figure 1 shows the PWFQ. The window size is fixed right after all of 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 will be the seventh one to be sent in the queue. We felt that this algorithm behavior was not good enough for the performance of session 3.

Figure 2 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 $k^{th}$ packet of session $i$. If WFQ is used, the order in which packets are sent would be $P_1^1, P_2^1, P_1^2, P_2^2, P_1^3, P_2^3, \ldots$. Using the SWFQ mechanism, packets $P_1^1, P_2^1, P_1^2, P_2^2, P_1^3, P_2^3, P_1^4, P_2^4, \ldots$ 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 $P_1^1, P_2^1, P_2^2, P_1^3, P_2^3, \ldots$. Since session 3 has the highest priority, $P_1^1$ will be sent first. Although the relative packet service order inside the window has been changed by priority levels from the share point view, the number of packets sent within the sliding window by session 1 still receives enough bandwidth according to its share.

![Figure 1 Sliding Window in PWFQ](image-url)

Figure 1 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 $k^{th}$ packet of session $i$. If WFQ was used, the order in which the packets are sent would be $P_1^1, P_2^1, P_1^2, P_2^2, P_1^3, P_2^3, \ldots$. Using the PWFQ mechanism, packets $P_1^1, P_2^1, P_1^2, P_2^2, P_1^3, P_2^3, P_1^4, P_2^4, \ldots$ 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 $P_1^1, P_2^1, P_2^2, P_1^3, P_2^3, \ldots$. Since session 3 has the highest priority, $P_1^1$ will be sent first. Although the relative packet service order inside the window has been changed by priority levels from the share point view, the number of packets sent within the sliding window by session 1 still receives enough bandwidth according to its share.

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.

4. Analytical Approach for PWFO and SWFQ Delay

In this section we will use an example from the problem in Section 3. The delay for low priority packet must be increased because SWFQ reduces the delay for high priority packets. We will use the following example to explain how the delay for high priority packets is reduced without increasing the delay for the low priority packets. Suppose that there are three sessions, and the priority order from high to low is sessions 3, 2 and 1 in turn. The service share order from high to low is session 1, 2 and 3 in turn. \( S_1, S_2 \) and \( S_3 \) are the traffic for sessions 1, 2 and 3. The ratios for \( S_1, S_2 \) and \( S_3 \) are 5:3:2. Figure 4.3 shows the WFQ service order.

We can calculate every session packet departure time in PWFO and SWFQ. The PWFO and SWFQ session arrival times are equal. However, only packet departure time needs to be calculated in order to determine the discrepancy of packet delay between PWFO and SWFQ. First, the packet departure time in PWFO is calculated. Figure 4 shows the PWFO. Since session 3 has the highest priority in the window, \( S_3 \) will be sent first. The \( tp^1 \) is the departure time for the \( nth \) \( S_i \) in session \( i \). The \( r \) is the link rate. The departure time for sessions 1, 2 and 3 in the PWFO is calculated as follows:

\[
\begin{align*}
\text{session 1: } & \quad tp^1_1 = \frac{S_1}{r} \delta, \quad tp^3_1 = \left( \frac{2S_1}{r} + \frac{S_2}{r} + \frac{S_3}{r} \right) \delta, \\
\text{session 2: } & \quad tp^1_j = \left( \frac{3S_1}{r} + \frac{2S_2}{r} + \frac{2S_3}{r} \right) \delta, \\
\text{session 3: } & \quad tp^2_3 = \frac{nS_3}{r} + \left( \frac{n-1}{r}S_2 + \frac{n-1}{r}S_1 \right) \delta.
\end{align*}
\]

(1)